

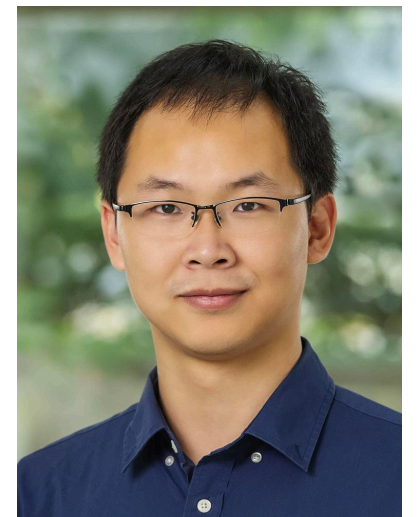
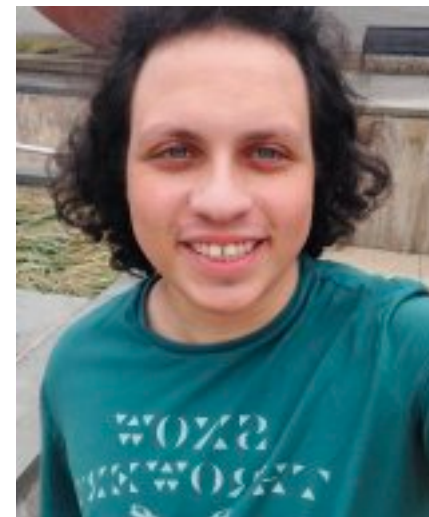
# Parameter-Free Adaptive Optimization



**SAMSUNG**  
**Research**




**PRINCETON**  
**UNIVERSITY**





Konstantin Mishchenko  
Based on joint work with  
Aaron Defazio, Ahmed Khaled, Chi Jin

# Numerical results

<https://github.com/facebookresearch/dadaptation>

 Fork 16


 Star 455



downloads 1M

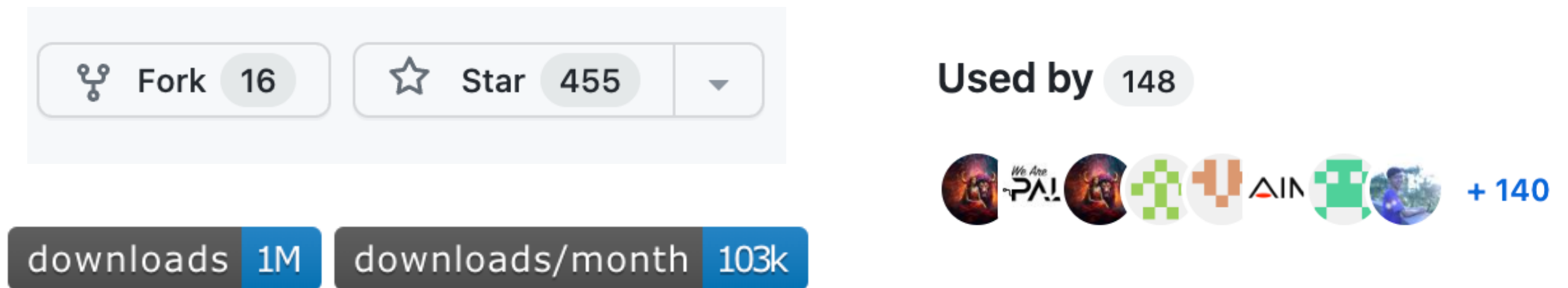
downloads/month 103k

Used by 148

 + 140

# Numerical results

<https://github.com/facebookresearch/dadaptation>



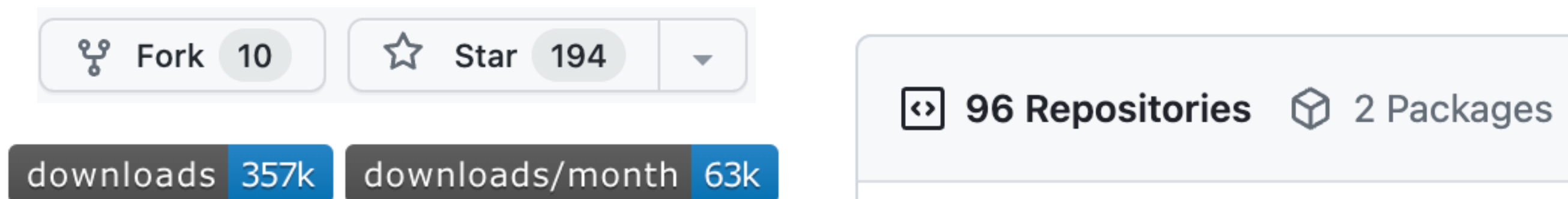
GitHub repository statistics for `facebookresearch/dadaptation`:

- Fork: 16
- Star: 455
- Used by: 148
- downloads: 1M
- downloads/month: 103k

Users using the repository (including avatars):

- Avatar 1
- Avatar 2
- Avatar 3
- Avatar 4
- Avatar 5
- Avatar 6
- Avatar 7
- Avatar 8
- + 140

<https://github.com/konstmish/prodigy>



GitHub repository statistics for `konstmish/prodigy`:

- Fork: 10
- Star: 194
- downloads: 357k
- downloads/month: 63k

Repository and Package counts:

- 96 Repositories
- 2 Packages

# Talk plan

1. The motivation
2. The prehistory
3. DoG and DoWG
4. D-Adaptation
5. Prodigy
6. Conclusion

# Adam and learning rate

## Adam: A Method for Stochastic Optimization

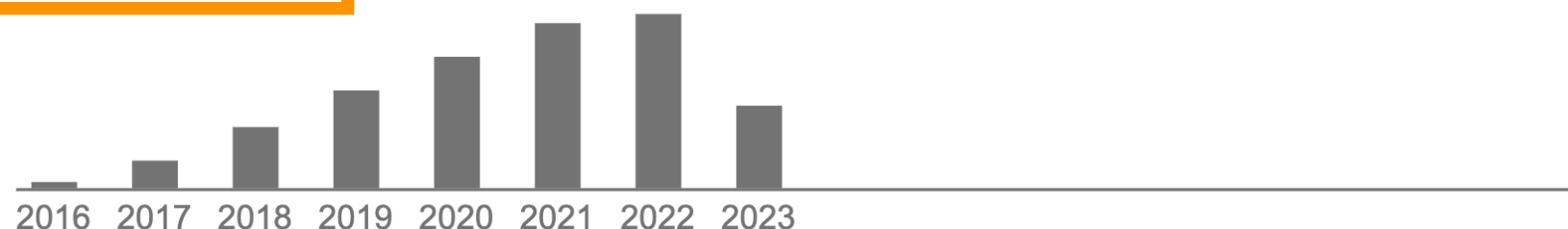
Authors Diederik P. Kingma, Jimmy Ba

Publication date 2014/12/22

Journal Proceedings of the 3rd International Conference on Learning Representations (ICLR)

Description We introduce Adam, an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments. The method is straightforward to implement, is computationally efficient, has little memory requirements, is invariant to diagonal rescaling of the gradients, and is well suited for problems that are large in terms of data and/or parameters. The method is also appropriate for non-stationary objectives and problems with very noisy and/or sparse gradients. The hyper-parameters have intuitive interpretations and typically require little tuning. Some connections to related algorithms, on which Adam was inspired, are discussed. We also analyze the theoretical convergence properties of the algorithm and provide a regret bound on the convergence rate that is comparable to the best known results under the online convex optimization framework. Empirical results demonstrate that Adam works well in practice and compares favorably to other stochastic optimization methods. Finally, we discuss AdaMax, a variant of Adam based on the infinity norm.

Total citations Cited by 150277



# Adam and learning rate

```
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
```

# Adam and learning rate

```
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
```

# Adam and learning rate

```
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
```

**But** it can depend on the batch size:

```
lr = lr_base * (batch_size / batch_size_base) ** 0.5
```

<https://github.com/huggingface/pytorch-image-models>



# Adam and learning rate

```
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
```

**But** it can depend on the batch size:

```
lr = lr_base * (batch_size / batch_size_base) ** 0.5
```

<https://github.com/huggingface/pytorch-image-models>

**And** on image resolution:

```
sched.G_lr_rate_dict = {128: 0.0015, 256: 0.002, 512: 0.003, 1024: 0.003}
```

<https://github.com/NVlabs/stylegan/blob/master/train.py>

# Adam and learning rate

```
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
```

**But** it can depend on the batch size:

```
lr = lr_base * (batch_size / batch_size_base) ** 0.5
```

<https://github.com/huggingface/pytorch-image-models>

**And** on image resolution:

```
sched.G_lr_rate_dict = {128: 0.0015, 256: 0.002, 512: 0.003, 1024: 0.003}
```

<https://github.com/NVlabs/stylegan/blob/master/train.py>

**And** on the network:

$G_{lr}=5e-5$

$D_{lr}=2e-4$

<https://github.com/ajbrock/BigGAN-PyTorch>

# Adam and learning rate

optimi

lr=0.001)

But it

lr = lr

**The goal:**  
**an automatic scheduler**

e-models

And on

sched.G\_lr

24: 0.003}

<https://github.com/NVlabs/stylegan/blob/master/train.py>

And on the network:


G\_lr=5e-5

D\_lr=2e-4

<https://github.com/ajbrock/BigGAN-PyTorch>

# A recap of Adam

**Problem:**  $\min_{x \in \mathbb{R}^p} f(x)$



**Parameters**

# A recap of Adam

---

## Algorithm 1 Adam optimizer

---

```
1: Input:  $x_0, \beta_1 \in [0, 1)$  (default 0.9),  $\beta_2 \in [0, 1)$  (default 0.999),  
2:  $\gamma_k$  (default 0.001),  $\epsilon$  (default  $10^{-8}$ )  
3: for  $k = 1$  to  $n$  do  
4:    $g_k \in \partial f(x_k)$   
5:    $m_{k+1} = \beta_1 m_k + (1 - \beta_1)g_k$   
6:    $v_{k+1} = \beta_2 v_k + (1 - \beta_2)g_k^2$   
7:    $x_{k+1} = x_k - \gamma_k \frac{m_{k+1}}{\sqrt{v_{k+1}} + \epsilon}$   
8: end for
```

---

**Compute the gradient**

**We omit stochasticity for simplicity**

# A recap of Adam

---

## Algorithm 1 Adam optimizer

---

1: **Input:**  $x_0, \beta_1 \in [0, 1)$  (default 0.9),  $\beta_2 \in [0, 1)$  (default 0.999),  
2:  $\gamma_k$  (default 0.001),  $\epsilon$  (default  $10^{-8}$ )  
3: **for**  $k = 1$  **to**  $n$  **do**  
4:      $g_k \in \partial f(x_k)$   
5:      $m_{k+1} = \beta_1 m_k + (1 - \beta_1) g_k$   
6:      $v_{k+1} = \beta_2 v_k + (1 - \beta_2) g_k^2$   
7:      $x_{k+1} = x_k - \gamma_k \frac{m_{k+1}}{\sqrt{v_{k+1}} + \epsilon}$   
8: **end for**

---

**Momentum estimate of gradients**

# A recap of Adam

---

## Algorithm 1 Adam optimizer

---

1: **Input:**  $x_0, \beta_1 \in [0, 1)$  (default 0.9),  $\beta_2 \in [0, 1)$  (default 0.999),  
2:  $\gamma_k$  (default 0.001),  $\epsilon$  (default  $10^{-8}$ )  
3: **for**  $k = 1$  **to**  $n$  **do**  
4:      $g_k \in \partial f(x_k)$   
5:      $m_{k+1} = \beta_1 m_k + (1 - \beta_1)g_k$   
6:      $v_{k+1} = \beta_2 v_k + (1 - \beta_2)g_k^2$   
7:      $x_{k+1} = x_k - \gamma_k \frac{m_{k+1}}{\sqrt{v_{k+1}} + \epsilon}$   
8: **end for**

---

**Coordinate-wise estimates of gradient magnitudes**

# A recap of Adam

---

## Algorithm 1 Adam optimizer

---

1: **Input:**  $x_0, \beta_1 \in [0, 1)$  (default 0.9),  $\beta_2 \in [0, 1)$  (default 0.999),  
2:  $\gamma_k$  (default 0.001),  $\epsilon$  (default  $10^{-8}$ )  
3: **for**  $k = 1$  **to**  $n$  **do**  
4:      $g_k \in \partial f(x_k)$   
5:      $m_{k+1} = \beta_1 m_k + (1 - \beta_1) g_k$   
6:      $v_{k+1} = \beta_2 v_k + (1 - \beta_2) g_k^2$   
7:      $x_{k+1} = x_k - \gamma_k \frac{m_{k+1}}{\sqrt{v_{k+1}} + \epsilon}$   
8: **end for**

---

**Estimate gradient and divide by its magnitude**



# A recap of Adam

---

## Algorithm 1 Adam optimizer

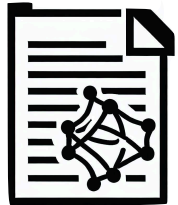
---

```
1: Input:  $x_0, \beta_1 \in [0, 1)$  (default 0.9),  $\beta_2 \in [0, 1)$  (default 0.999),  
2:  $\gamma_k$  (default 0.001),  $\epsilon$  (default  $10^{-8}$ )  
3: for  $k = 1$  to  $n$  do  
4:    $g_k \in \partial f(x_k)$   
5:    $m_{k+1} = \beta_1 m_k + (1 - \beta_1)g_k$   
6:    $v_{k+1} = \beta_2 v_k + (1 - \beta_2)g_k^2$   
7:    $x_{k+1} = x_k - \gamma_k \frac{m_{k+1}}{\sqrt{v_{k+1}} + \epsilon}$   
8: end for
```

---

**Many hyper parameters  
but they work *most* of the time**

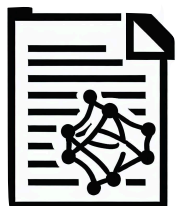
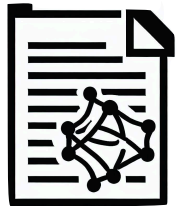
# Papers the talk is based on



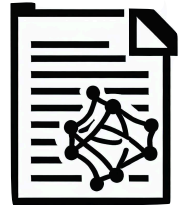
A. Defazio, K. Mishchenko

**Learning-Rate-Free Learning by D-Adaptation**

*ICML 2023 (released December 2022)*



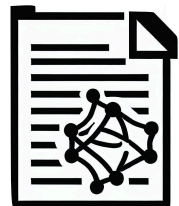
# Papers the talk is based on



A. Defazio, K. Mishchenko

**Learning-Rate-Free Learning by D-Adaptation**

*ICML 2023 (released December 2022)*



M. Ivgi, O. Hinder, Y. Carmon

**DoG is SGD's Best Friend: A Parameter-Free  
Dynamic Step Size Schedule**

*ICML 2023 (released February 2023)*



# Papers the talk is based on



A. Defazio, K. Mishchenko

**Learning-Rate-Free Learning by D-Adaptation**

*ICML 2023 (released December 2022)*



M. Ivgi, O. Hinder, Y. Carmon

**DoG is SGD's Best Friend: A Parameter-Free  
Dynamic Step Size Schedule**

*ICML 2023 (released February 2023)*



A. Khaled, K. Mishchenko, C. Jin

**DoWG Unleashed: An Efficient Universal Parameter-  
Free Gradient Descent Method**

*NeurIPS 2023*



# Papers the talk is based on



A. Defazio, K. Mishchenko  
**Learning-Rate-Free Learning by D-Adaptation**  
*ICML 2023 (released December 2022)*



M. Ivgi, O. Hinder, Y. Carmon  
**DoG is SGD's Best Friend: A Parameter-Free  
Dynamic Step Size Schedule**  
*ICML 2023 (released February 2023)*




A. Khaled, K. Mishchenko, C. Jin  
**DoWG Unleashed: An Efficient Universal Parameter-  
Free Gradient Descent Method**  
*NeurIPS 2023*



K. Mishchenko, A. Defazio  
**Prodigy: An Expediently Adaptive Parameter-Free Learner**  
*arXiv:2306.06101 June 2023*

# The setting

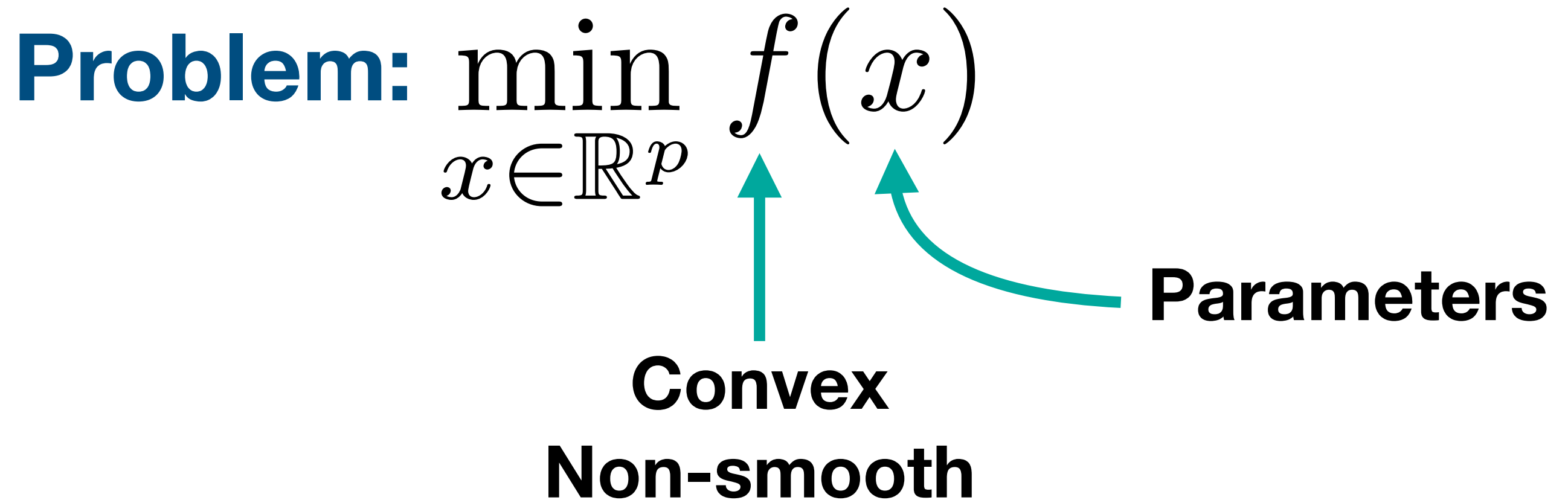
**Problem:**  $\min_{x \in \mathbb{R}^p} f(x)$



**Parameters**

# The setting

**Problem:**  $\min_{x \in \mathbb{R}^p} f(x)$



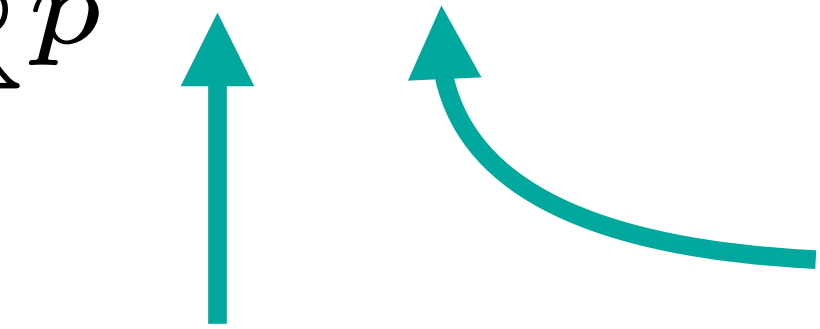
The diagram illustrates the components of the optimization problem and their properties. A teal arrow points from the word "Convex" to the function  $f$  in the expression  $f(x)$ . Another teal arrow points from the word "Parameters" to the variable  $x$  in the expression  $f(x)$ . The words "Convex" and "Non-smooth" are stacked vertically below the first arrow.

**Convex**  
**Non-smooth**

**Parameters**

# The setting

**Problem:**  $\min_{x \in \mathbb{R}^p} f(x)$



Convex

**Non-smooth**

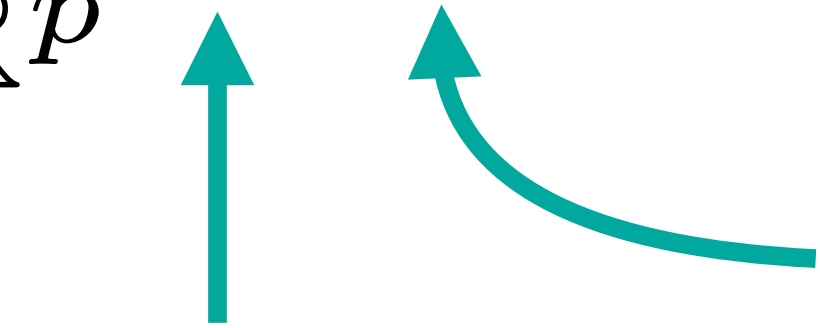
Parameters

$g \in \partial f(x)$



# The setting

**Problem:**  $\min_{x \in \mathbb{R}^p} f(x)$



**Convex**

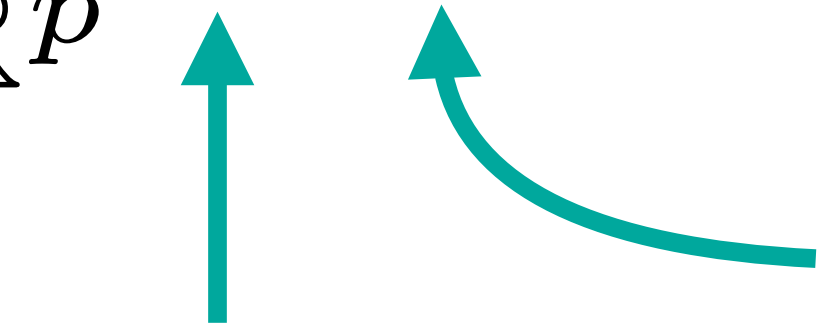
**Parameters**

Non-smooth

$$g \in \partial f(x) \implies \langle g, x - x_* \rangle \geq f(x) - f_*$$

# The setting

**Problem:**  $\min_{x \in \mathbb{R}^p} f(x)$



**Convex**

**Parameters**

**Non-smooth**

$$g \in \partial f(x) \implies \langle g, x - x_* \rangle \geq f(x) - f_*$$

$$g \in \partial f(x) \implies \|g\| \leq G$$

# **Why bother with convexity?**

**1. We're not tackling nonconvexity here**

# **Why bother with convexity?**

- 1. We're not tackling nonconvexity here**
- 2. Makes derivation much easier**

# **Why bother with convexity?**

- 1. We're not tackling nonconvexity here**
- 2. Makes derivation much easier**
- 3. We have to run a lot of experiments anyway**

# **Why bother with convexity?**

- 1. We're not tackling nonconvexity here**
- 2. Makes derivation much easier**
- 3. We have to run a lot of experiments anyway**
- 4. We do care a lot about non smoothness**

# Talk plan

1. The motivation
2. The prehistory
3. DoG and DoWG
4. D-Adaptation
5. Prodigy
6. Conclusion

# Subgradient Descent

---

## Algorithm 2 Subgradient Descent

---

```
1: Input:  $x_0, \gamma_k$   
2: for  $k = 0$  to  $n$  do  
3:    $g_k \in \partial f(x_k)$   
4:    $x_{k+1} = x_k - \gamma_k g_k$   
5: end for
```

---



# Subgradient Descent

---

## Algorithm 2 Subgradient Descent

---

```
1: Input:  $x_0, \gamma_k$   
2: for  $k = 0$  to  $n$  do  
3:    $g_k \in \partial f(x_k)$   
4:    $x_{k+1} = x_k - \gamma_k g_k$   
5: end for
```

---

# Subgradient Descent

---

## Algorithm 2 Subgradient Descent

---

```
1: Input:  $x_0, \gamma_k$ 
2: for  $k = 0$  to  $n$  do
3:    $g_k \in \partial f(x_k)$ 
4:    $x_{k+1} = x_k - \gamma_k g_k$ 
5: end for
```

---

# Subgradient Descent

---

## Algorithm 2 Subgradient Descent

---

```
1: Input:  $x_0, \gamma_k$   
2: for  $k = 0$  to  $n$  do  
3:    $g_k \in \partial f(x_k)$   
4:    $x_{k+1} = x_k - \gamma_k g_k$   
5: end for
```

---

Optimal stepsize:  $\gamma_k = \frac{D}{\sqrt{k}G}$

# Subgradient Descent

---

## Algorithm 2 Subgradient Descent

---

- 1: **Input:**  $x_0, \gamma_k$
  - 2: **for**  $k = 0$  **to**  $n$  **do**
  - 3:      $g_k \in \partial f(x_k)$
  - 4:      $x_{k+1} = x_k - \gamma_k g_k$
  - 5: **end for**
- 

Optimal stepsize:  $\gamma_k = \frac{D}{\sqrt{k}G},$

$$D = \|x_0 - x_*\|$$

# Polyak stepsize

$$x_{k+1} = x_k - \gamma_k g_k$$

# Polyak stepsize

$$x_{k+1} = x_k - \gamma_k g_k,$$
$$\gamma_k = \frac{f(x_k) - f(x_*)}{\|g_k\|^2}$$

The method has **amazing** guarantees

# Polyak stepsize

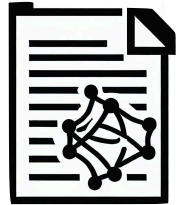
$$x_{k+1} = x_k - \gamma_k g_k,$$

$$\gamma_k = \frac{f(x_k) - f(x_*)}{\|g_k\|^2}$$

The method has **amazing** guarantees

But how do we know  $f(x_*)$ ?

# Polyak stepsize



B. Polyak

**Minimization of Unsmooth Functionals**

*Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki, 1987*





# Polyak stepsize



B. Polyak

## **Minimization of Unsmooth Functionals**

*Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki, 1987*



N. Loizou, S. Vaswani, I. Laradji, S. Lacoste-Julien

## **Stochastic Polyak Step-size for SGD: An Adaptive Learning Rate for Fast Convergence**

*AISTATS, 2021*

# Normalized Gradient by Shor

**Problem:**  $\min_{x \in \mathbb{R}^d} f(x)$       Oracle:  $g_k \in \partial f(x_k)$

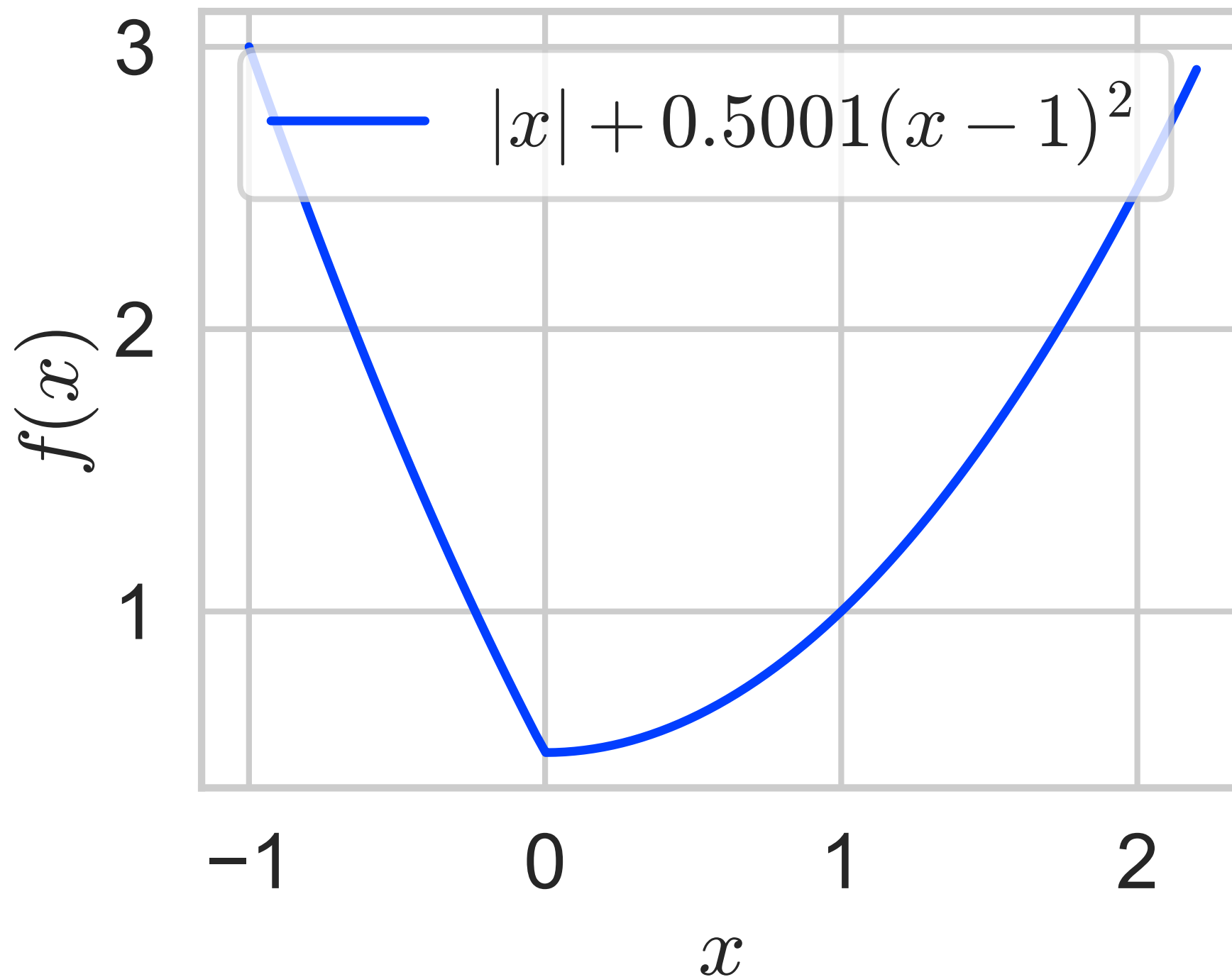
# Normalized Gradient by Shor

**Problem:**  $\min_{x \in \mathbb{R}^d} f(x)$     Oracle:  $g_k \in \partial f(x_k)$

$$x_{k+1} = x_k - \alpha_k \frac{g_k}{\|g_k\|}$$

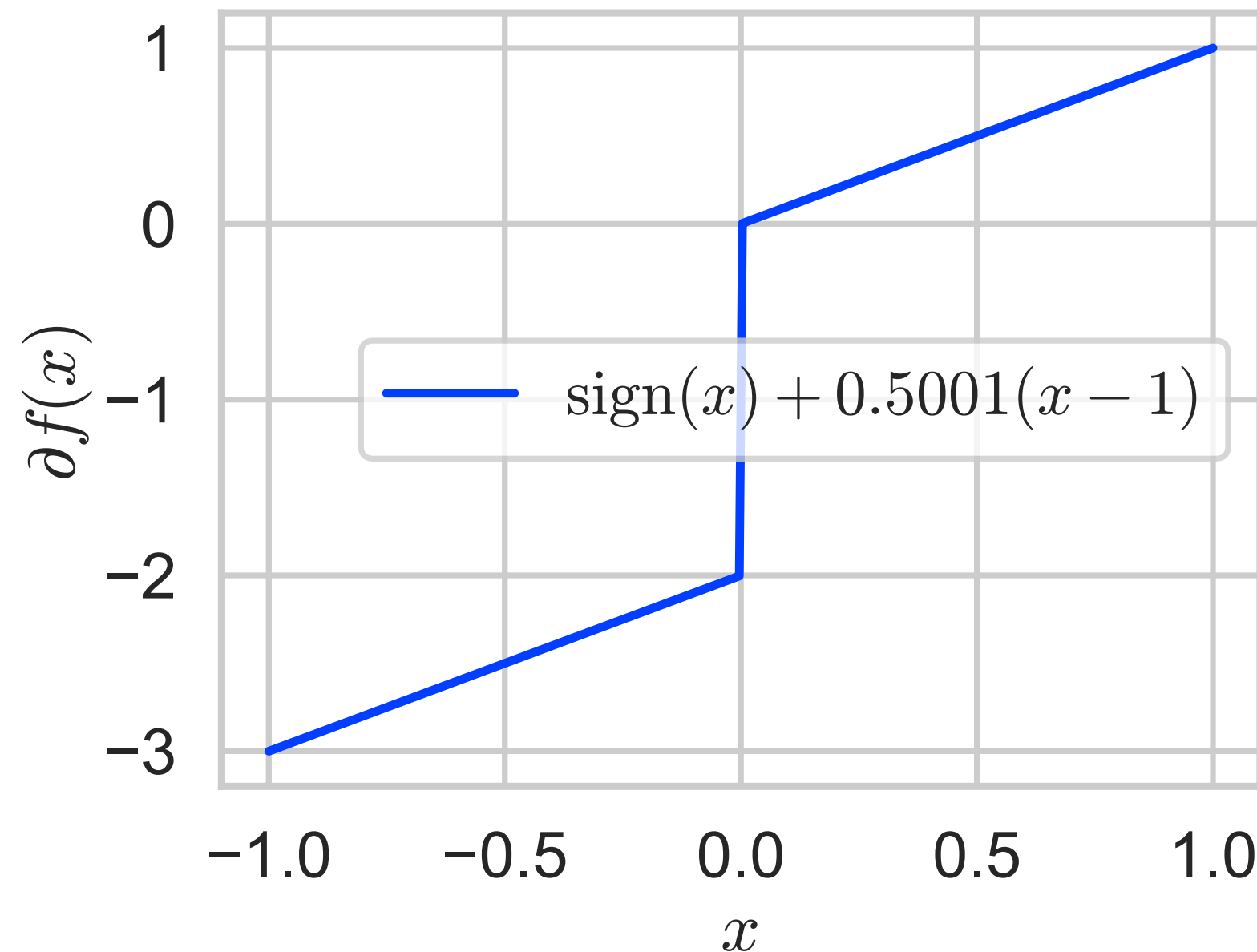
# Why normalization

$$f(x) = |x| + 0.5001(x - 1)^2$$



# Why normalization

$$f(x) = |x| + 0.5001(x - 1)^2$$



**subgradients:**

$$-2 \in \partial f(0)$$

$$-2 \cdot 10^{-4} \in \partial f(0)$$

# Why normalization

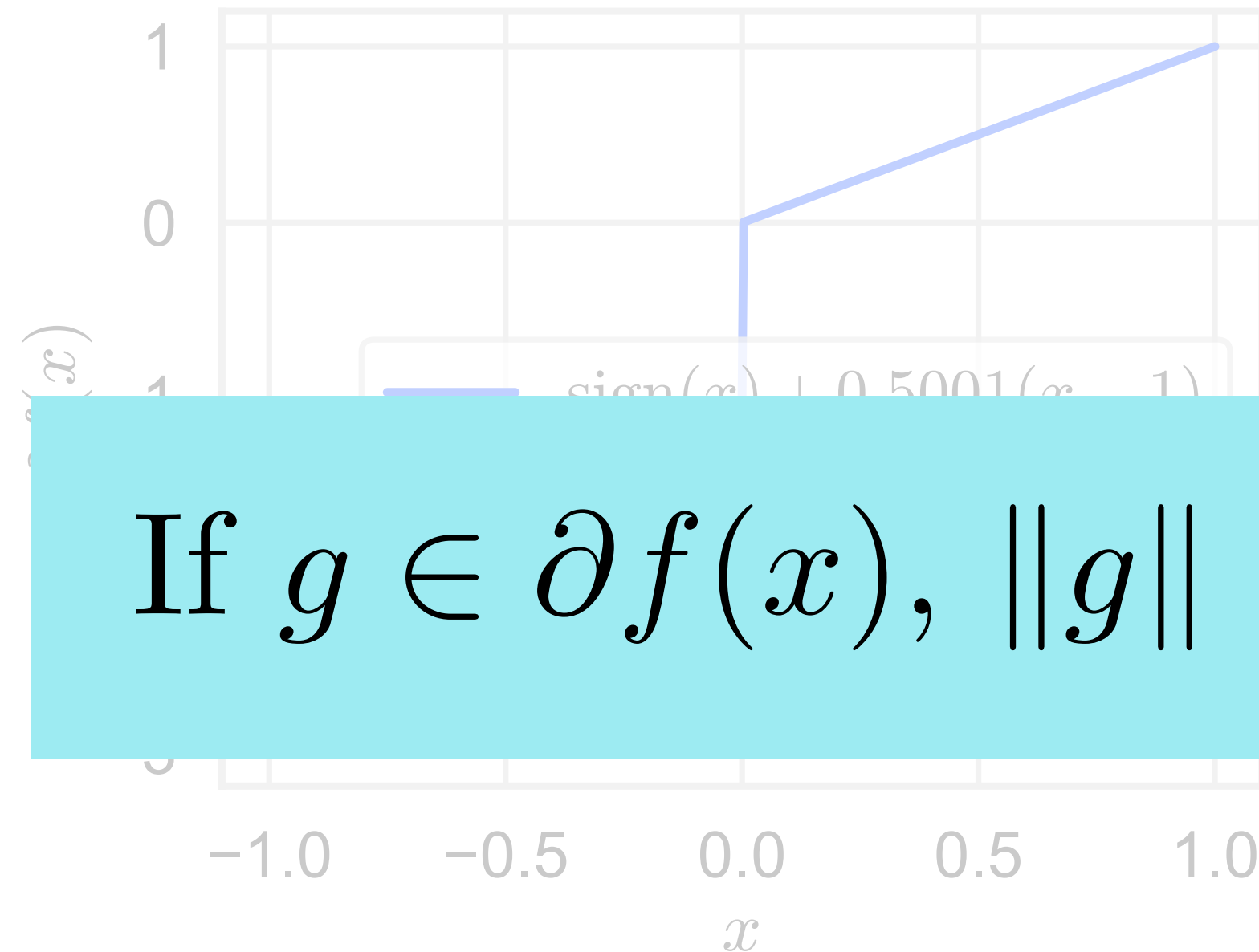
$$f(x) = |x| + 0.5001(x - 1)^2$$

subgradients:

$$-2 \in \partial f(0)$$

$$-2 \cdot 10^{-4} \in \partial f(0)$$

If  $g \in \partial f(x)$ ,  $\|g\|$  is meaningless



# Normalized Gradient by Shor

**Problem:**  $\min_{x \in \mathbb{R}^d} f(x)$     Oracle:  $g_k \in \partial f(x_k)$

$$x_{k+1} = x_k - \alpha_k \frac{g_k}{\|g_k\|}$$

# Normalized Gradient by Shor

**Problem:**  $\min_{x \in \mathbb{R}^d} f(x)$     Oracle:  $g_k \in \partial f(x_k)$

$$x_{k+1} = x_k - \alpha_k \frac{g_k}{\|g_k\|}$$

**Theorem.** For nonsmooth  $f$  with  $\|g_k\| \leq G$ ,  $\alpha_k = \frac{D}{\sqrt{k}}$

$$\min_{k \leq n} [f(x_k) - f_*] = \tilde{\mathcal{O}} \left( \frac{GD}{\sqrt{n}} \right).$$



# Normalized Gradient by Shor

**Problem:**  $\min_{x \in \mathbb{R}^d} f(x)$     Oracle:  $g_k \in \partial f(x_k)$

$$x_{k+1} = x_k - \alpha_k \frac{g_k}{\|g_k\|}$$

**Theorem.** For nonsmooth  $f$  with  $\|g_k\| \leq G$ ,  $\alpha_k = \frac{D}{\sqrt{k}}$

$$\min_{k \leq n} [f(x_k) - f_*] = \tilde{\mathcal{O}} \left( \frac{GD}{\sqrt{n}} \right).$$

**No need to know  $G$ !**

# Normalized Gradient by Shor

**Problem:**  $\min_{x \in \mathbb{R}^d} f(x)$     Oracle:  $g_k \in \partial f(x_k)$

$$x_{k+1} = x_k - \alpha_k \frac{g_k}{\|g_k\|}$$

**Theorem.** For nonsmooth  $f$  with  $\|g_k\| \leq G$ ,  $\alpha_k = \frac{D}{\sqrt{k}}$

$$\min_{k \leq n} [f(x_k) - f_*] = \tilde{\mathcal{O}} \left( \frac{GD}{\sqrt{n}} \right).$$

**No need to know  $G$ !**

**But it needs to decrease  $\alpha_k$  and requires  $D$**

# Normalized Gradient by Shor



N. Shor

**Minimization Methods for Non-Differentiable Functions**

*page 23, 1985*



# Normalized Gradient by Shor



N. Shor

**Minimization Methods for Non-Differentiable Functions**

*page 23, 1985*



B. Grimmer

**Convergence Rates for Deterministic and Stochastic  
Subgradient Methods Without Lipschitz Continuity**

*SIAM Journal on Optimization, 2019*

# Adagrad-Norm

$$\text{NGD: } x_{k+1} = x_k - \frac{D}{\sqrt{k}} \frac{g_k}{\|g_k\|}$$

# Adagrad-Norm

$$\text{NGD: } x_{k+1} = x_k - \frac{D}{\sqrt{k}} \frac{g_k}{\|g_k\|}$$

$$\text{Adagrad: } x_{k+1} = x_k - D \frac{g_k}{\sqrt{\sum_{t=0}^k \|g_t\|^2}}$$



Matthew Streeter, H. Brendan McMahan  
**Less Regret via Online Conditioning**  
*COLT 2010*

# Adagrad-Norm

$$\text{NGD: } x_{k+1} = x_k - \frac{D}{\sqrt{k}} \frac{g_k}{\|g_k\|}$$

$$\text{Adagrad: } x_{k+1} = x_k - D \frac{g_k}{\sqrt{\sum_{t=0}^k \|g_t\|^2}}$$

---

## Algorithm 1 Adam optimizer

---

- 1: **Input:**  $x_0, \beta_1 \in [0, 1)$  (default 0.9),  $\beta_2 \in [0, 1)$  (default 0.999),
  - 2:  $\gamma_k$  (default 0.001),  $\epsilon$  (default  $10^{-8}$ )
  - 3: **for**  $k = 1$  **to**  $n$  **do**
  - 4:      $g_k \in \partial f(x_k)$
  - 5:      $m_{k+1} = \beta_1 m_k + (1 - \beta_1) g_k$
  - 6:      $v_{k+1} = \beta_2 v_k + (1 - \beta_2) g_k^2$
  - 7:      $x_{k+1} = x_k - \gamma_k \frac{m_{k+1}}{\sqrt{v_{k+1}} + \epsilon}$
  - 8: **end for**
-

# Adagrad-Norm

$$\text{NGD: } x_{k+1} = x_k - \frac{D}{\sqrt{k}} \frac{g_k}{\|g_k\|}$$

$$\text{Adagrad: } x_{k+1} = x_k - D \frac{g_k}{\sqrt{\sum_{t=0}^k \|g_t\|^2}}$$

A bit more adaptive but it still **requires**  $D$



# Talk plan

1. The motivation
2. The prehistory
3. DoG and DoWG
4. D-Adaptation
5. Prodigy
6. Conclusion

# DoG and DoWG

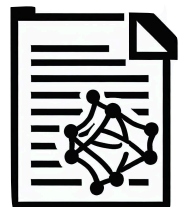


M. Ivgi, O. Hinder, Y. Carmon

**DoG is SGD's Best Friend: A Parameter-Free  
Dynamic Step Size Schedule**

*ICML 2023*

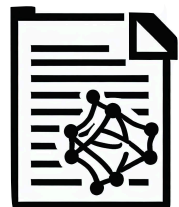
# DoG and DoWG



M. Ivgi, O. Hinder, Y. Carmon

**DoG is SGD's Best Friend: A Parameter-Free  
Dynamic Step Size Schedule**

*ICML 2023*



A. Khaled, K. Mishchenko, C. Jin

**DoWG Unleashed: An Efficient Universal Parameter-  
Free Gradient Descent Method**

*NeurIPS 2023*

# Distance-over-Gradients (DoG)

**Problem:**  $\min_{x \in \mathbb{R}^p} f(x)$

# Distance-over-Gradients (DoG)

**Problem:**  $\min_{x \in \mathbb{R}^p} f(x)$

**Setting:**  $g_k \in \partial f(x_k), \quad \|g_k\| \leq G$

$$D = \|x_0 - x_*\|$$

# Distance-over-Gradients (DoG)

**Problem:**  $\min_{x \in \mathbb{R}^p} f(x)$

**Setting:**  $g_k \in \partial f(x_k), \quad \|g_k\| \leq G$

$$D = \|x_0 - x_*\|$$

Subgradient stepsize:  $\gamma_k = \frac{D}{G\sqrt{k}}, \quad D = \|x_0 - x_*\|$

# Distance-over-Gradients (DoG)

**Problem:**  $\min_{x \in \mathbb{R}^p} f(x)$

**Setting:**  $g_k \in \partial f(x_k), \quad \|g_k\| \leq G$

$$D = \|x_0 - x_*\|$$

Subgradient stepsize:  $\gamma_k = \frac{D}{G\sqrt{k}}, \quad D = \|x_0 - x_*\|$

Estimate using Adagrad:

$$G\sqrt{k} \approx \sqrt{\sum_{k=0}^n \|g_k\|^2}$$


# Distance-over-Gradients (DoG)

**Problem:**  $\min_{x \in \mathbb{R}^p} f(x)$

**Setting:**  $g_k \in \partial f(x_k), \quad \|g_k\| \leq G$

$$D = \|x_0 - x_*\|$$

Subgradient stepsize:  $\gamma_k = \frac{D}{G\sqrt{k}}, \quad D = \|x_0 - x_*\|$

Estimate using Adagrad:

$$G\sqrt{k} \approx \sqrt{\sum_{k=0}^n \|g_k\|^2}$$

**We need to estimate  $D$   
to remove the stepsize**



# Distance-over-Gradients (DoG)

**Problem:**  $\min_{x \in \mathbb{R}^p} f(x)$

**Setting:**  $g_k \in \partial f(x_k), \quad \|g_k\| \leq G$

$$D = \|x_0 - x_*\|$$

Subgradient stepsize:  $\gamma_k = \frac{D}{G\sqrt{k}}, \quad D = \|x_0 - x_*\|$

Estimate using Adagrad:

$$G\sqrt{k} \approx \sqrt{\sum_{k=0}^n \|g_k\|^2}$$

**We need to estimate  $D$   
to remove the stepsize**

$$d_0 \implies x_1$$

# Distance-over-Gradients (DoG)

**Problem:**  $\min_{x \in \mathbb{R}^p} f(x)$

**Setting:**  $g_k \in \partial f(x_k), \quad \|g_k\| \leq G$

$$D = \|x_0 - x_*\|$$

Subgradient stepsize:  $\gamma_k = \frac{D}{G\sqrt{k}}, \quad D = \|x_0 - x_*\|$

Estimate using Adagrad:

$$G\sqrt{k} \approx \sqrt{\sum_{k=0}^n \|g_k\|^2}$$

**We need to estimate  $D$   
to remove the stepsize**

$$d_0 \implies x_1 \implies d_k = \|x_0 - x_k\| \approx \|x_0 - x_*\|$$

# DoG

$$d_k = \max_{t \leq k} \|x_0 - x_t\| \approx D$$

# DoG

$$d_k = \max_{t \leq k} \|x_0 - x_t\| \approx D$$

$$x_{k+1} = x_k - \frac{d_k}{\sqrt{\sum_{t=0}^k \|g_t\|^2}} g_k$$

**Distance over Gradients**

# DoG

$$d_k = \max_{t \leq k} \|x_0 - x_t\| \approx D$$

$$x_{k+1} = x_k - \frac{d_k}{\sqrt{\sum_{t=0}^k \|g_t\|^2}} g_k$$

**Theorem.** If  $\|g_k\| \leq G$ , then

$$\min_{k \leq t} [f(x_k) - f_*] = \mathcal{O} \left( \frac{DG \log \frac{D}{d_0}}{\sqrt{t}} \right)$$

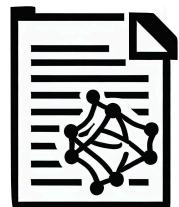
# DoG and DoWG



M. Ivgi, O. Hinder, Y. Carmon

**DoG is SGD's Best Friend: A Parameter-Free  
Dynamic Step Size Schedule**

*ICML 2023*



A. Khaled, K. Mishchenko, C. Jin

**DoWG Unleashed: An Efficient Universal Parameter-  
Free Gradient Descent Method**

*NeurIPS 2023*

# DoWG

$$d_k = \max_{t \leq k} \|x_0 - x_t\|$$

$$\text{DoG: } x_{k+1} = x_k - \frac{d_k}{\sqrt{\sum_{t=0}^k \|g_t\|^2}} g_k$$

# DoWG

$$d_k = \max_{t \leq k} \|x_0 - x_t\|$$

$$\text{DoG: } x_{k+1} = x_k - \frac{d_k}{\sqrt{\sum_{t=0}^k \|g_t\|^2}} g_k$$

$$\text{DoWG: } x_{k+1} = x_k - \frac{d_k^2}{\sqrt{\sum_{t=0}^k d_t^2 \|g_t\|^2}} g_k$$

Distance over **Weighted** Gradients



# DoWG

$$d_k = \max_{t \leq k} \|x_0 - x_t\|$$

$$\text{DoG: } x_{k+1} = x_k - \frac{d_k}{\sqrt{\sum_{t=0}^k \|g_t\|^2}} g_k$$

$$\text{DoWG: } x_{k+1} = x_k - \frac{d_k^2}{\sqrt{\sum_{t=0}^k d_t^2 \|g_t\|^2}} g_k$$

**Distance over Weighted Gradients**

# DoWG

$$d_k = \max_{t \leq k} \|x_0 - x_t\|$$

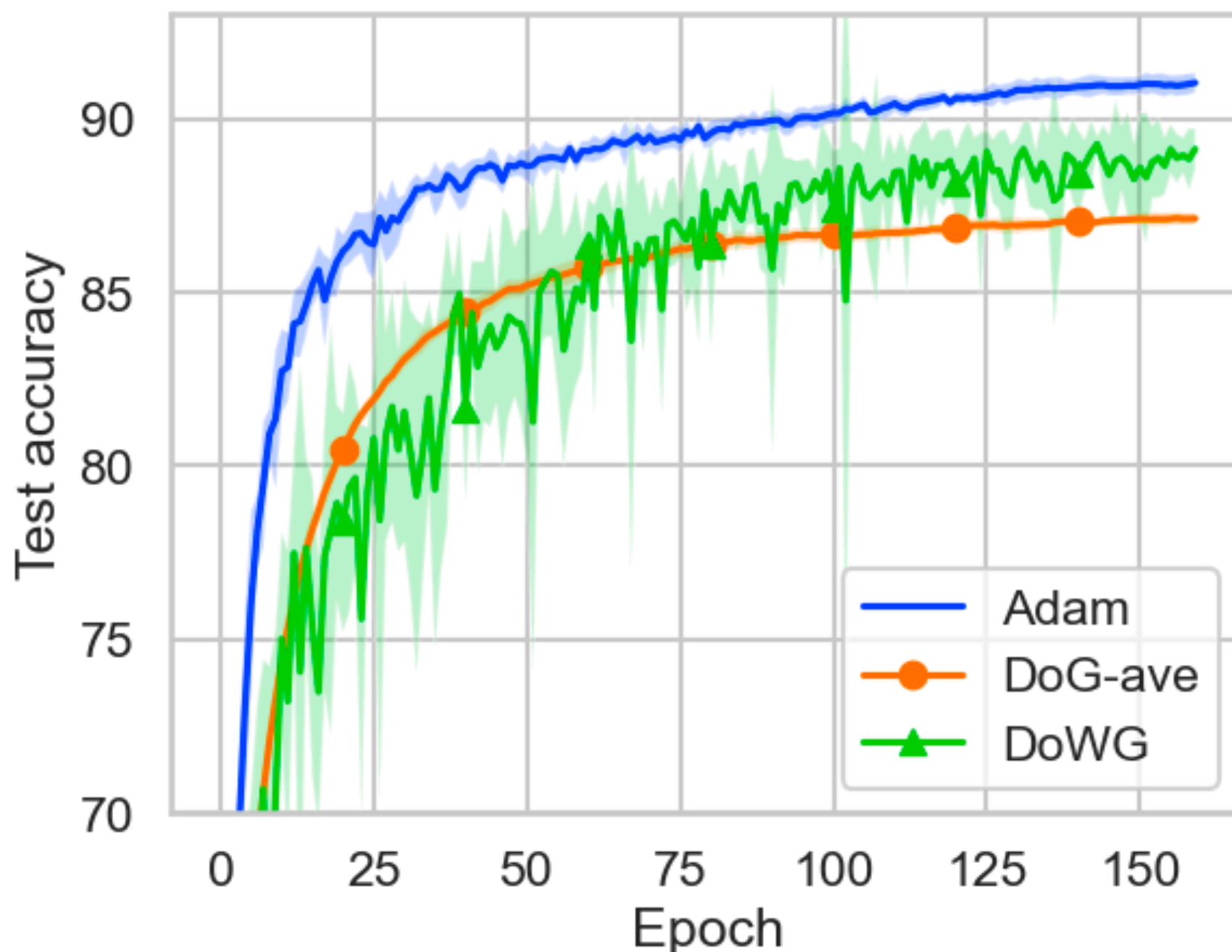
$$\text{DoG: } x_{k+1} = x_k - \frac{d_k}{\sqrt{\sum_{t=0}^k \|g_t\|^2}} g_k$$

$$\text{DoWG: } x_{k+1} = x_k - \frac{d_k^2}{\sqrt{\sum_{t=0}^k d_t^2 \|g_t\|^2}} g_k$$

**Theorem.** For nonsmooth  $f$  with  $\|g_k\| \leq G$

$$\min_{k \leq n} [f(x_k) - f_*] = \mathcal{O} \left( \frac{GD \sqrt{\log \frac{D}{d_0}}}{\sqrt{n}} \right).$$

# Empirical behaviour



**Test accuracy, VGG11 on CIFAR10. Unstable behaviour.  
Coordinate-wise estimation doesn't work.**

# Talk plan

1. The motivation
2. The prehistory
3. DoG and DoWG
4. D-Adaptation
5. Prodigy
6. Conclusion

# D-Adaptation

$$\hat{d}_{k+1} = \frac{\sum_{t=0}^k d_t \langle g_t, x_0 - x_t \rangle}{\left\| \sum_{t=0}^k d_t g_t \right\|}$$

**How much the gradient is correlated  
with our updates so far**

# D-Adaptation

$$\hat{d}_{k+1} = \frac{\sum_{t=0}^k d_t \langle g_t, x_0 - x_t \rangle}{\left\| \sum_{t=0}^k d_t g_t \right\|}$$

**Normalize correlations by gradient magnitudes**

# D-Adaptation

$$\hat{d}_{k+1} = \frac{\sum_{t=0}^k d_t \langle g_t, x_0 - x_t \rangle}{\left\| \sum_{t=0}^k d_t g_t \right\|}$$

$$d_{k+1} = \max(\hat{d}_{k+1}, d_k)$$

# D-Adaptation

$$\hat{d}_{k+1} = \frac{\sum_{t=0}^k d_t \langle g_t, x_0 - x_t \rangle}{\left\| \sum_{t=0}^k d_t g_t \right\|}$$

$$d_{k+1} = \max(\hat{d}_{k+1}, d_k)$$

**Theorem 1.**  $D \geq \hat{d}_{n+1}$     **(This estimate makes sense)**



# D-Adaptation

$$\hat{d}_{k+1} = \frac{\sum_{t=0}^k d_t \langle g_t, x_0 - x_t \rangle}{\left\| \sum_{t=0}^k d_t g_t \right\|}$$

$$d_{k+1} = \max(\hat{d}_{k+1}, d_k)$$

**Theorem 1.**  $D \geq \hat{d}_{n+1}$

**Theorem 2.** Asymptotically,  
$$f(\hat{x}_n) - f(x_*) = \mathcal{O}\left(\frac{DG}{\sqrt{n}}\right)$$

**(You pay nothing)**

# D-Adaptation

$$\hat{d}_{k+1} = \frac{\sum_{t=0}^k d_t \langle g_t, x_0 - x_t \rangle}{\left\| \sum_{t=0}^k d_t g_t \right\|}$$

$$d_{k+1} = \max(\hat{d}_{k+1}, d_k)$$

**Theorem 1.**  $D \geq \hat{d}_{n+1}$

**Theorem 2.** Asymptotically,  
$$f(\hat{x}_n) - f(x_*) = \mathcal{O}\left(\frac{DG}{\sqrt{n}}\right)$$

**Theorem 3.** Non-asymptotically,  
$$f(\hat{x}_n) - f(x_*) \leq \frac{8DG \log(D/d_0)}{\sqrt{n}}$$

**Same as DoG**

# Coordinate-wise version

$$s_{k+1} = s_k + d_k g_k$$

**Weighted gradient sum for Dual Averaging**

# Coordinate-wise version

$$s_{k+1} = s_k + d_k g_k$$

$$a_{k+1} = a_k + g_k^2$$

**Coordinate-wise  
estimate of Adagrad**

# Coordinate-wise version

$$s_{k+1} = s_k + d_k g_k$$

$$a_{k+1} = a_k + g_k^2$$

**Coordinate-wise  
estimate of Adagrad**

---

## Algorithm 1 Adam optimizer

---

- 1: **Input:**  $x_0, \beta_1 \in [0, 1)$  (default 0.9),  $\beta_2 \in [0, 1)$  (default 0.999),
  - 2:  $\gamma_k$  (default 0.001),  $\epsilon$  (default  $10^{-8}$ )
  - 3: **for**  $k = 1$  **to**  $n$  **do**
  - 4:      $g_k \in \partial f(x_k)$
  - 5:      $m_{k+1} = \beta_1 m_k + (1 - \beta_1) g_k$
  - 6:      $v_{k+1} = \beta_2 v_k + (1 - \beta_2) g_k^2$
  - 7:      $x_{k+1} = x_k - \gamma_k \frac{m_{k+1}}{\sqrt{v_{k+1}} + \epsilon}$
  - 8: **end for**
-

# Coordinate-wise version

$$s_{k+1} = s_k + d_k g_k$$

$$a_{k+1} = a_k + g_k^2$$

$$\hat{d}_{k+1} = \frac{\sum_{t=0}^k d_t \langle g_t, x_0 - x_t \rangle}{\left\| \sum_{t=0}^k d_t g_t \right\|_1}$$

**The main difference with non-coordinate version**

# Coordinate-wise version

$$s_{k+1} = s_k + d_k g_k$$

$$a_{k+1} = a_k + g_k^2$$

$$\hat{d}_{k+1} = \frac{\sum_{t=0}^k d_t \langle g_t, x_0 - x_t \rangle}{\left\| \sum_{t=0}^k d_t g_t \right\|_1}$$

$$x_{k+1} = x_0 - \frac{s_{k+1}}{\sqrt{a_{k+1}}}$$

# Coordinate-wise version

$$s_{k+1} = s_k + d_k g_k$$

$$a_{k+1} = a_k + g_k^2$$

$$\hat{d}_{k+1} = \frac{\sum_{t=0}^k d_t \langle g_t, x_0 - x_t \rangle}{\left\| \sum_{t=0}^k d_t g_t \right\|_1}$$

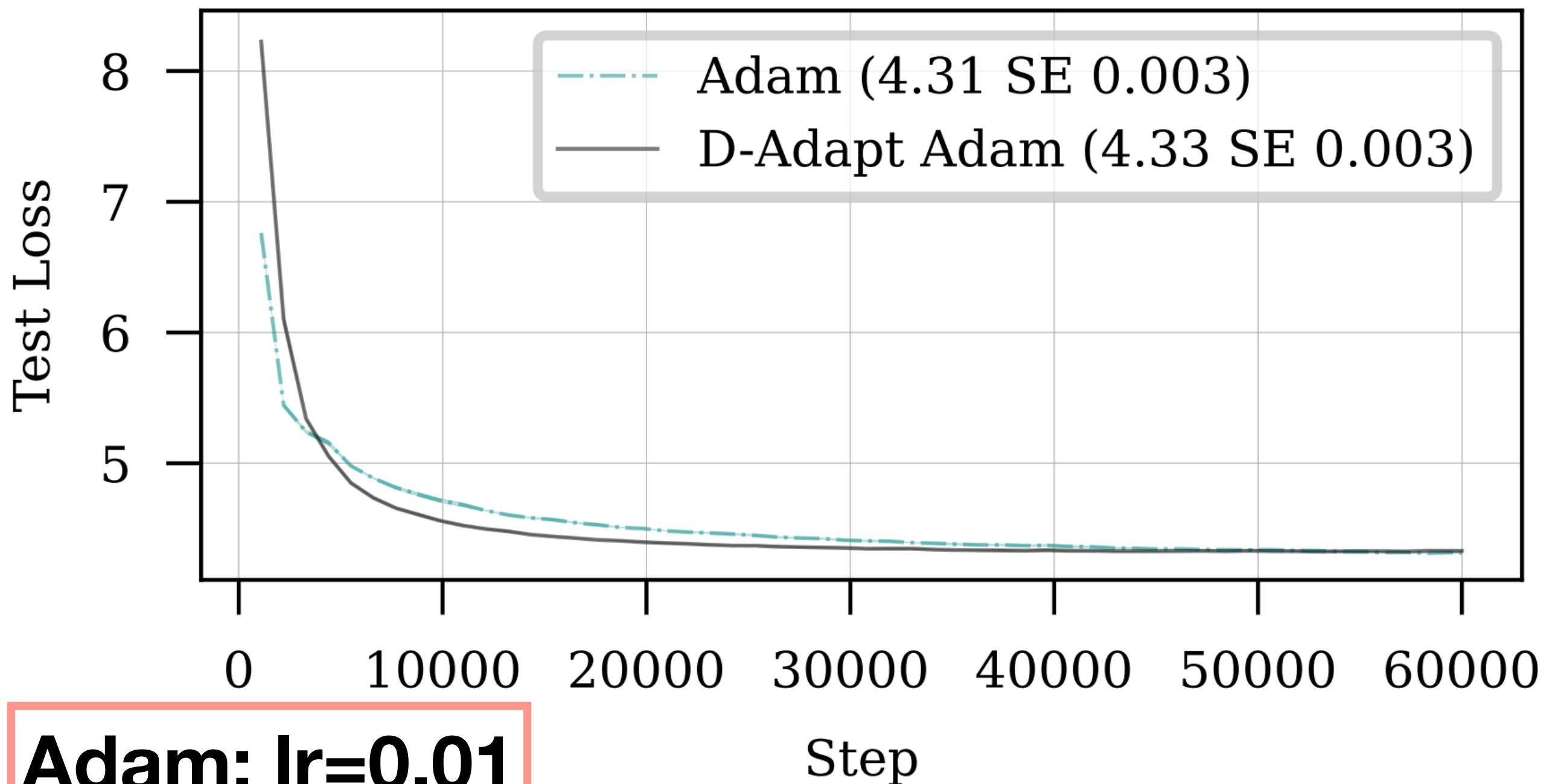
$$x_{k+1} = x_0 - \frac{s_{k+1}}{\sqrt{a_{k+1}}}$$

**Theorem.** Asymptotically,  $f(\hat{x}_n) - f_* = \mathcal{O}\left(\frac{pG_\infty D_\infty}{\sqrt{n}}\right)$ .



# Experiments

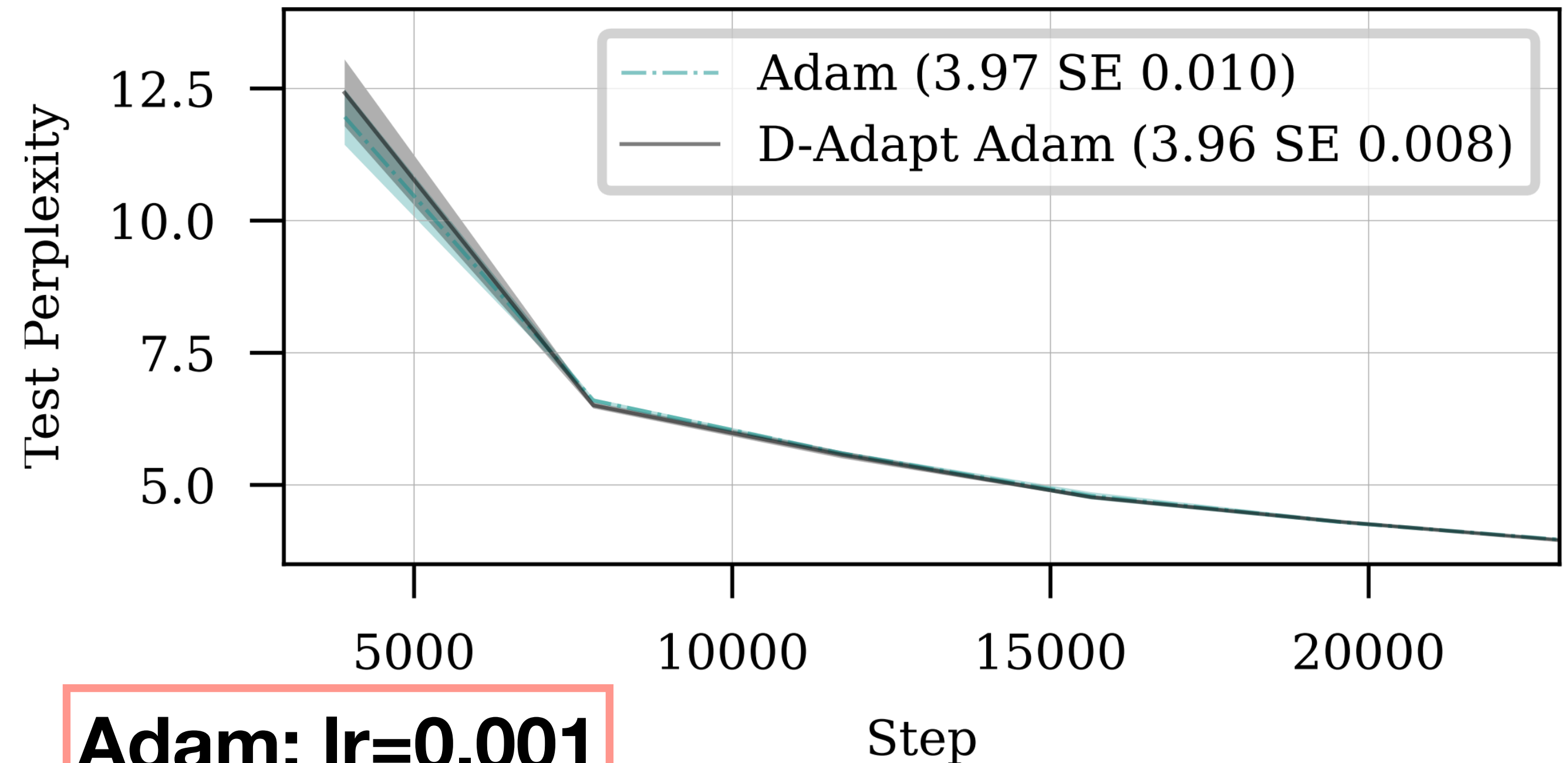
IWSLT14 (LSTM)



**Adam: lr=0.01**

# Experiments

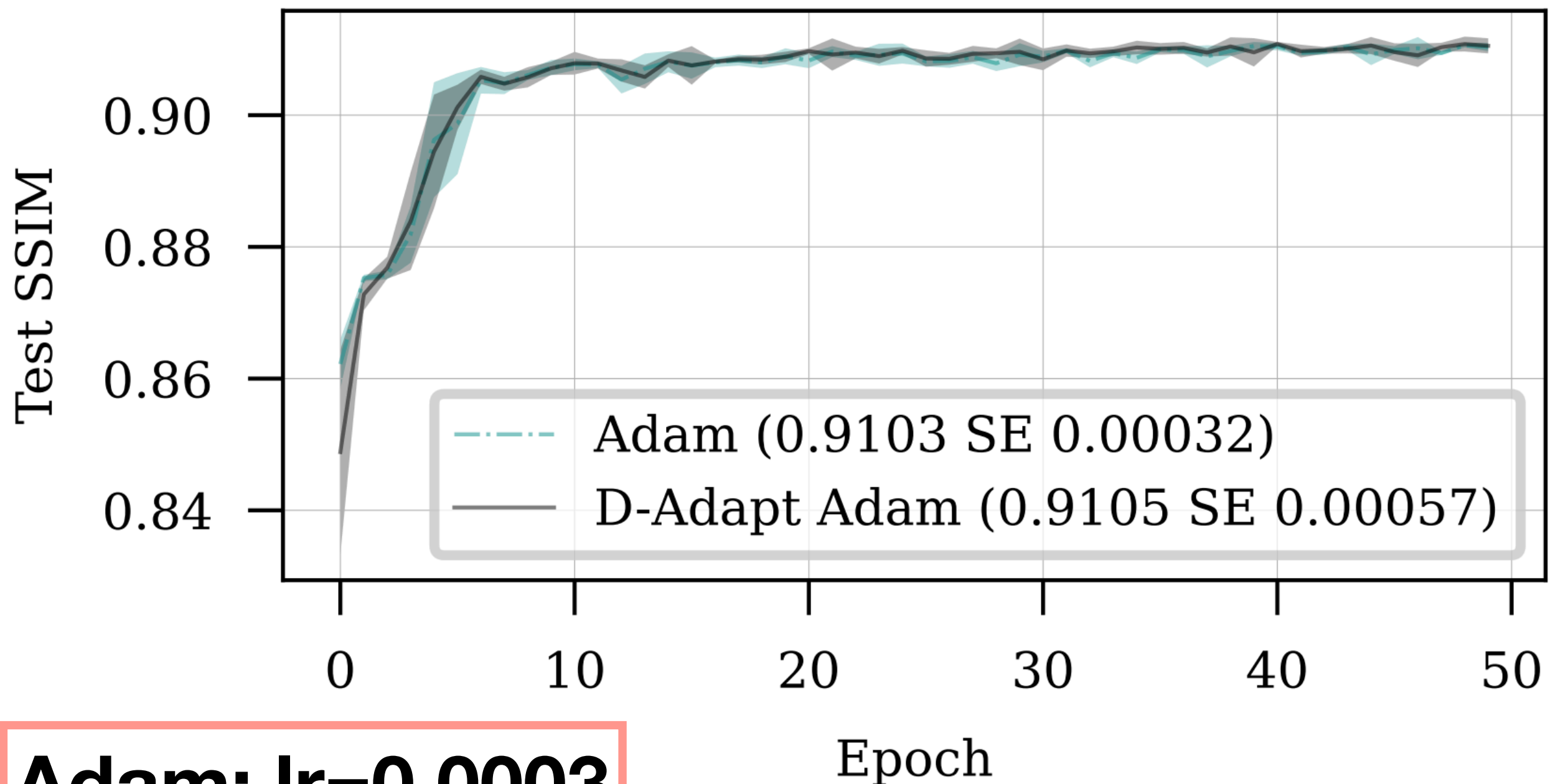
BookWiki (RoBERTa)



**Adam: lr=0.001**

# Experiments

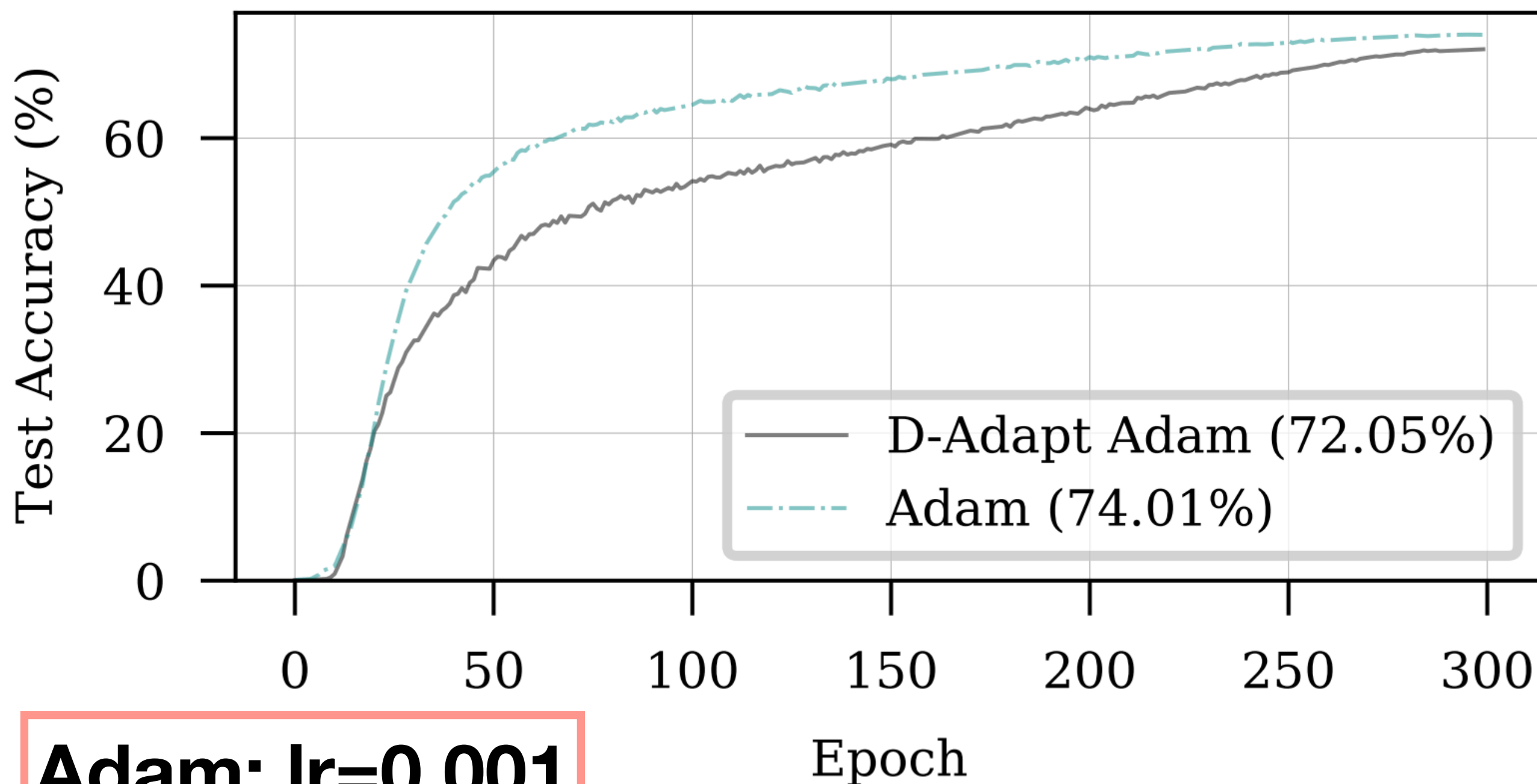
fastMRI Knee (VarNet 2.0)



**Adam: lr=0.0003**

# Experiments

ILSVRC 2012 ImageNet (Vision Transformer)



**Adam: lr=0.001**

# Talk plan

1. The motivation
2. The prehistory
3. DoG and DoWG
4. D-Adaptation
5. Prodigy
6. Conclusion

# Papers the talk is based on



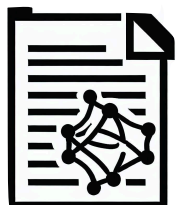
A. Defazio, K. Mishchenko  
**Learning-Rate-Free Learning by D-Adaptation**  
*ICML 2023 (released December 2022)*



M. Ivgi, O. Hinder, Y. Carmon  
**DoG is SGD's Best Friend: A Parameter-Free  
Dynamic Step Size Schedule**  
*ICML 2023 (released February 2023)*



A. Khaled, K. Mishchenko, C. Jin  
**DoWG Unleashed: An Efficient Universal Parameter-  
Free Gradient Descent Method**  
*arXiv:2305.16284, May 2023*



K. Mishchenko, A. Defazio  
**Prodigy: An Expediently Adaptive Parameter-Free Learner**  
*arXiv:2306.06101 June 2023*

# Prodigy

$$x_{k+1} = x_k - \eta_k g_k$$

$$\eta_k = \frac{d_k}{\sqrt{\sum_{t=0}^k d_t^2 \|g_t\|^2}} d_k$$

# Prodigy

$$x_{k+1} = x_k - \eta_k g_k$$

$$\eta_k = \frac{d_k}{\sqrt{\sum_{t=0}^k d_t^2 \|g_t\|^2}} d_k$$

**This is the same idea as DoWG**



# Prodigy

$$x_{k+1} = x_k - \eta_k g_k$$

$$\eta_k = \frac{d_k}{\sqrt{\sum_{t=0}^k d_t^2 \|g_t\|^2}} d_k$$

**But let's go one step further**

$$\eta_k = \frac{\lambda_k d_k}{\sqrt{\sum_{t=0}^k \lambda_t^2 d_t^2 \|g_t\|^2}} d_k$$

# Prodigy

$$x_{k+1} = x_k - \eta_k g_k$$

$$\eta_k = \frac{d_k}{\sqrt{\sum_{t=0}^k d_t^2 \|g_t\|^2}} d_k$$

**But let's go one step further**

$$\eta_k = \frac{\lambda_k d_k}{\sqrt{\sum_{t=0}^k \lambda_t^2 d_t^2 \|g_t\|^2}} d_k$$

$$\lambda_t^2 = \beta_2^{-t} \implies \text{same weights as in Adam}$$

# Prodigy

**Theorem.** If  $\|g_k\| \leq G$ , then

$$\min_{k \leq t} [f(x_k) - f_*] = \mathcal{O} \left( \frac{DG \sqrt{\log \frac{D}{d_0}}}{\sqrt{t}} \right)$$

**Prodigy to D-Adaptation is same  
as DoWG to DoG**

# Prodigy

---

**Algorithm 4** Prodigy (Adam version)

---

**for**  $k = 0$  **to**  $n$  **do**

$$g_k \in \partial f(x_k)$$

$$m_{k+1} = \beta_1 m_k + (1 - \beta_1) d_k g_k$$

$$v_{k+1} = \beta_2 v_k + (1 - \beta_2) d_k^2 g_k^2$$

$$x_{k+1} = x_k - \gamma_k d_k m_{k+1} / (\sqrt{v_{k+1}} + \epsilon)$$

$$r_{k+1} = \sqrt{\beta_2} r_k + (1 - \sqrt{\beta_2}) \langle g_k, x_0 - x_k \rangle$$

$$s_{k+1} = \sqrt{\beta_2} s_k + (1 - \sqrt{\beta_2}) d_k g_k$$

$$\hat{d}_{k+1} = \frac{r_{k+1}}{\|s_{k+1}\|_1}$$

$$d_{k+1} = \max(d_k, \hat{d}_{k+1})$$

**end for**

---

# Prodigy

---

**Algorithm 4** Prodigy (Adam version)

---

**for**  $k = 0$  **to**  $n$  **do**

$$g_k \in \partial f(x_k)$$

$$m_{k+1} = \beta_1 m_k + (1 - \beta_1) d_k g_k$$

$$v_{k+1} = \beta_2 v_k + (1 - \beta_2) d_k^2 g_k^2$$

$$x_{k+1} = x_k - \gamma_k d_k m_{k+1} / (\sqrt{v_{k+1}} + \epsilon)$$

$$r_{k+1} = \sqrt{\beta_2} r_k + (1 - \sqrt{\beta_2}) \langle g_k, x_0 - x_k \rangle$$

$$s_{k+1} = \sqrt{\beta_2} s_k + (1 - \sqrt{\beta_2}) d_k g_k$$

$$\hat{d}_{k+1} = \frac{r_{k+1}}{\|s_{k+1}\|_1}$$

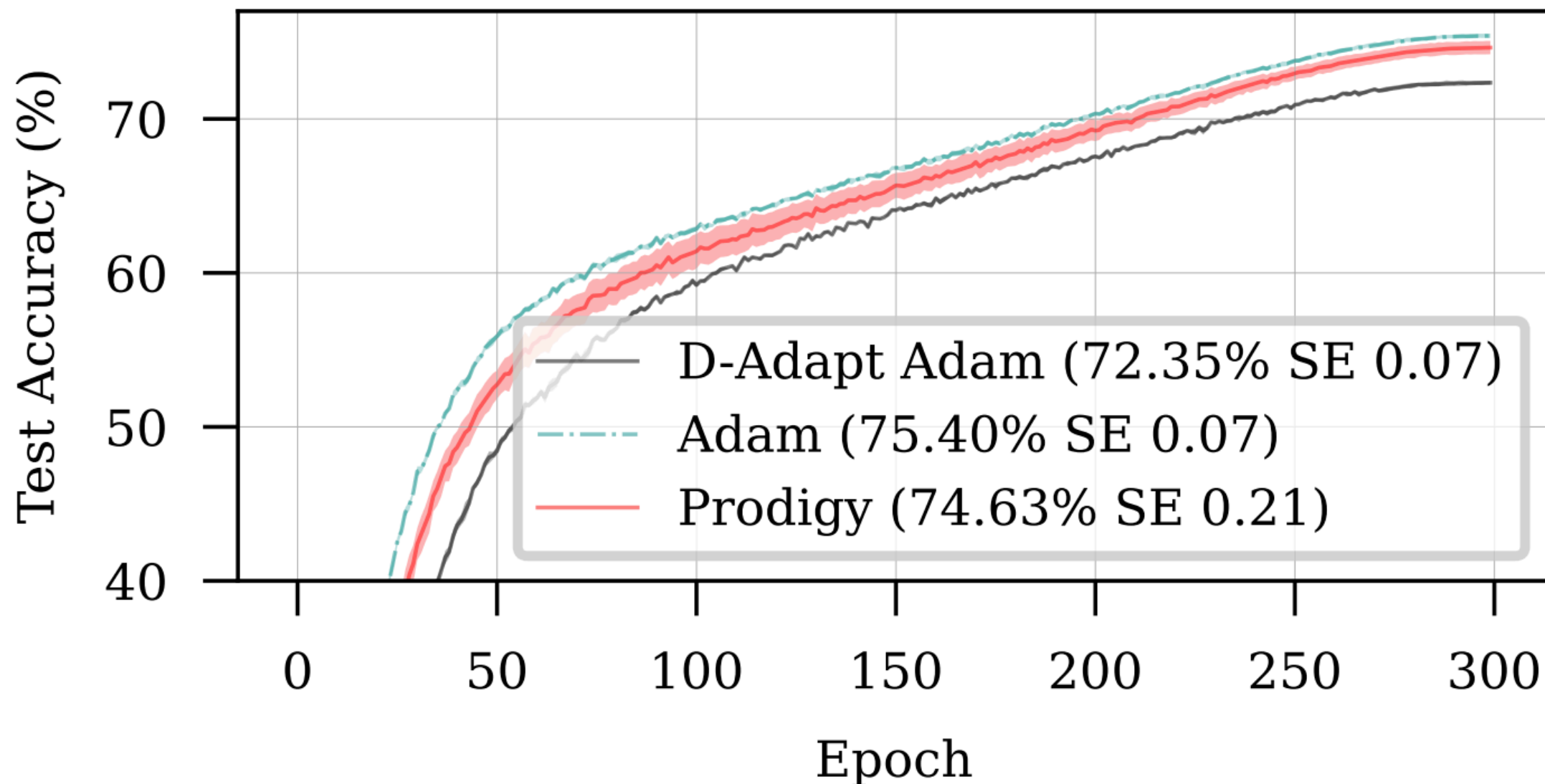
$$d_{k+1} = \max(d_k, \hat{d}_{k+1})$$

**end for**

---

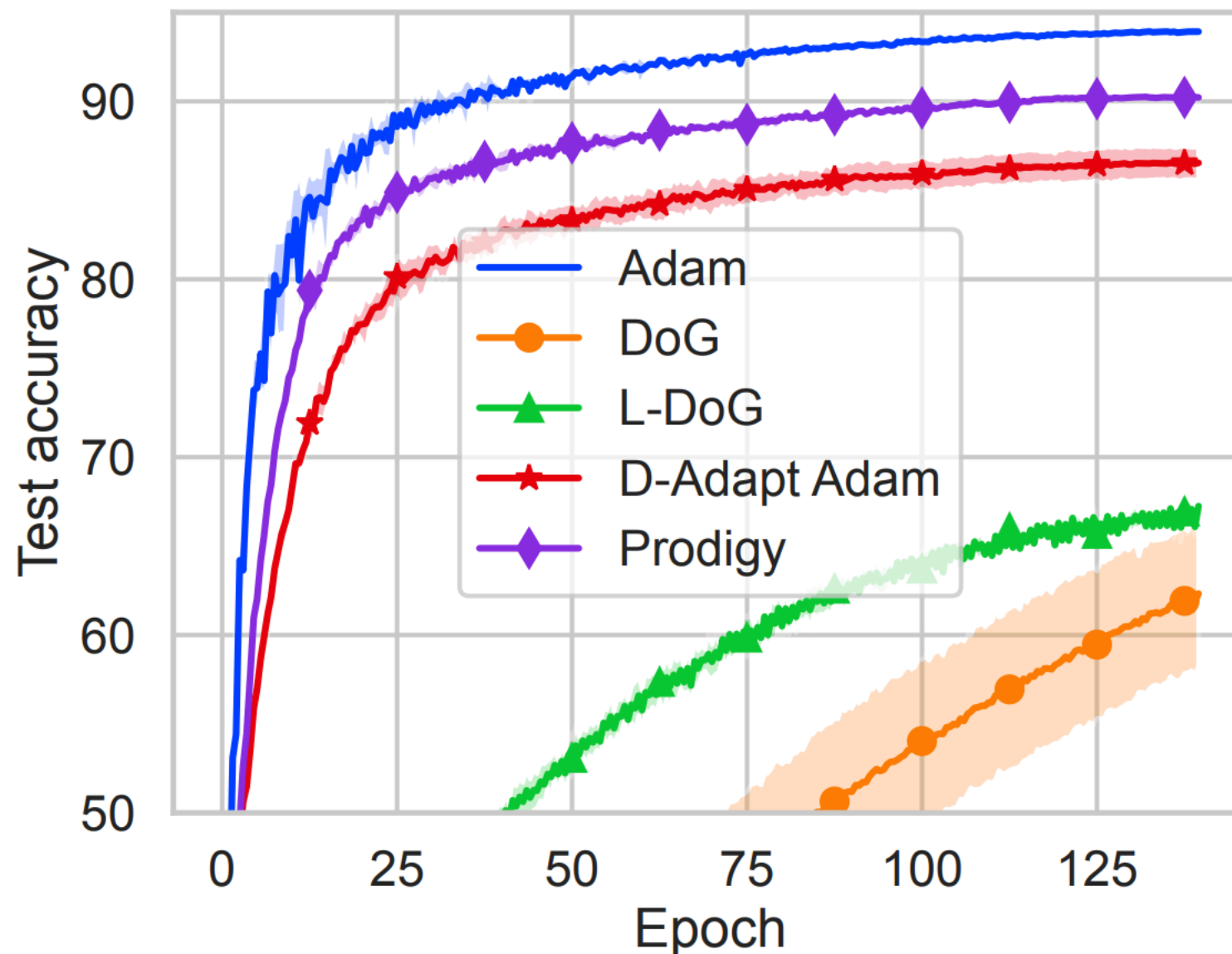
# Numerical results

ILSVRC 2012 ImageNet (Vision Transformer)



**The gap when training ViT on Imagenet is almost closed**

# Numerical results



**ResNet-50 on CIFAR10 is still a bit suboptimal**

# Talk plan

1. The motivation
2. The prehistory
3. DoG and DoWG
4. D-Adaptation
5. Prodigy
6. Conclusion



# Conclusion

- 1. There is still a small performance gap**

# Conclusion

- 1. There is still a small performance gap**
- 2. Stochastic analysis is very hard**

# Conclusion

- 1. There is still a small performance gap**
- 2. Stochastic analysis is very hard**
- 3. No coordinate-wise theory beyond DA**

# Conclusion

- 1. There is still a small performance gap**
- 2. Stochastic analysis is very hard**
- 3. No coordinate-wise theory beyond DA**
- 4.  $d_k$ : can we decrease it?**

# Conclusion

1. There is still a small performance gap
2. Stochastic analysis is very hard
3. No coordinate-wise theory beyond DA
4.  $d_k$ : can we decrease it?
5.  $d_k$ : can we use it coordinate-wise?

# Conclusion

1. There is still a small performance gap
2. Stochastic analysis is very hard
3. No coordinate-wise theory beyond DA
4.  $d_k$ : can we decrease it?
5.  $d_k$ : can we use it coordinate-wise?
6. Why is Adam helpful?