SGD or Adam?

Presenter: Wei Li

Advisor: I-Chen Wu

Outline

- Recap: gradient descent optimization algorithms
  - Gradient descent variants
  - Gradient Descent Optimization Algorithms
- SGD vs. Adam
  - Pros and Cons of Adam
  - SWATS: Adam+SGD
- Choice of optimizer

Reference

- Keskar, Nitish Shirish, and Richard Socher. "**Improving Generalization Performance by Switching from Adam to SGD.**" arXiv preprint arXiv:1712.07628 (2017).
- Reddi, Sashank J., Satyen Kale, and Sanjiv Kumar. "**On the convergence of adam and beyond.**" International Conference on Learning Representations. 2018.
- Wilson, Ashia C., et al. "**The marginal value of adaptive gradient methods in machine learning**." Advances in Neural Information Processing Systems. 2017.
- Ruder, Sebastian. "**An overview of gradient descent optimization algorithms**." *arXiv preprint arXiv:1609.04747*(2016).

Recap: Gradient descent optimization

Gradient descent variants

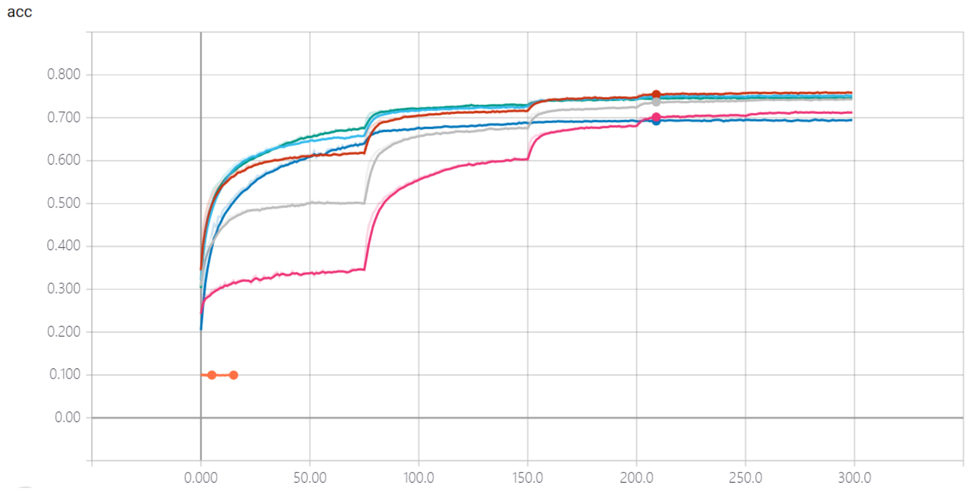
Gradient Descent Optimization Algorithms

Gradient descent variants

- Batch Gradient Descent(BGD)
- Stochastic Gradient Descent(SGD)
- Mini-Batch Gradient Descent(MBGD)
  - Which is called **SGD** in **Deep learning**

- About batch size:
  - Batch size = 1 (SGD)
    - **Maybe** reach best testing accuracy
    - Sometimes can't converge
    - Training speed: slow
  - Batch size = ∞ (BGD)
    - Can get best training accuracy, bad testing accuracy
    - Training speed: fast
  - Batch size = **a reasonable number** ?
    - can get good performance
    - Training speed: depend on batch size

About batch size (LeNet on CIFAR-10)



Name	Smoothed	Value	Step	Time	Relative
lenet_a_1	0.09986	0.09986	5.000	Fri Mar 9, 02:54:31	11m 8s
lenet_b_10	0.09972	0.09930	15.00	Fri Mar 9, 12:48:45	3m 48s
lenet_c_32	0.7127	0.7136	299.0	Fri Mar 9, 08:01:40	1h 3m 11s
lenet_d_64	0.7436	0.7449	299.0	Fri Mar 9, 09:55:10	58m 13s
lenet_e_128	$\boxed{0.7588}$	0.7593	299.0	Fri Mar 9, 06:02:26	55m 40s
lenet_f_256	$\boxed{0.7519}$	0.7533	299.0	Fri Mar 9, 06:58:13	55m 32s
lenet_g_512	0.7467	0.7470	299.0	Fri Mar 9, 08:56:40	54m 46s
lenet_h_1024	0.6945	0.6944	299.0	Fri Mar 9, 03:50:06	53m 32s

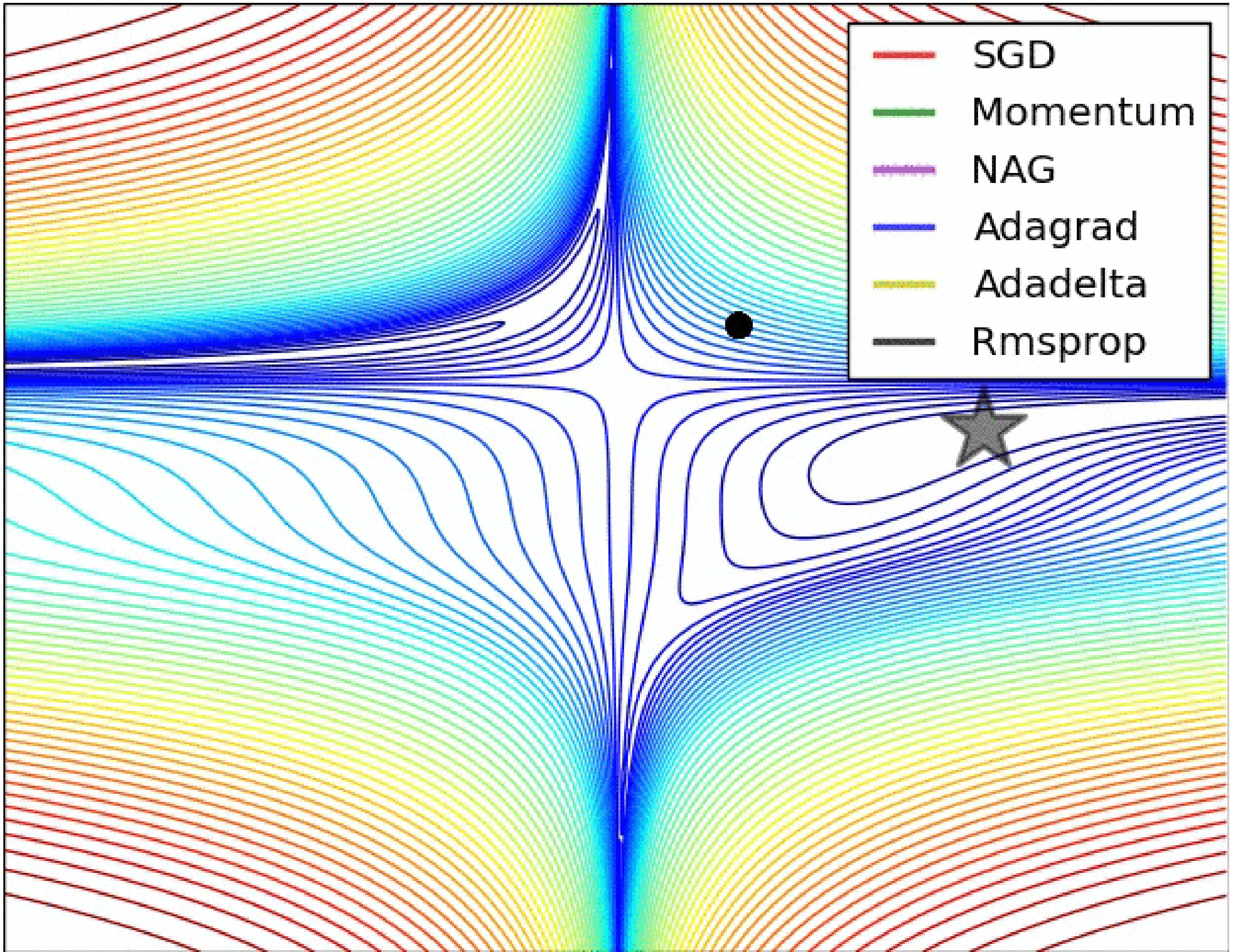
About batch size (LeNet on MNIST)

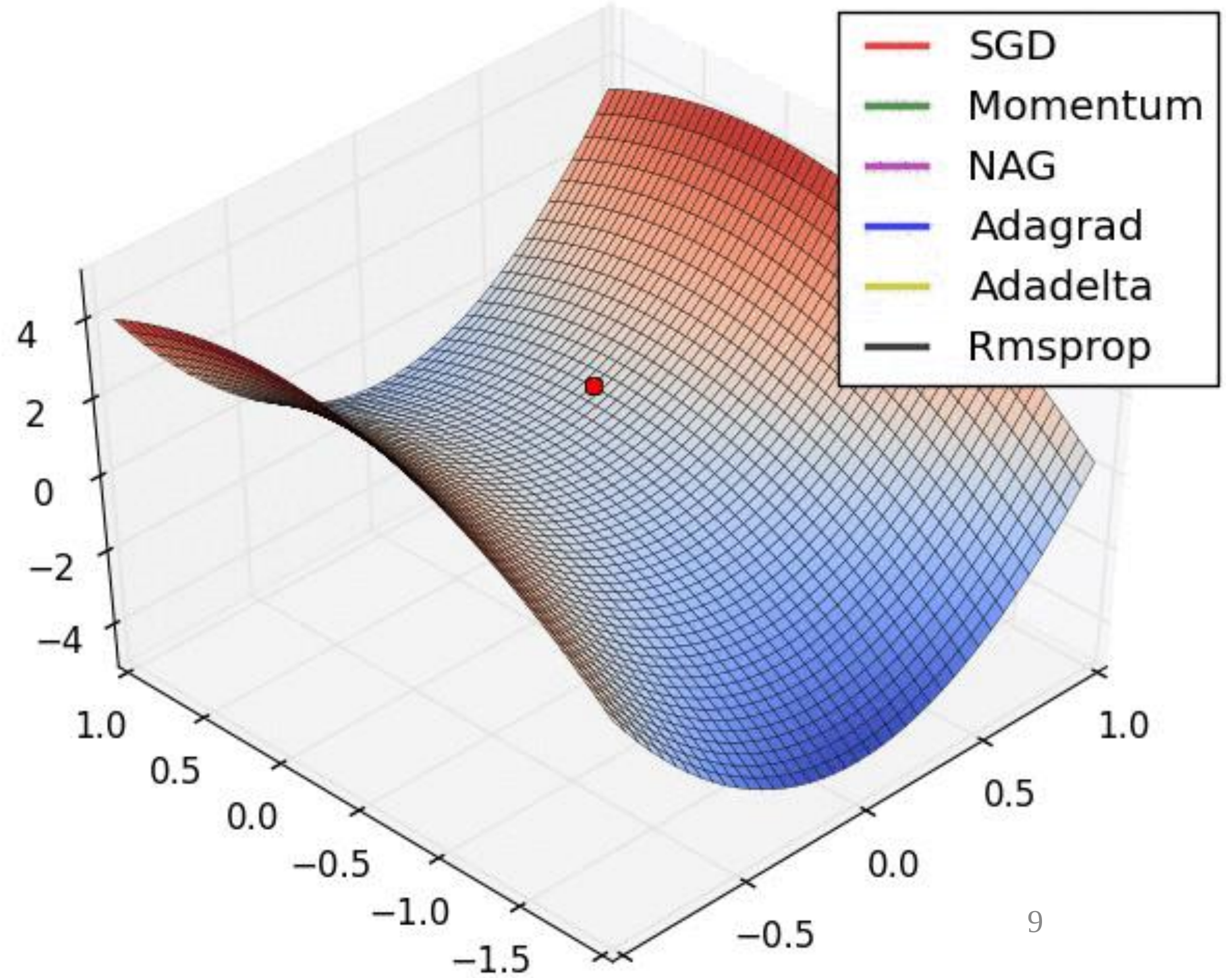
Batch_Size	5000	2000	1000	500	256	100	50	20	10	5	2	1
Total Epoches	200	200	200	200	200	200	200	200	200	200	200	200
Total Iterations	1999	4999	9999	19999	38999	99999	199999	499999	999999	1999999	cannot converge	
Time of 200 Epoches	1	1.068	1.16	1.38	1.75	3.016	5.027	8.513	13.773	24.055		
Achieve 0.99 Accuracy at Epoch	-	-	135	78	41	45	24	9	9	-		
Time of Achieve 0.99 Accuracy	-	-	2.12	1.48	1	1.874	1.7	1.082	1.729	-		
Best Validation Score	0.015	0.011	0.01	0.01	0.01	0.009	0.0098	0.0084	0.01	0.032		
Best Score Achieved at Epoch	182	170	198	100	93	111	38	49	51	17		
Best Test Score	0.014	0.01	0.01	0.01	0.01	0.008	0.0083	0.0088	0.008	0.0262		
Final Test Error (200 epoches)	0.0134	0.01	0.01	0.01	0.01	0.009	0.0082	0.0088	0.008	0.0662		

Gradient Descent Optimization Algorithms

- SGD
- SGD with Momentum(**SGDM**, default use in many projects)
- SGD with Nesterov Accelerated Gradient (**NAG**)
- **Adaptive optimization algorithms**:
  - AdaGrad
  - AdaDelta / RMSProp
  - Adam
  - Nadam
  - etc.

Gradient Descent Optimization Algorithms



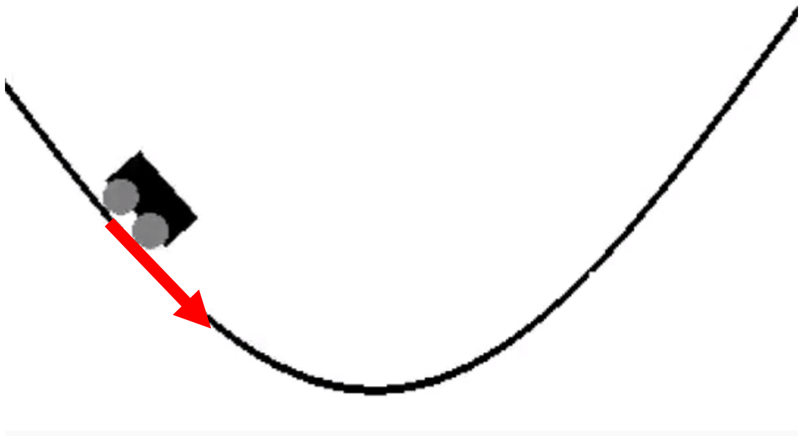


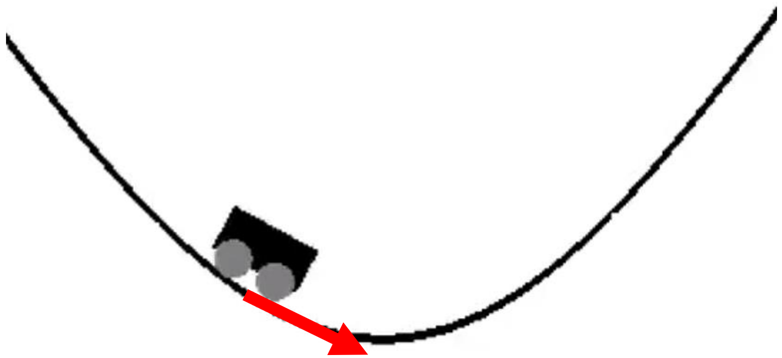
SGD

- **SGD**

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \boldsymbol{\alpha} \cdot \boldsymbol{G}(\boldsymbol{J}(\boldsymbol{\theta}))$$

θ: weights
α: learnig rate β: momentum term
$G(\cdot)$: gradient $J(\cdot)$: loss function





SGD(cont.)

- SGD

$v = \mathbf{0} \cdot \boldsymbol{v} + \alpha \cdot G(J(\theta))$

$\theta = \theta - v$

- **SGDM**

$\boldsymbol{v} = \boldsymbol{\beta v} + \boldsymbol{\alpha} \cdot \boldsymbol{G}(\boldsymbol{J}(\boldsymbol{\theta}))$

$\boldsymbol{\theta} = \boldsymbol{\theta} - \boldsymbol{v}$

θ: weights
α: learnig rate β: momentum term
$G(\cdot)$: gradient $J(\cdot)$: loss function

SGD(cont.)

- SGD

$v = \mathbf{0} \cdot \boldsymbol{v} + \alpha \cdot G(J(\theta))$

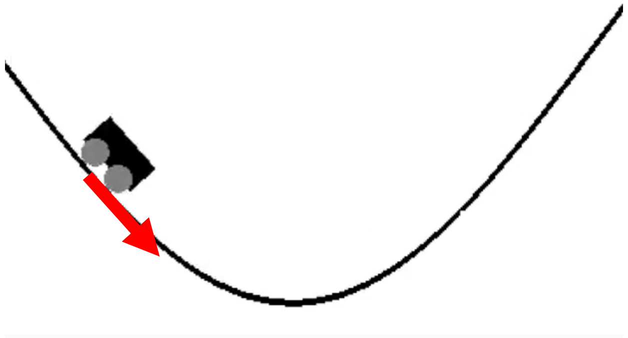
$\theta = \theta - v$

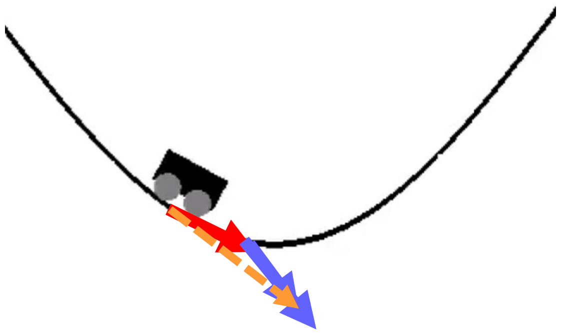
- **SGDM**

$\boldsymbol{v} = \boldsymbol{\beta v} + \boldsymbol{\alpha} \cdot \boldsymbol{G}(\boldsymbol{J}(\boldsymbol{\theta}))$

$\boldsymbol{\theta} = \boldsymbol{\theta} - \boldsymbol{v}$

θ: weights
α: learnig rate β: momentum term
$G(\cdot)$: gradient $J(\cdot)$: loss function





SGD(cont.)

θ: weights
α: learnig rate β: momentum term
$G(\cdot)$: gradient $J(\cdot)$: loss function

- SGD

$$v = \mathbf{0} \cdot \boldsymbol{v} + \alpha \cdot G(J(\theta))$$

$$\theta = \theta - v$$

- SGDM

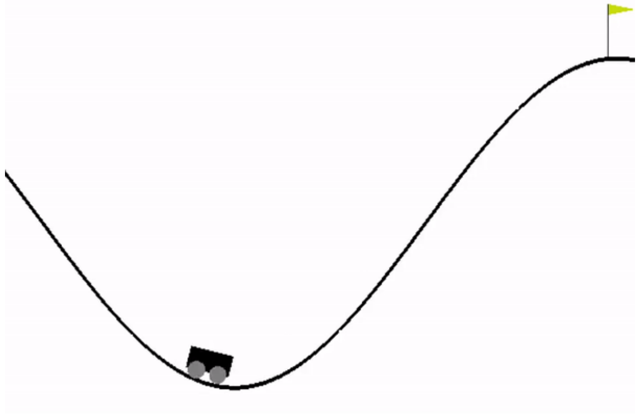
$$v = \boldsymbol{\beta v} + \alpha \cdot G(J(\theta))$$

$$\theta = \theta - v$$

- **NAG**

$$\boldsymbol{v} = \boldsymbol{\beta v} + \boldsymbol{\alpha} \cdot \boldsymbol{G}(\boldsymbol{J}(\boldsymbol{\theta} - \boldsymbol{\beta v}))$$

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \boldsymbol{v}$$



SGD(cont.)

- SGD

$v = \mathbf{0} \cdot \boldsymbol{v} + \alpha \cdot G(J(\theta))$

$\theta = \theta - v$

- SGDM

$v = \boldsymbol{\beta v} + \alpha \cdot G(J(\theta))$

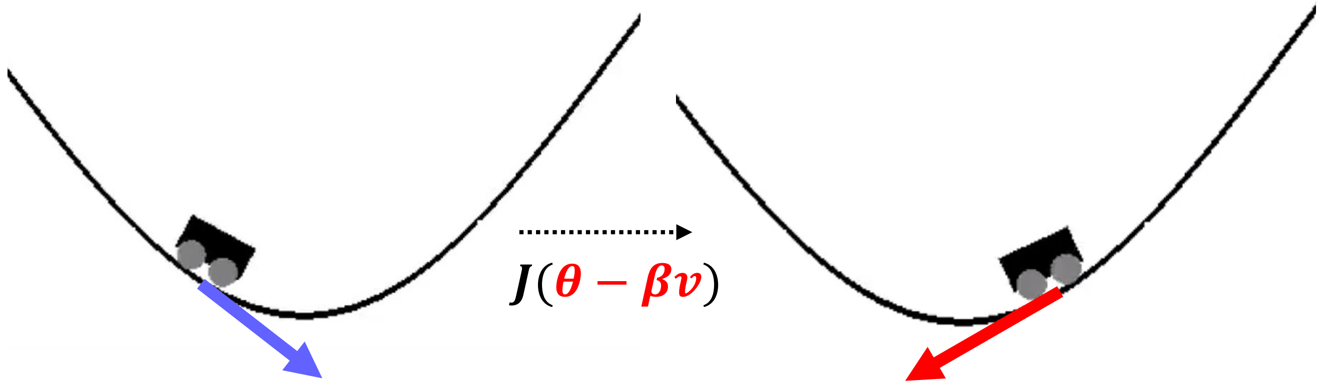
$\theta = \theta - v$

- **NAG**

$\boldsymbol{v} = \boldsymbol{\beta v} + \boldsymbol{\alpha} \cdot \boldsymbol{G}(\boldsymbol{J}(\boldsymbol{\theta} - \boldsymbol{\beta v}))$

$\boldsymbol{\theta} = \boldsymbol{\theta} - \boldsymbol{v}$

θ: weights
α: learnig rate β: momentum term
$G(\cdot)$: gradient $J(\cdot)$: loss function



SGD(cont.)

- SGD

$$v = \mathbf{0} \cdot \boldsymbol{v} + \alpha \cdot G(J(\theta))$$

$$\theta = \theta - v$$

- SGDM

$$v = \boldsymbol{\beta v} + \alpha \cdot G(J(\theta))$$

$$\theta = \theta - v$$

- **NAG**

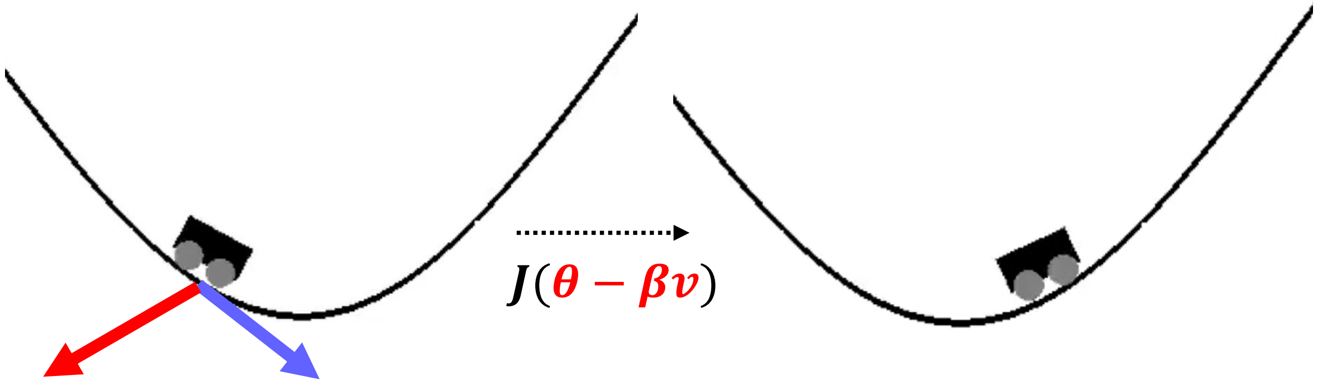
$$\boldsymbol{v} = \boldsymbol{\beta v} + \boldsymbol{\alpha} \cdot \boldsymbol{G}(\boldsymbol{J}(\boldsymbol{\theta} - \boldsymbol{\beta v}))$$

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \boldsymbol{v}$$

θ: weights

α: learnig rate β: momentum term

$G(\cdot)$: gradient $J(\cdot)$: loss function



SGD(cont.)

- SGD

$v = \mathbf{0} \cdot \boldsymbol{v} + \alpha \cdot G(J(\theta))$

$\theta = \theta - v$

- SGDM

$v = \boldsymbol{\beta v} + \alpha \cdot G(J(\theta))$

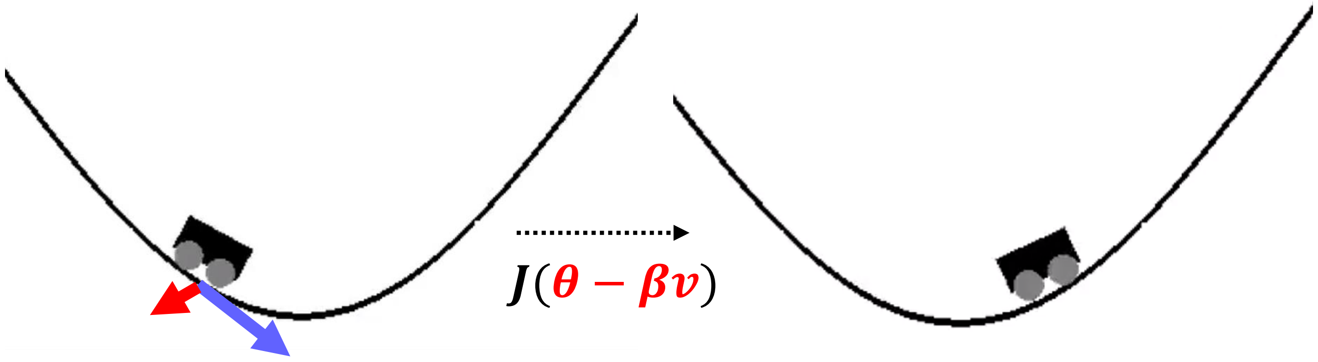
$\theta = \theta - v$

- **NAG**

$\boldsymbol{v} = \boldsymbol{\beta v} + \boldsymbol{\alpha} \cdot \boldsymbol{G}(\boldsymbol{J}(\boldsymbol{\theta} - \boldsymbol{\beta v}))$

$\boldsymbol{\theta} = \boldsymbol{\theta} - \boldsymbol{v}$

θ: weights
α: learnig rate β: momentum term
$G(\cdot)$: gradient $J(\cdot)$: loss function



SGD(cont.)

θ: weights
α: learnig rate β: momentum term
$G(\cdot)$: gradient $J(\cdot)$: loss function

- SGD

$$v = \mathbf{0} \cdot \boldsymbol{v} + \alpha \cdot G(J(\theta))$$

$$\theta = \theta - v$$

- SGDM

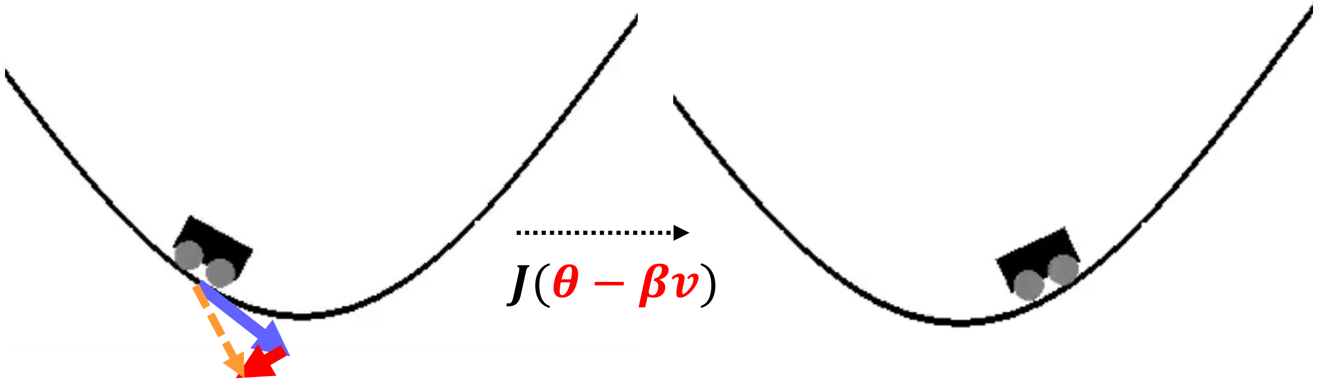
$$v = \boldsymbol{\beta v} + \alpha \cdot G(J(\theta))$$

$$\theta = \theta - v$$

- **NAG**

$$\boldsymbol{v} = \boldsymbol{\beta v} + \boldsymbol{\alpha} \cdot \boldsymbol{G}(\boldsymbol{J}(\boldsymbol{\theta} - \boldsymbol{\beta v}))$$

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \boldsymbol{v}$$



Adaptive optimization

- **Adagrad**

$$\boldsymbol{s} = \boxed{\frac{\boldsymbol{\alpha}}{\sqrt{\sum_{k=1}^{t}\left(\boldsymbol{g}_{k,i}^{2}\right) + \boldsymbol{\epsilon}}}} \cdot \boldsymbol{g}_{t,i}$$

$$\theta = \theta - s$$

θ: weights
α: learnig rate
β: momentum term
$G(\cdot)$: gradient
$J(\cdot)$: loss function
$g_{t,i} = G\left(J(\theta_{t,i})\right)$

Adaptive optimization(cont.)

- Adagrad

$$s = \frac{\alpha}{\sqrt{\sum_{k=1}^{t}\left(g_{k,i}^{2}\right) + \epsilon}} \cdot g_{t,i}$$

$$\theta = \theta - s$$

- **Adadelta**

$$s = \frac{\alpha}{\sqrt{\mathrm{E}[g^{2}]_{t,i} + \epsilon}} \cdot g_{t,i}$$

$$\theta = \theta - s$$

θ: weights
α: learnig rate
β: momentum term
$G(\cdot)$: gradient
$J(\cdot)$: loss function
$g_{t,i} = G\left(J(\theta_{t,i})\right)$

Adaptive optimization(cont.)

- **Adam**
  - Adam use estimates of first and second moments of the gradients.

$$\boldsymbol{m} = \boldsymbol{\beta_1 v} + (\mathbf{1} - \boldsymbol{\beta_1})\boldsymbol{G}^{\mathbf{1}}(\boldsymbol{J}(\boldsymbol{\theta}))$$

$$\boldsymbol{v} = \boldsymbol{\beta_2 v} + (\mathbf{1} - \boldsymbol{\beta_2})\boldsymbol{G}^{\mathbf{2}}(\boldsymbol{J}(\boldsymbol{\theta}))$$

$$\boldsymbol{s} = \frac{\boldsymbol{\alpha}}{\sqrt{\boldsymbol{v}} + \boldsymbol{\epsilon}} \cdot \boldsymbol{m}$$

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \boldsymbol{s}$$

θ: weights
α: learnig rate
$\beta_{1,2}$: momentum term
$G(\cdot)$: gradient
$J(\cdot)$: loss function

SGD vs. Adam

Pros and Cons of Adam

SWATS: Adam+SGD

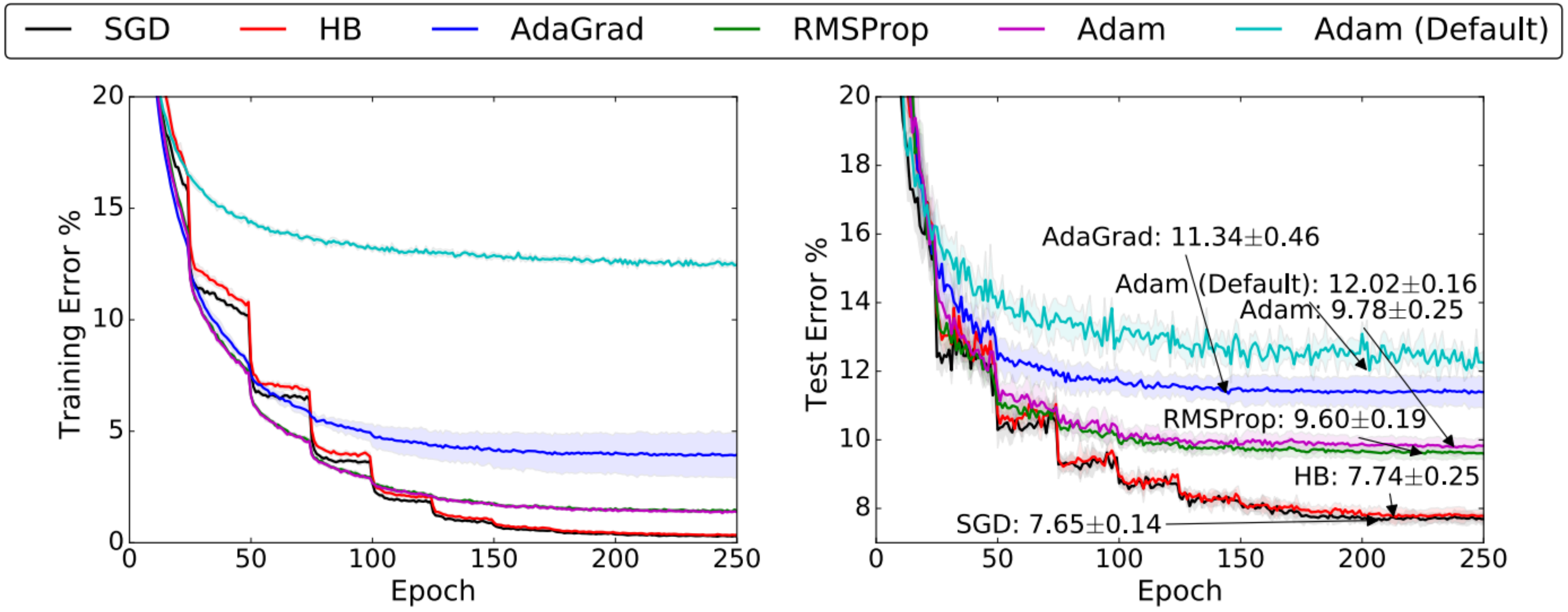
Pros and Cons of Adam

- Advantages:
  - Fast convergence
  - Good Performance than other adaptive optimization algorithms
  - Needn't to change learning rate by yourself
- Disadvantages:
  - More memory usage
  - Generalize worse (often significantly worse) than SGD, even when these solutions have better training performance
    - **The Marginal Value of Adaptive Gradient Methods in Machine Learning**

Pros and Cons of Adam

- Advantages:
  - Fast convergence
  - Good Performance than other adaptive optimization algorithms
  - Needn't to change learning rate by yourself
- Disadvantages:
  - More memory usage
  - Generalize worse (often significantly worse) than SGD, even when these solutions have better training performance
    - **The Marginal Value of Adaptive Gradient Methods in Machine Learning**
- **Another Problem**:
  - May not converge to the optimal solution
    - ICLR2018: **On the Convergence of Adam and Beyond**
    - **Many** proofs in appendix

VGG+BN+Dropout network for CIFAR-10



(a) CIFAR-10 (Train) **(b)** CIFAR-10 (Test)

Figure 1: Training (left) and top-1 test error (right) on CIFAR-10. The annotations indicate where the best performance is attained for each method. The shading represents $\pm$ one standard deviation computed across five runs from random initial starting points. In all cases, adaptive methods are performing worse on both train and test than non-adaptive methods.

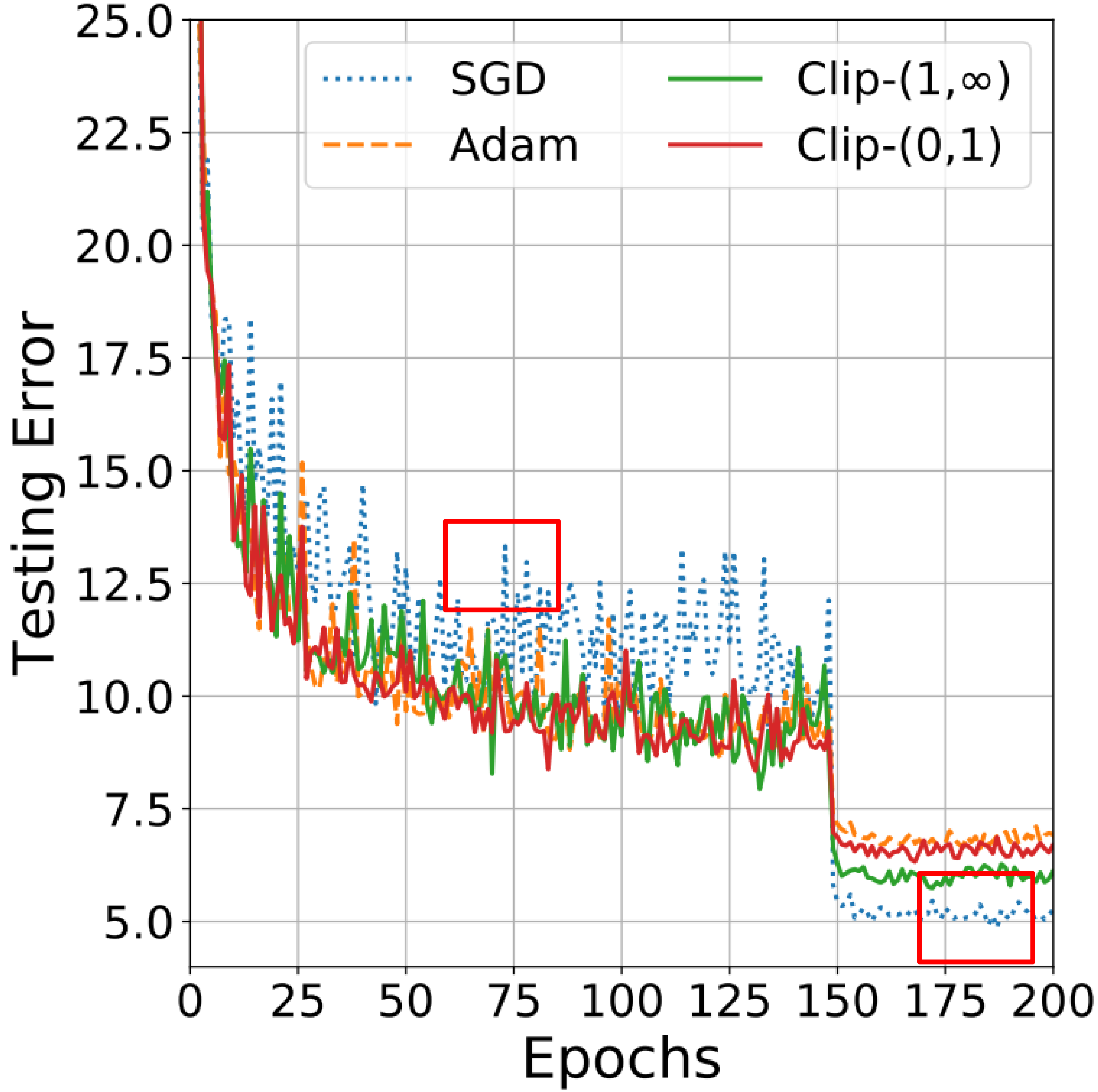
Learning Rate Clip

$$w_k = w_{k-1} - \alpha_{k-1} \cdot \frac{\sqrt{1-\beta_2^k}}{1-\beta_1^k} \cdot \frac{m_{k-1}}{\sqrt{v_{k-1}} + \epsilon},$$

$\mathrm{clip}(x, a, b)$

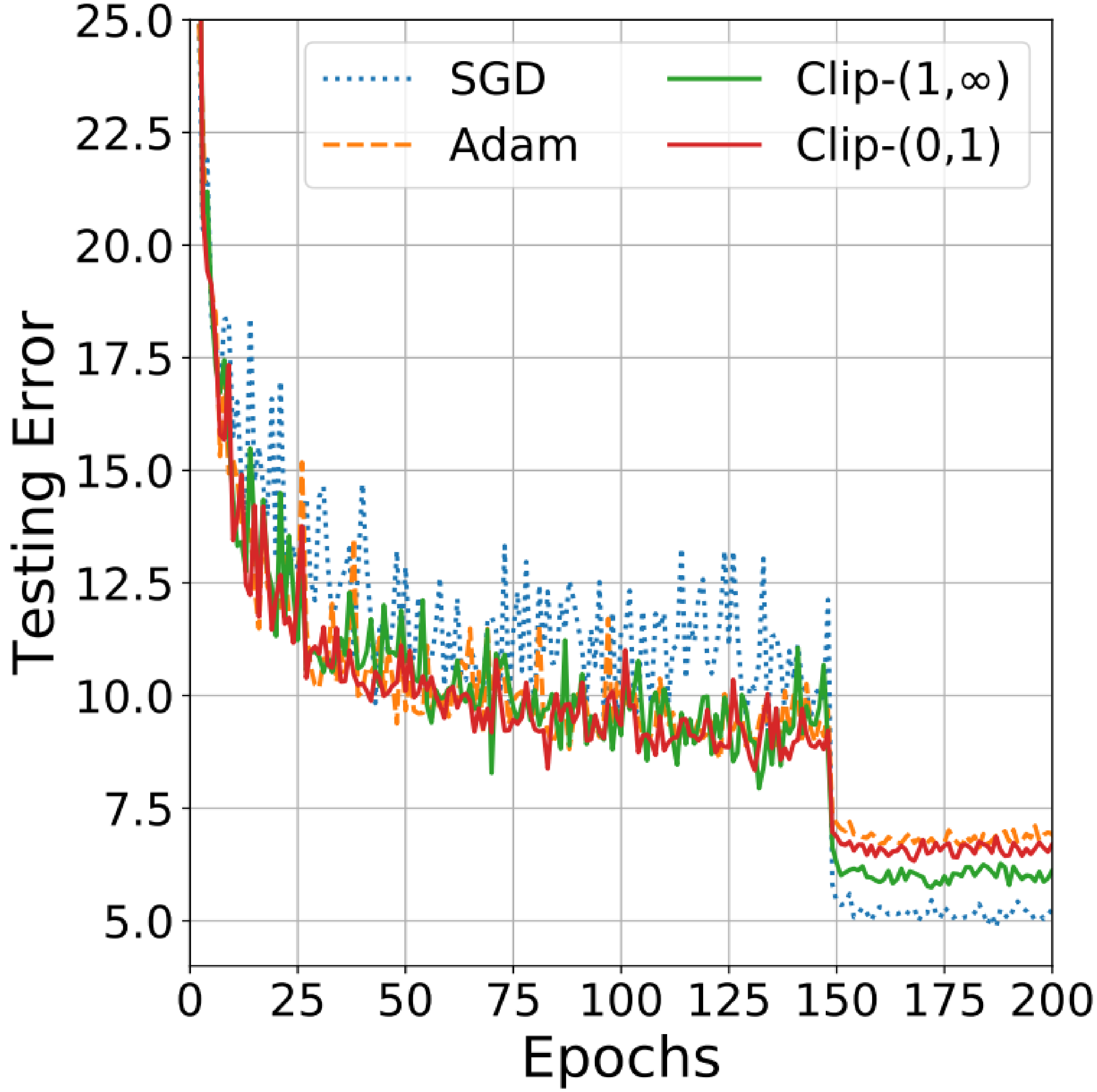
The function $\boldsymbol{clip}(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{b})$ clips the vector x element-wise such that the output is constrained to be in $[a, b]$.

$$w_k = w_{k-1} - \mathrm{clip}\left(\frac{\sqrt{1-\beta_2^k}}{1-\beta_1^k} \frac{\alpha_{k-1}}{\sqrt{v_{k-1}} + \epsilon}, p \cdot \alpha_{sgd}, q \cdot \alpha_{sgd} \right) m_{k-1}.$$



Training the **DenseNet** architecture on the CIFAR-10 data set with four optimizers

1. Adam's testing error decrease faster than SGD
2. SGD converges to the expected testing error of ≈ 5% while Adam stagnates in performance at around ≈ 7% error.
3. Adam(with clip)'s performance are better than Adam(default)



Training the **DenseNet** architecture on the CIFAR-10 data set with four optimizers

1. Adam's testing error decrease faster than SGD
2. SGD converges to the expected testing error of $\approx$ 5% while Adam stagnates in performance at around $\approx$ 7% error.
3. Adam(with clip)'s performance are better than Adam(default)

SWATS: Adam+SGD

- Why not combine Adam and SGD?
  - Use Adam first
  - Then switch to SGD

- Questions:
  - **When to switch Adam to SGD?**
  - **What learning rate to choose for SGD after the switch?**
- Answer:
  - **SWATS**

SWATS: Adam+SGD

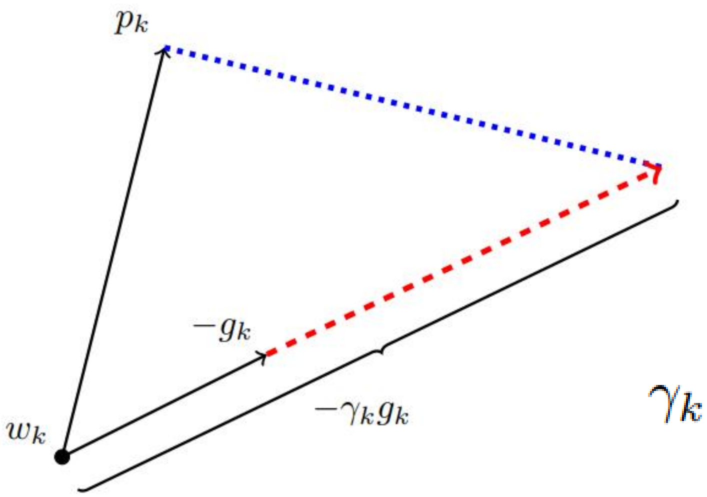
Algorithm 1 SWATS

Inputs: Objective function f, initial point w_0, learning rate $\alpha = 10^{-3}$, accumulator coefficients $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon = 10^{-9}$, phase=`Adam`.

Initialize $k \leftarrow 0$, $m_k \leftarrow 0$, $a_k \leftarrow 0$, $\lambda_k \leftarrow 0$
while stopping criterion not met **do**
    $k = k + 1$
    Compute stochastic gradient $g_k = \hat{\nabla} f(w_{k-1})$
    if phase = `SGD` **then**
        $v_k = \beta_1 v_{k-1} + g_k$
        $w_k = w_{k-1} - (1 - \beta_1)\Lambda v_k$
        continue
    end if
    $m_k = \beta_1 m_{k-1} + (1 - \beta_1) g_k$
    $a_k = \beta_2 a_{k-1} + (1 - \beta_2) g_k^2$
    $p_k = -\alpha_k \frac{\sqrt{1-\beta_2^k}}{1-\beta_1^k} \frac{m_k}{\sqrt{a_k} + \epsilon}$
    $w_k = w_k + p_k$
    if $p_k^T g_k \neq 0$ **then**
        $\gamma_k = \frac{p_k^T p_k}{-p_k^T g_k}$
        $\lambda_k = \beta_2 \lambda_{k-1} + (1 - \beta_2)\gamma_k$
        if $k > 1$ and $\left| \frac{\lambda_k}{(1-\beta_2^k)} - \gamma_k \right| < \epsilon$ **then**
            phase = `SGD`
            $v_k = 0$
            $\Lambda = \lambda_k / (1 - \beta_2^k)$
        end if
    else
        $\lambda_k = \lambda_{k-1}$
    end if
end while
return w_k

SWATS: Adam+SGD

- What learning rate to choose for SGD after the switch?



$$\gamma_k = \frac{p_k^T p_k}{-p_k^T g_k},$$

Figure 3. Illustrating the learning rate for SGD (γ_k) estimated by our proposed projection given an iterate w_k, a stochastic gradient g_k and the Adam step p_k.

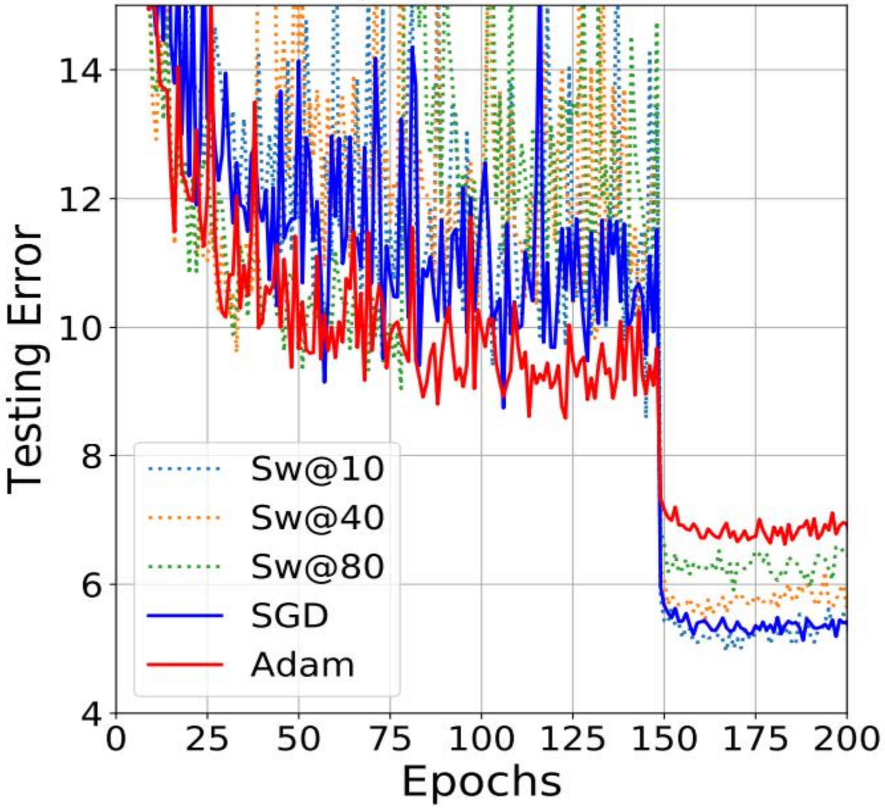
Algorithm 1 SWATS

Inputs: Objective function f, initial point w_0, learning rate $\alpha = 10^{-3}$, accumulator coefficients $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon = 10^{-9}$, phase=`Adam`.

```
Initialize k ← 0, m_k ← 0, a_k ← 0, λ_k ← 0
while stopping criterion not met do
    k = k + 1
    Compute stochastic gradient g_k = ∇̂f(w_{k-1})
    if phase = SGD then
        v_k = β_1 v_{k-1} + g_k
        w_k = w_{k-1} − (1 − β_1)Λ v_k
        continue
    end if
    m_k = β_1 m_{k-1} + (1 − β_1) g_k
    a_k = β_2 a_{k-1} + (1 − β_2) g_k^2
    p_k = −α_k (√(1 − β_2^k) / (1 − β_1^k)) · m_k / (√a_k + ε)
    w_k = w_k + p_k
    if p_k^T g_k ≠ 0 then
        γ_k = p_k^T p_k / (−p_k^T g_k)
        λ_k = β_2 λ_{k-1} + (1 − β_2) γ_k
        if k > 1 and |λ_k / (1 − β_2^k) − γ_k| < ε then
            phase = SGD
            v_k = 0
            Λ = λ_k / (1 − β_2^k)
        end if
    else
        λ_k = λ_{k-1}
    end if
end while
return w_k
```

SWATS: Adam+SGD

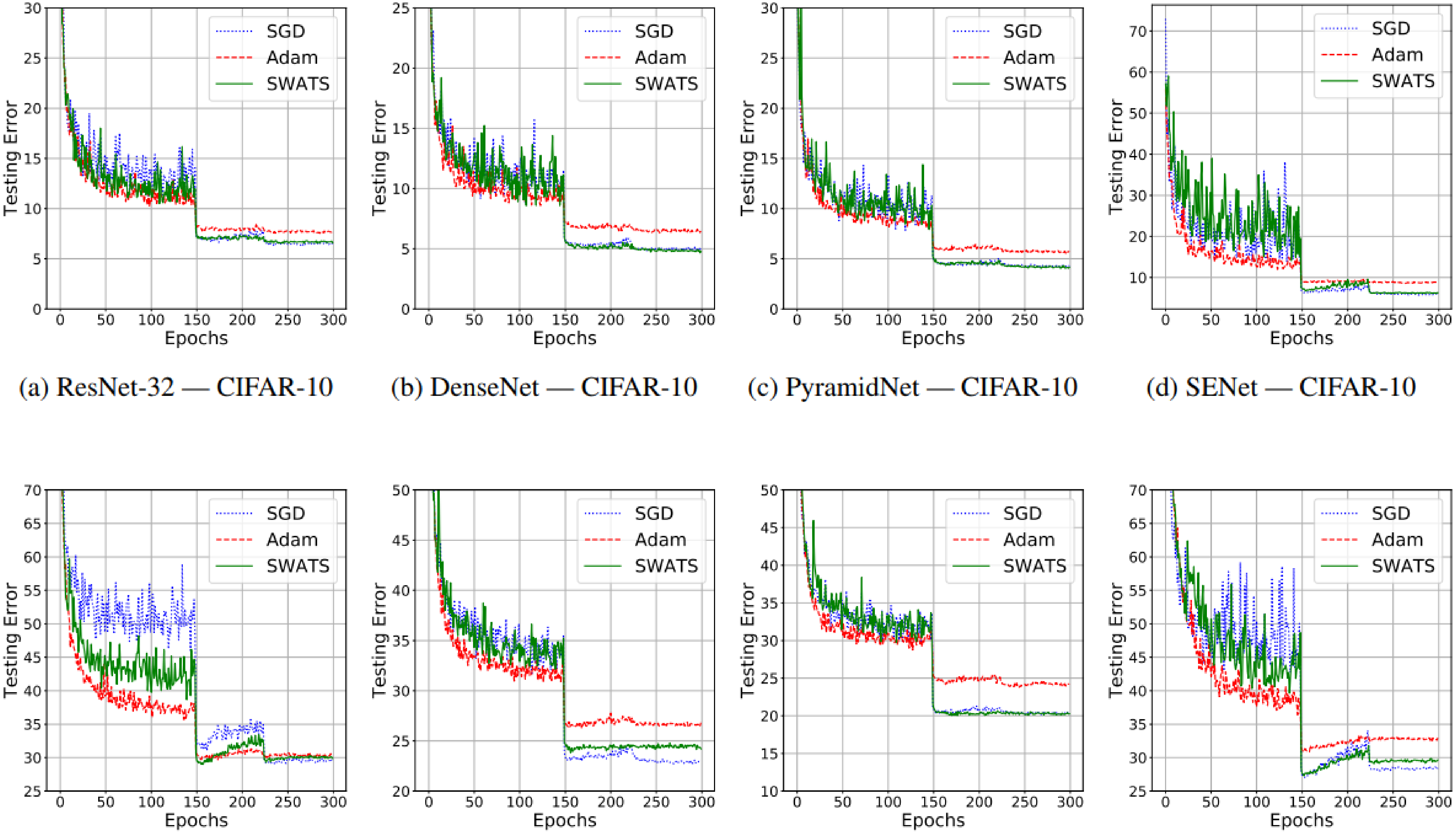
- When to switch Adam to SGD?



Algorithm 1 SWATS

Inputs: Objective function f, initial point w_0, learning rate $\alpha = 10^{-3}$, accumulator coefficients $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon = 10^{-9}$, phase=`Adam`.

Initialize $k \leftarrow 0$, $m_k \leftarrow 0$, $a_k \leftarrow 0$, $\lambda_k \leftarrow 0$
while stopping criterion not met **do**
  $k = k + 1$
  Compute stochastic gradient $g_k = \hat{\nabla} f(w_{k-1})$
  if phase = `SGD` **then**
    $v_k = \beta_1 v_{k-1} + g_k$
    $w_k = w_{k-1} - (1 - \beta_1)\Lambda v_k$
    continue
  end if
  $m_k = \beta_1 m_{k-1} + (1 - \beta_1) g_k$
  $a_k = \beta_2 a_{k-1} + (1 - \beta_2) g_k^2$
  $p_k = -\alpha_k \frac{\sqrt{1-\beta_2^k}}{1-\beta_1^k} \frac{m_k}{\sqrt{a_k}+\epsilon}$
  $w_k = w_k + p_k$
  if $p_k^T g_k \neq 0$ **then**
    $\gamma_k = \frac{p_k^T p_k}{-p_k^T g_k}$
    $\lambda_k = \beta_2 \lambda_{k-1} + (1 - \beta_2)\gamma_k$
    $\boxed{\textbf{if } k > 1 \text{ and } \left|\frac{\lambda_k}{(1-\beta_2^k)} - \gamma_k\right| < \epsilon \textbf{ then}}$
      phase = `SGD`
      $v_k = 0$
      $\Lambda = \lambda_k / (1 - \beta_2^k)$
    end if
  else
    $\lambda_k = \lambda_{k-1}$
  end if
end while
return w_k



(a) ResNet-32 — CIFAR-10 (b) DenseNet — CIFAR-10 (c) PyramidNet — CIFAR-10 (d) SENet — CIFAR-10

(e) ResNet-32 — CIFAR-100 (f) DenseNet — CIFAR-100 (g) PyramidNet — CIFAR-100 (h) SENet — CIFAR-100

Figure 4. Numerical experiments comparing SGD(M), Adam and SWATS with tuned learning rates on the ResNet-32, DenseNet, PyramidNet and SENet architectures on CIFAR-10 and CIFAR-100 data sets.

Choice of optimizer

Choice of optimizer

- The two recommended updates to use are either SGD+Nesterov Momentum or Adam
- Use Adam to check algorithm's correctness.
- Use SGD to training
  - Determine a good learning rate schedule
  - Adam may have lowest training error/loss, but not val.
- Use Adam first, then switch to SGD.(such as SWATS)
  - Consider when to switch and learning rate after switch