

Part

Appendices

The appendices are structured as follows. In Section A, we prove Theorem 1. In Section B, we review an error accumulation result of the Nesterov accelerated gradient with δ -inexact gradient. In Section C, we prove Theorem 2. In Section D, we provide some experimental details; in particular, the calibration of restarting hyperparameters. In Section E, we compare SRS GD with benchmark optimization algorithms on some other tasks, including training LSTM and Wasserstein GAN. In Section F, we provide detailed experimental settings in studying the effects of reducing the number of epoch in training deep neural networks with SRS GD, and we provide some more experimental results. In Section G and H, we further study the effects of restarting frequency and training with less epochs by using SRS GD. In Section I, we visualize the optimization trajectory of SRS GD and compare it with benchmark methods. A snippet of our implementation of SRS GD in PyTorch and Keras are available in Section J and K, respectively.

Table of Contents

Appendix	14
A Uncontrolled Bound of NASGD	15
A.1 Preliminaries	15
A.2 Uncontrolled Bound of NASGD: Analysis	16
B NAG with δ-Inexact Oracle & Experimental Settings in Section 3.1	19
C Convergence of SRS GD	20
C.1 Numerical Verification of the assumptions in Theorem 2	22
D Datasets and Implementation Details	23
D.1 CIFAR	23
D.2 ImageNet	23
D.3 Training ImageNet in Fewer Number of Epochs:	23
D.4 Details on Restarting Hyper-parameters Search	23
E SRS GD vs. SGD and SGD + NM on ImageNet Classification and Other Tasks	24
E.1 Comparing with SGD with Nesterov Momentum on ImageNet Classification	24
E.2 Long Short-Term Memory (LSTM) Training for Pixel-by-Pixel MNIST	24
E.3 Wasserstein Generative Adversarial Networks (WGAN) Training on MNIST	25
F Error Rate vs. Reduction in Training Epochs	27
F.1 Implementation Details	27
F.2 Short Training on CIFAR10/CIFAR100 Using SGD	28
F.3 Additional Experimental Results	28
G Impact of Restarting Frequency for ImageNet and CIFAR100	29
G.1 Implementation Details	29
G.2 Impact of the Growth Rate r	30
G.3 Additional Experimental Results	31
H Full Training with Less Epochs at the Intermediate Learning Rates	32
I Visualization of SRS GD’s trajectory	33
J SRS GD Implementation in Pytorch	35
K SRS GD Implementation in Keras	36

A UNCONTROLLED BOUND OF NASGD

Consider the following optimization problem

$$\min_{\mathbf{w}} f(\mathbf{w}), \quad (13)$$

where $f(\mathbf{w})$ is L -smooth and convex.

Start from \mathbf{w}^k , GD update, with step size $\frac{1}{r}$, can be obtained based on the minimization of the function

$$Q_r(\mathbf{v}, \mathbf{w}^k) := \langle \mathbf{v} - \mathbf{w}^k, \nabla f(\mathbf{w}^k) \rangle + \frac{r}{2} \|\mathbf{v} - \mathbf{w}^k\|_2^2. \quad (14)$$

With direct computation, we can get that

$$Q_r(\mathbf{v}^{k+1}, \mathbf{w}^k) - \min_{\mathbf{v}} Q_r(\mathbf{v}, \mathbf{w}^k) = \frac{\|\mathbf{g}^k - \nabla f(\mathbf{w}^k)\|_2}{2r},$$

where $\mathbf{g}^k := \frac{1}{m} \sum_{j=1}^m \nabla f_{i_j}(\mathbf{w}^k)$. We assume the variance is bounded, which gives The stochastic gradient rule, \mathcal{R}_s , satisfies $\mathbb{E}[Q_r(\mathbf{v}^{k+1}, \mathbf{w}^k) - \min_{\mathbf{v}} Q_r(\mathbf{v}, \mathbf{w}^k) | \chi^k] \leq \delta$, with δ being a constant and χ^k being the sigma algebra generated by $\mathbf{w}^1, \mathbf{w}^2, \dots, \mathbf{w}^k$, i.e.,

$$\chi^k := \sigma(\mathbf{w}^1, \mathbf{w}^2, \dots, \mathbf{w}^k).$$

NASGD can be reformulated as

$$\begin{aligned} \mathbf{v}^{k+1} &\approx \arg \min_{\mathbf{v}} Q_r(\mathbf{v}, \mathbf{w}^k) \text{ with rule } \mathcal{R}_s, \\ \mathbf{w}^{k+1} &= \mathbf{v}^{k+1} + \frac{t_k - 1}{t_{k+1}} (\mathbf{v}^{k+1} - \mathbf{v}^k), \end{aligned} \quad (15)$$

where $t_0 = 1$ and $t_{k+1} = (1 + \sqrt{1 + 4t_k^2})/2$.

A.1 PRELIMINARIES

To proceed, we introduce several definitions and some useful properties in variational and convex analysis. More detailed background can be found at Mordukhovich (2006); Nesterov (1998); Rockafellar & Wets (2009); Rockafellar (1970).

Let f be a convex function, we say that f is L -smooth (*gradient Lipschitz*) if f is differentiable and

$$\|\nabla f(\mathbf{v}) - \nabla f(\mathbf{w})\|_2 \leq L \|\mathbf{v} - \mathbf{w}\|_2,$$

and we say f is ν -strongly convex if for any $\mathbf{w}, \mathbf{v} \in \text{dom}(f)$

$$f(\mathbf{w}) \geq f(\mathbf{v}) + \langle \nabla f(\mathbf{v}), \mathbf{w} - \mathbf{v} \rangle + \frac{\nu}{2} \|\mathbf{w} - \mathbf{v}\|_2^2.$$

Below of this subsection, we list several basic but useful lemmas, the proof can be found in Nesterov (1998).

Lemma 1. *If f is ν -strongly convex, then for any $\mathbf{v} \in \text{dom}(f)$ we have*

$$f(\mathbf{v}) - f(\mathbf{v}^*) \geq \frac{\nu}{2} \|\mathbf{v} - \mathbf{v}^*\|_2^2, \quad (16)$$

where \mathbf{v}^* is the minimizer of f .

Lemma 2. *If f is L -smooth, for any $\mathbf{w}, \mathbf{v} \in \text{dom}(f)$,*

$$f(\mathbf{w}) \leq f(\mathbf{v}) + \langle \nabla f(\mathbf{v}), \mathbf{w} - \mathbf{v} \rangle + \frac{L}{2} \|\mathbf{w} - \mathbf{v}\|_2^2.$$

A.2 UNCONTROLLED BOUND OF NASGD: ANALYSIS

In this part, we denote

$$\tilde{\mathbf{v}}^{k+1} := \arg \min_{\mathbf{v}} Q_r(\mathbf{v}, \mathbf{w}^k). \quad (17)$$

Lemma 3. *If the constant $r > 0$, then*

$$\mathbb{E} (\|\mathbf{v}^{k+1} - \tilde{\mathbf{v}}^{k+1}\|_2^2 | \chi^k) \leq \frac{2\delta}{r}. \quad (18)$$

Proof. Note that $Q_r(\mathbf{v}, \mathbf{w}^k)$ is strongly convex with constant r , and $\tilde{\mathbf{v}}^{k+1}$ in (17) is the minimizer of $Q_r(\mathbf{v}, \mathbf{w}^k)$. With Lemma 1 we have

$$Q_r(\mathbf{v}^{k+1}, \mathbf{w}^k) - Q_r(\tilde{\mathbf{v}}^{k+1}, \mathbf{w}^k) \geq \frac{r}{2} \|\mathbf{v}^{k+1} - \tilde{\mathbf{v}}^{k+1}\|_2^2. \quad (19)$$

Notice that

$$\mathbb{E} [Q_r(\mathbf{v}^{k+1}, \mathbf{w}^k) - Q_r(\tilde{\mathbf{v}}^{k+1}, \mathbf{w}^k)] = \mathbb{E} [Q_r(\mathbf{v}^{k+1}, \mathbf{w}^k) - \min_{\mathbf{v}} Q_r(\mathbf{v}, \mathbf{w}^k)] \leq \delta.$$

The inequality (18) can be established by combining the above two inequalities. \square

Lemma 4. *If the constant satisfy $r > L$, then we have*

$$\begin{aligned} & \mathbb{E} \left(f(\tilde{\mathbf{v}}^{k+1}) + \frac{r}{2} \|\tilde{\mathbf{v}}^{k+1} - \mathbf{w}^k\|_2^2 - (f(\mathbf{v}^{k+1}) + \frac{r}{2} \|\mathbf{v}^{k+1} - \mathbf{w}^k\|_2^2) \right) \\ & \geq -\tau\delta - \frac{r-L}{2} \mathbb{E} [\|\mathbf{w}^k - \tilde{\mathbf{v}}^{k+1}\|_2^2], \end{aligned} \quad (20)$$

where $\tau = \frac{L^2}{r(r-L)} + 1$.

Proof. The convexity of f gives us

$$0 \leq \langle \nabla f(\mathbf{v}^{k+1}), \mathbf{v}^{k+1} - \tilde{\mathbf{v}}^{k+1} \rangle + f(\tilde{\mathbf{v}}^{k+1}) - f(\mathbf{v}^{k+1}). \quad (21)$$

From the definition of the stochastic gradient rule \mathcal{R}_s , we have

$$\begin{aligned} -\delta & \leq \mathbb{E} (Q_r(\tilde{\mathbf{v}}^{k+1}, \mathbf{w}^k) - Q_r(\mathbf{v}^{k+1}, \mathbf{w}^k)) \\ & = \mathbb{E} \left[\langle \tilde{\mathbf{v}}^{k+1} - \mathbf{w}^k, \nabla f(\mathbf{w}^k) \rangle + \frac{r}{2} \|\tilde{\mathbf{v}}^{k+1} - \mathbf{w}^k\|_2^2 \right] - \\ & \quad \mathbb{E} \left[\langle \mathbf{v}^{k+1} - \mathbf{w}^k, \nabla f(\mathbf{w}^k) \rangle + \frac{r}{2} \|\mathbf{v}^{k+1} - \mathbf{w}^k\|_2^2 \right]. \end{aligned} \quad (22)$$

With (21) and (22), we have

$$\begin{aligned} -\delta & \leq \left(f(\tilde{\mathbf{v}}^{k+1}) + \frac{r}{2} \|\tilde{\mathbf{v}}^{k+1} - \mathbf{w}^k\|_2^2 \right) - \left(f(\mathbf{v}^{k+1}) + \frac{r}{2} \|\mathbf{v}^{k+1} - \mathbf{w}^k\|_2^2 \right) + \\ & \quad \mathbb{E} \langle \nabla f(\mathbf{w}^k) - \nabla f(\tilde{\mathbf{v}}^{k+1}), \tilde{\mathbf{v}}^{k+1} - \mathbf{v}^{k+1} \rangle. \end{aligned} \quad (23)$$

With the Schwarz inequality $\langle \mathbf{a}, \mathbf{b} \rangle \leq \frac{\|\mathbf{a}\|_2^2}{2\mu} + \frac{\mu}{2} \|\mathbf{b}\|_2^2$ with $\mu = \frac{L^2}{r-L}$, $\mathbf{a} = \nabla f(\mathbf{v}^{k+1}) - \nabla f(\tilde{\mathbf{v}}^{k+1})$ and $\mathbf{b} = \mathbf{w}^k - \tilde{\mathbf{v}}^{k+1}$,

$$\begin{aligned} & \langle \nabla f(\mathbf{w}^k) - \nabla f(\tilde{\mathbf{v}}^{k+1}), \tilde{\mathbf{v}}^{k+1} - \mathbf{v}^{k+1} \rangle \\ & \leq \frac{(r-L)}{2L^2} \|\nabla f(\mathbf{w}^k) - \nabla f(\tilde{\mathbf{v}}^{k+1})\|_2^2 + \frac{L^2}{2(r-L)} \|\mathbf{v}^{k+1} - \tilde{\mathbf{v}}^{k+1}\|_2^2 \\ & \leq \frac{(r-L)}{2} \|\mathbf{w}^k - \tilde{\mathbf{v}}^{k+1}\|_2^2 + \frac{L^2}{2(r-L)} \|\mathbf{v}^{k+1} - \tilde{\mathbf{v}}^{k+1}\|_2^2. \end{aligned} \quad (24)$$

Combining (23) and (24), we have

$$\begin{aligned} -\delta & \leq \mathbb{E} \left(f(\tilde{\mathbf{v}}^{k+1}) + \frac{r}{2} \|\tilde{\mathbf{v}}^{k+1} - \mathbf{w}^k\|_2^2 \right) - \mathbb{E} \left(f(\mathbf{v}^{k+1}) + \frac{r}{2} \|\mathbf{v}^{k+1} - \mathbf{w}^k\|_2^2 \right) \\ & \quad + \frac{L^2}{2(r-L)} \mathbb{E} \|\mathbf{v}^{k+1} - \tilde{\mathbf{v}}^{k+1}\|_2^2 + \frac{r-L}{2} \mathbb{E} \|\mathbf{w}^k - \tilde{\mathbf{v}}^{k+1}\|_2^2. \end{aligned} \quad (25)$$

By rearrangement of the above inequality (25) and using Lemma 3, we obtain the result. \square

Lemma 5. *If the constants satisfy $r > L$, then we have the following bounds*

$$\mathbb{E} (f(\mathbf{v}^k) - f(\mathbf{v}^{k+1})) \geq \frac{r}{2} \mathbb{E} \|\mathbf{w}^k - \mathbf{v}^{k+1}\|_2^2 + r \mathbb{E} \langle \mathbf{w}^k - \mathbf{v}^k, \tilde{\mathbf{v}}^{k+1} - \mathbf{w}^k \rangle - \tau \delta, \quad (26)$$

$$\mathbb{E} (f(\mathbf{v}^*) - f(\mathbf{v}^{k+1})) \geq \frac{r}{2} \mathbb{E} \|\mathbf{w}^k - \mathbf{v}^{k+1}\|_2^2 + r \mathbb{E} \langle \mathbf{w}^k - \mathbf{v}^*, \tilde{\mathbf{v}}^{k+1} - \mathbf{w}^k \rangle - \tau \delta, \quad (27)$$

where $\tau := \frac{L^2}{r(r-L)} + 1$ and \mathbf{v}^* is the minimum .

Proof. With Lemma 2, we have

$$-f(\tilde{\mathbf{v}}^{k+1}) \geq -f(\mathbf{w}^k) - \langle \tilde{\mathbf{v}}^{k+1} - \mathbf{w}^k, \nabla f(\mathbf{w}^k) \rangle - \frac{L}{2} \|\tilde{\mathbf{v}}^{k+1} - \mathbf{w}^k\|_2^2. \quad (28)$$

Using the convexity of f , we have

$$f(\mathbf{v}^k) - f(\mathbf{w}^k) \geq \langle \mathbf{v}^k - \mathbf{w}^k, \nabla f(\mathbf{w}^k) \rangle,$$

i.e.,

$$f(\mathbf{v}^k) \geq f(\mathbf{w}^k) + \langle \mathbf{v}^k - \mathbf{w}^k, \nabla f(\mathbf{w}^k) \rangle. \quad (29)$$

According to the definition of $\tilde{\mathbf{v}}^{k+1}$ in (14), i.e.,

$$\tilde{\mathbf{v}}^{k+1} = \arg \min_{\mathbf{v}} Q_r(\mathbf{v}, \mathbf{w}^k) = \arg \min_{\mathbf{v}} \langle \mathbf{v} - \mathbf{w}^k, \nabla f(\mathbf{w}^k) \rangle + \frac{r}{2} \|\mathbf{v} - \mathbf{w}^k\|_2^2,$$

and the optimization condition gives

$$\tilde{\mathbf{v}}^{k+1} = \mathbf{w}^k - \frac{1}{r} \nabla f(\mathbf{w}^k). \quad (30)$$

Substituting (30) into (29), we obtain

$$f(\mathbf{v}^k) \geq f(\mathbf{w}^k) + \langle \mathbf{v}^k - \mathbf{w}^k, r(\mathbf{w}^k - \tilde{\mathbf{v}}^{k+1}) \rangle. \quad (31)$$

Direct summation of (28) and (31) gives

$$f(\mathbf{v}^k) - f(\tilde{\mathbf{v}}^{k+1}) \geq \left(r - \frac{L}{2}\right) \|\tilde{\mathbf{v}}^{k+1} - \mathbf{w}^k\|_2^2 + r \langle \mathbf{w}^k - \mathbf{v}^k, \tilde{\mathbf{v}}^{k+1} - \mathbf{w}^k \rangle. \quad (32)$$

Summing (32) and (20), we obtain the inequality (26)

$$\mathbb{E} [f(\mathbf{v}^k) - f(\mathbf{v}^{k+1})] \geq \frac{r}{2} \mathbb{E} \|\mathbf{w}^k - \mathbf{v}^{k+1}\|_2^2 + r \mathbb{E} \langle \mathbf{w}^k - \mathbf{v}^k, \tilde{\mathbf{v}}^{k+1} - \mathbf{w}^k \rangle - \tau \delta. \quad (33)$$

On the other hand, with the convexity of f , we have

$$f(\mathbf{v}^*) - f(\mathbf{w}^k) \geq \langle \mathbf{v}^* - \mathbf{w}^k, \nabla f(\mathbf{w}^k) \rangle = \langle \mathbf{v}^* - \mathbf{w}^k, r(\mathbf{w}^k - \tilde{\mathbf{v}}^{k+1}) \rangle. \quad (34)$$

The summation of (28) and (34) results in

$$f(\mathbf{v}^*) - f(\tilde{\mathbf{v}}^{k+1}) \geq \left(r - \frac{L}{2}\right) \|\mathbf{w}^k - \tilde{\mathbf{v}}^{k+1}\|_2^2 + r \langle \mathbf{w}^k - \mathbf{v}^*, \tilde{\mathbf{v}}^{k+1} - \mathbf{w}^k \rangle. \quad (35)$$

Summing (35) and (20), we obtain

$$\mathbb{E} (f(\mathbf{v}^*) - f(\mathbf{v}^{k+1})) \geq \frac{r}{2} \mathbb{E} \|\mathbf{w}^k - \mathbf{v}^{k+1}\|_2^2 + r \mathbb{E} \langle \mathbf{w}^k - \mathbf{v}^*, \tilde{\mathbf{v}}^{k+1} - \mathbf{w}^k \rangle - \tau \delta, \quad (36)$$

which is the same as (27). \square

Theorem 3 (Uncontrolled Bound of NASGD (Theorem 1 with detailed bounded)). *Let the constant r satisfy $r < L$ and the sequence $\{\mathbf{v}^k\}_{k \geq 0}$ be generated by NASGD with stochastic gradient that has bounded variance. By using any constant step size $s_k \equiv s \leq 1/L$, then we have*

$$\mathbb{E}[f(\mathbf{v}^k) - \min_{\mathbf{v}} f(\mathbf{v})] \leq \left(\frac{2\tau\delta}{r} + R^2\right) \frac{4k}{3}. \quad (37)$$

Proof. We denote

$$F^k := \mathbb{E}(f(\mathbf{v}^k) - f(\mathbf{v}^*)).$$

By (26) $\times (t_k - 1)$ + (27), we have

$$\begin{aligned} \frac{2[(t_k - 1)F^k - t_k F^{k+1}]}{r} &\geq t_k \mathbb{E} \|\mathbf{v}^{k+1} - \mathbf{w}^k\|_2^2 \\ &+ 2\mathbb{E} \langle \tilde{\mathbf{v}}^{k+1} - \mathbf{w}^k, t_k \mathbf{w}^k - (t_k - 1)\mathbf{v}^k - \mathbf{v}^* \rangle - \frac{2\tau t_k \delta}{r}. \end{aligned} \quad (38)$$

With $t_{k-1}^2 = t_k^2 - t_k$, (38) $\times t_k$ yields

$$\begin{aligned} \frac{2[t_{k-1}^2 F^k - t_k^2 F^{k+1}]}{r} &\geq \mathbb{E} \|t_k \mathbf{v}^{k+1} - t_k \mathbf{w}^k\|_2^2 \\ &+ 2t_k \mathbb{E} \langle \tilde{\mathbf{v}}^{k+1} - \mathbf{w}^k, t_k \mathbf{w}^k - (t_k - 1)\mathbf{v}^k - \mathbf{v}^* \rangle - \frac{2\tau t_k^2 \delta}{r} \end{aligned} \quad (39)$$

Substituting $\mathbf{a} = t_k \mathbf{v}^{k+1} - (t_k - 1)\mathbf{v}^k - \mathbf{v}^*$ and $\mathbf{b} = t_k \mathbf{w}^k - (t_k - 1)\mathbf{v}^k - \mathbf{v}^*$ into identity

$$\|\mathbf{a} - \mathbf{b}\|_2^2 + 2\langle \mathbf{a} - \mathbf{b}, \mathbf{b} \rangle = \|\mathbf{a}\|_2^2 - \|\mathbf{b}\|_2^2. \quad (40)$$

It follows that

$$\begin{aligned} &\mathbb{E} \|t_k \mathbf{v}^{k+1} - t_k \mathbf{w}^k\|_2^2 + 2t_k \mathbb{E} \langle \tilde{\mathbf{v}}^{k+1} - \mathbf{w}^k, t_k \mathbf{w}^k - (t_k - 1)\mathbf{v}^k - \mathbf{v}^* \rangle \\ = &\mathbb{E} \|t_k \mathbf{v}^{k+1} - t_k \mathbf{w}^k\|_2^2 + 2t_k \mathbb{E} \langle \mathbf{v}^{k+1} - \mathbf{w}^k, t_k \mathbf{w}^k - (t_k - 1)\mathbf{v}^k - \mathbf{v}^* \rangle \\ &+ 2t_k \mathbb{E} \langle \tilde{\mathbf{v}}^{k+1} - \mathbf{v}^{k+1}, t_k \mathbf{w}^k - (t_k - 1)\mathbf{v}^k - \mathbf{v}^* \rangle \\ \stackrel{(40)}{=} &\mathbb{E} \|t_k \mathbf{v}^{k+1} - (t_k - 1)\mathbf{v}^k - \mathbf{v}^*\|_2^2 - \|t_k \mathbf{w}^k - (t_k - 1)\mathbf{v}^k - \mathbf{v}^*\|_2^2 \\ &+ 2t_k \mathbb{E} \langle \tilde{\mathbf{v}}^{k+1} - \mathbf{v}^{k+1}, t_k \mathbf{w}^k - (t_k - 1)\mathbf{v}^k - \mathbf{v}^* \rangle \\ = &\mathbb{E} \|t_k \mathbf{v}^{k+1} - (t_k - 1)\mathbf{v}^k - \mathbf{v}^*\|_2^2 - \mathbb{E} \|t_{k-1} \mathbf{v}^k - (t_{k-1} - 1)\mathbf{v}^{k-1} - \mathbf{v}^*\|_2^2 \\ + &2t_k \mathbb{E} \langle \tilde{\mathbf{v}}^{k+1} - \mathbf{v}^{k+1}, t_{k-1} \mathbf{v}^k - (t_{k-1} - 1)\mathbf{v}^{k-1} - \mathbf{v}^* \rangle. \end{aligned} \quad (41)$$

In the third identity, we used the fact $t_k \mathbf{w}^k = t_k \mathbf{v}^k + (t_{k-1} - 1)(\mathbf{v}^k - \mathbf{v}^{k-1})$. If we denote $u^k = \mathbb{E} \|t_{k-1} \mathbf{v}^k - (t_{k-1} - 1)\mathbf{v}^{k-1} - \mathbf{v}^*\|_2^2$, (39) can be rewritten as

$$\begin{aligned} \frac{2t_k^2 F^{k+1}}{r} + u^{k+1} &\leq \frac{2t_{k-1}^2 F^k}{r} + u^k + \frac{2\tau t_k^2 \delta}{r} \\ &+ 2t_k \mathbb{E} \langle \mathbf{v}^{k+1} - \tilde{\mathbf{v}}^{k+1}, t_{k-1} \mathbf{v}^k - (t_{k-1} - 1)\mathbf{v}^{k-1} - \mathbf{v}^* \rangle \\ &\leq \frac{2t_k^2 F^k}{r} + u^k + \frac{2\tau t_k^2 \delta}{r} + t_{k-1}^2 R^2, \end{aligned} \quad (42)$$

where we used

$$\begin{aligned} &2t_k \mathbb{E} \langle \mathbf{v}^{k+1} - \tilde{\mathbf{v}}^{k+1}, t_{k-1} \mathbf{v}^k - (t_{k-1} - 1)\mathbf{v}^{k-1} - \mathbf{v}^* \rangle \\ \leq &t_k^2 \mathbb{E} \|\mathbf{v}^{k+1} - \tilde{\mathbf{v}}^{k+1}\|_2^2 + \mathbb{E} \|t_{k-1} \mathbf{v}^k - (t_{k-1} - 1)\mathbf{v}^{k-1} - \mathbf{v}^*\|_2^2 \\ = &2t_k^2 \delta / r + t_{k-1}^2 R^2. \end{aligned}$$

Denoting

$$\xi_k := \frac{2t_{k-1}^2 F^k}{r} + u^k,$$

then, we have

$$\xi_{k+1} \leq \xi_0 + \left(\frac{2\tau\delta}{r} + R^2\right) \sum_{i=1}^k t_i^2 = \left(\frac{2\tau\delta}{r} + R^2\right) \frac{k^3}{3}. \quad (43)$$

With the fact, $\xi_k \geq \frac{2t_{k-1}^2 F^k}{r} \geq k^2 F^k / 4$, we then proved the result. \square

B NAG WITH δ -INEXACT ORACLE & EXPERIMENTAL SETTINGS IN SECTION 3.1

In Devolder et al. (2014), the authors defines δ -inexact gradient oracle for convex smooth optimization as follows:

Definition 1 (δ -Inexact Oracle). *Devolder et al. (2014)* For a convex L -smooth function $f : \mathbb{R}^d \rightarrow \mathbb{R}$. For $\forall \mathbf{w} \in \mathbb{R}^d$ and exact first-order oracle returns a pair $(f(\mathbf{w}), \nabla f(\mathbf{w})) \in \mathbb{R} \times \mathbb{R}^d$ so that for $\forall \mathbf{v} \in \mathbb{R}^d$ we have

$$0 \leq f(\mathbf{v}) - (f(\mathbf{w}) + \langle \nabla f(\mathbf{w}), \mathbf{v} - \mathbf{w} \rangle) \leq \frac{L}{2} \|\mathbf{w} - \mathbf{v}\|_2^2.$$

A δ -inexact oracle returns a pair $(f^\delta(\mathbf{w}), \nabla f^\delta(\mathbf{w})) \in \mathbb{R} \times \mathbb{R}^d$ so that $\forall \mathbf{v} \in \mathbb{R}^d$ we have

$$0 \leq f(\mathbf{v}) - (f^\delta(\mathbf{w}) + \langle \nabla f^\delta(\mathbf{w}), \mathbf{v} - \mathbf{w} \rangle) \leq \frac{L}{2} \|\mathbf{w} - \mathbf{v}\|_2^2 + \delta.$$

We have the following convergence results of GD and NAG under a δ -Inexact Oracle for convex smooth optimization.

Theorem 4. *Devolder et al. (2014)*³ Consider

$$\min f(\mathbf{w}), \quad \mathbf{w} \in \mathbb{R}^d,$$

where $f(\mathbf{w})$ is convex and L -smooth with \mathbf{w}^* being the minimum. Given access to δ -inexact oracle, GD with step size $1/L$ returns a point \mathbf{w}^k after k steps so that

$$f(\mathbf{w}^k) - f(\mathbf{w}^*) = O\left(\frac{L}{k}\right) + \delta.$$

On the other hand, NAG, with step size $1/L$ returns

$$f(\mathbf{w}^k) - f(\mathbf{w}^*) = O\left(\frac{L}{k^2}\right) + O(k\delta).$$

Theorem 4 says that NAG may not robust to a δ -inexact gradient. In the following, we will study the numerical behavior of a variety of first-order algorithms for convex smooth optimizations with the following different inexact gradients.

Constant Variance Gaussian Noise: We consider the inexact oracle where the true gradient is contaminated with a Gaussian noise $\mathcal{N}(0, 0.001^2)$. We run 50K iterations of different algorithms. For SRNAG, we restart after every 200 iterations. Fig. 2 (b) shows the iteration vs. optimal gap, $f(\mathbf{x}^k) - f(\mathbf{x}^*)$, with \mathbf{x}^* being the minimum. NAG with the inexact gradient due to constant variance noise does not converge. GD performs almost the same as ARNAG asymptotically, because ARNAG restarts too often and almost degenerates into GD. GD with constant momentum outperforms the three schemes above, and SRNAG slightly outperforms GD with constant momentum.

Decaying Variance Gaussian Noise: Again, consider minimizing (8) with the same experimental setting as before except that $\nabla f(\mathbf{x})$ is now contaminated with a decaying Gaussian noise $\mathcal{N}(0, (\frac{0.1}{\lfloor t/100 \rfloor + 1})^2)$. For SRNAG, we restart every 200 iterations in the first $10k$ iterations, and restart every 400 iterations in the remaining 40K iterations. Fig. 2 (c) shows the iteration vs. optimal gap by different schemes. ARNAG still performs almost the same as GD. The path of NAG is oscillatory. GD with constant momentum again outperforms the previous three schemes. Here SRNAG significantly outperforms all the other schemes.

Logistic Regression for MNIST Classification: We apply the above schemes with stochastic gradient to train a logistic regression model for MNIST classification LeCun & Cortes (2010). We consider five different schemes, namely, SGD, SGD + (constant) momentum, NASGD, ASGD, and SRSGD. In ARSGD, we perform restart based on the loss value of the mini-batch training data. In SRSGD, we restart the NAG momentum after every 10 iterations. We train the logistic regression model with a ℓ_2 weight decay of 10^{-4} by running 20 epochs using different schemes with batch size of 128. The step sizes for all the schemes are set to 0.01. Fig. 3 (a) plots the training loss vs. iteration. In this case, NASGD does not converge, and SGD with momentum does not speed up SGD. ARSGD's performance is on par with SGD's. Again, SRSGD gives the best performance with the smallest training loss among these five schemes.

³We adopt the result from Hardt (2014).

C CONVERGENCE OF SRS GD

We prove the convergence of Nesterov accelerated SGD with scheduled restart, i.e., the convergence of SRS GD. We denote that $\theta^k := \frac{t_k-1}{t_{k+1}}$ in the Nesterov iteration and $\hat{\theta}^k$ is its use in the restart version, i.e., SRS GD. For any restart frequency F (positive integer), we have $\hat{\theta}^k = \theta^{k-\lfloor k/F \rfloor * F}$. In the restart version, we can see that

$$\hat{\theta}^k \leq \theta^F =: \bar{\theta} < 1.$$

Lemma 6. *Let the constant satisfies $r > L$ and the sequence $\{\mathbf{v}^k\}_{k \geq 0}$ be generated by the SRS GD with restart frequency F (any positive integer), we have*

$$\sum_{i=1}^k \|\mathbf{v}^i - \mathbf{v}^{i-1}\|_2^2 \leq \frac{r^2 k R^2}{(1 - \bar{\theta})^2}, \quad (44)$$

where $\bar{\theta} := \theta^F < 1$ and $R := \sup_{\mathbf{x}} \{\|\nabla f(\mathbf{x})\|_2\}$.

Proof. It holds that

$$\begin{aligned} \|\mathbf{v}^{k+1} - \mathbf{w}^k\|_2 &= \|\mathbf{v}^{k+1} - \mathbf{v}^k + \mathbf{v}^k - \mathbf{w}^k\|_2 \\ &\geq \|\mathbf{v}^{k+1} - \mathbf{v}^k\|_2 - \|\mathbf{v}^k - \mathbf{w}^k\|_2 \\ &\geq \|\mathbf{v}^{k+1} - \mathbf{v}^k\|_2 - \bar{\theta} \|\mathbf{v}^k - \mathbf{v}^{k-1}\|_2. \end{aligned} \quad (45)$$

Thus,

$$\begin{aligned} \|\mathbf{v}^{k+1} - \mathbf{w}^k\|_2^2 &\geq (\|\mathbf{v}^{k+1} - \mathbf{v}^k\|_2 - \bar{\theta} \|\mathbf{v}^k - \mathbf{v}^{k-1}\|_2)^2 \\ &= \|\mathbf{v}^{k+1} - \mathbf{v}^k\|_2^2 - 2\bar{\theta} \|\mathbf{v}^k - \mathbf{v}^{k-1}\|_2 \|\mathbf{v}^k - \mathbf{v}^{k-1}\|_2 + \bar{\theta}^2 \|\mathbf{v}^k - \mathbf{v}^{k-1}\|_2^2 \\ &\geq (1 - \bar{\theta}) \|\mathbf{v}^{k+1} - \mathbf{v}^k\|_2^2 - \bar{\theta}(1 - \bar{\theta}) \|\mathbf{v}^{k+1} - \mathbf{v}^k\|_2^2. \end{aligned} \quad (46)$$

Summing (46) from $k = 1$ to K , we get

$$(1 - \bar{\theta})^2 \sum_{k=1}^K \|\mathbf{v}^k - \mathbf{v}^{k-1}\|_2^2 \leq \sum_{k=1}^K \|\mathbf{v}^{k+1} - \mathbf{w}^k\|_2^2 \leq r^2 K R^2. \quad (47)$$

□

In the following, we denote

$$\mathcal{A} := \{k \in \mathbb{Z}^+ \mid \mathbb{E}f(\mathbf{v}^k) \geq \mathbb{E}f(\mathbf{v}^{k-1})\}.$$

Theorem 5 (Convergence of SRS GD). *(Theorem 2 with detailed bound) Suppose $f(\mathbf{w})$ is L -smooth. Consider the sequence $\{\mathbf{w}^k\}_{k \geq 0}$ generated by (10) with stochastic gradient that is bounded and has bound variance. Using any restart frequency F and any constant step size $s_k := s \leq 1/L$. Assume that $\sum_{k \in \mathcal{A}} (\mathbb{E}f(\mathbf{w}^{k+1}) - \mathbb{E}f(\mathbf{w}^k)) = \tilde{R} < +\infty$, then we have*

$$\min_{1 \leq k \leq K} \{\mathbb{E}\|\nabla f(\mathbf{w}^k)\|_2^2\} \leq \frac{rR^2}{(1 - \bar{\theta})^2} \frac{L(1 + \bar{\theta})}{2} + \frac{rLR^2}{2} + \frac{\tilde{\theta}\tilde{R}}{rK}. \quad (48)$$

If $f(\mathbf{w})$ is further convex and the set $\mathcal{B} := \{k \in \mathbb{Z}^+ \mid \mathbb{E}\|\mathbf{w}^{k+1} - \mathbf{w}^*\|^2 \geq \mathbb{E}\|\mathbf{w}^k - \mathbf{w}^*\|^2\}$ obeys $\sum_{k \in \mathcal{B}} (\mathbb{E}f(\mathbf{w}^{k+1}) - \mathbb{E}f(\mathbf{w}^k)) = \hat{R} < +\infty$, then

$$\min_{1 \leq k \leq K} \{\mathbb{E}(f(\mathbf{w}^k) - f(\mathbf{w}^*))\} \leq \frac{\|\mathbf{w}^0 - \mathbf{w}^*\|^2 + \hat{R}}{2\gamma k} + \frac{\gamma R^2}{2}, \quad (49)$$

where \mathbf{w}^* is the minimum of f . To obtain ϵ ($\forall \epsilon > 0$) error, we set $s = O(\epsilon)$ and $K = O(1/\epsilon^2)$.

Proof. Firstly, we show the convergence of SRS GD for nonconvex optimization. L -smoothness of f , i.e., Lipschitz gradient continuity, gives us

$$f(\mathbf{v}^{k+1}) \leq f(\mathbf{w}^k) + \langle \nabla f(\mathbf{w}^k), \mathbf{v}^{k+1} - \mathbf{w}^k \rangle + \frac{L}{2} \|\mathbf{v}^{k+1} - \mathbf{w}^k\|_2^2. \quad (50)$$

Taking expectation, we get

$$\mathbb{E}f(\mathbf{v}^{k+1}) \leq \mathbb{E}f(\mathbf{w}^k) - r\mathbb{E}\|\nabla f(\mathbf{w}^k)\|_2^2 + \frac{r^2LR^2}{2}. \quad (51)$$

On the other hand, we have

$$f(\mathbf{w}^k) \leq f(\mathbf{v}^k) + \hat{\theta}^k \langle \nabla f(\mathbf{v}^k), \mathbf{v}^k - \mathbf{v}^{k-1} \rangle + \frac{L(\hat{\theta}^k)^2}{2} \|\mathbf{v}^k - \mathbf{v}^{k-1}\|_2^2. \quad (52)$$

Then, we have

$$\begin{aligned} \mathbb{E}f(\mathbf{v}^{k+1}) &\leq \mathbb{E}f(\mathbf{v}^k) + \hat{\theta}^k \mathbb{E} \langle \nabla f(\mathbf{v}^k), \mathbf{v}^k - \mathbf{v}^{k-1} \rangle \\ &+ \frac{L(\hat{\theta}^k)^2}{2} \mathbb{E} \|\mathbf{v}^k - \mathbf{v}^{k-1}\|_2^2 - r\mathbb{E} \|\nabla f(\mathbf{w}^k)\|_2^2 + \frac{r^2LR^2}{2}. \end{aligned} \quad (53)$$

We also have

$$\hat{\theta}^k \langle \nabla f(\mathbf{v}^k), \mathbf{v}^k - \mathbf{v}^{k-1} \rangle \leq \hat{\theta}^k \left(f(\mathbf{v}^k) - f(\mathbf{v}^{k-1}) + \frac{L}{2} \|\mathbf{v}^k - \mathbf{v}^{k-1}\|_2^2 \right). \quad (54)$$

We then get that

$$\mathbb{E}f(\mathbf{v}^{k+1}) \leq \mathbb{E}f(\mathbf{v}^k) + \hat{\theta}^k (\mathbb{E}f(\mathbf{v}^k) - \mathbb{E}f(\mathbf{v}^{k-1})) - r\mathbb{E} \|\nabla f(\mathbf{w}^k)\|_2^2 + A_k, \quad (55)$$

where

$$A_k := \mathbb{E} \frac{L}{2} \|\mathbf{v}^k - \mathbf{v}^{k-1}\|_2^2 + \frac{L(\hat{\theta}^k)^2}{2} \mathbb{E} \|\mathbf{v}^k - \mathbf{v}^{k-1}\|_2^2 + \frac{r^2LR^2}{2}.$$

Summing the inequality gives us

$$\begin{aligned} \mathbb{E}f(\mathbf{v}^{K+1}) &\leq \mathbb{E}f(\mathbf{v}^0) + \tilde{\theta} \sum_{k \in \mathcal{A}} (\mathbb{E}f(\mathbf{v}^k) - \mathbb{E}f(\mathbf{v}^{k-1})) \\ &- r \sum_{k=1}^K \mathbb{E} \|\nabla f(\mathbf{w}^k)\|_2^2 + \sum_{k=1}^K A_k. \end{aligned} \quad (56)$$

It is easy to see that

$$\tilde{\theta} \sum_{k \in \mathcal{A}} (\mathbb{E}f(\mathbf{v}^k) - \mathbb{E}f(\mathbf{v}^{k-1})) = \tilde{\theta} \tilde{R}.$$

We get the result by using Lemma 6

Secondly, we prove the convergence of SRS GD for convex optimization. Let \mathbf{w}^* be the minimizer of f . We have

$$\begin{aligned} \mathbb{E} \|\mathbf{v}^{k+1} - \mathbf{w}^*\|_2^2 &= \mathbb{E} \|\mathbf{w}^k - \gamma \nabla f(\mathbf{w}^k) - \mathbf{w}^*\|_2^2 \\ &= \mathbb{E} \|\mathbf{w}^k - \mathbf{w}^*\|_2^2 - 2\gamma \mathbb{E} \langle \nabla f(\mathbf{w}^k), \mathbf{w}^k - \mathbf{w}^* \rangle + \gamma^2 \mathbb{E} \|\nabla f(\mathbf{w}^k)\|_2^2 \\ &\leq \mathbb{E} \|\mathbf{w}^k - \mathbf{w}^*\|_2^2 - 2\gamma \mathbb{E} \langle \nabla f(\mathbf{w}^k), \mathbf{w}^k - \mathbf{w}^* \rangle + \gamma^2 R^2. \end{aligned} \quad (57)$$

We can also derive

$$\begin{aligned} \mathbb{E} \|\mathbf{w}^k - \mathbf{w}^*\|_2 &= \mathbb{E} \|\mathbf{v}^k + \hat{\theta}^k (\mathbf{v}^k - \mathbf{v}^{k-1}) - \mathbf{w}^*\|_2^2 \\ &= \mathbb{E} \|\mathbf{v}^k - \mathbf{w}^*\|_2^2 + 2\hat{\theta}^k \mathbb{E} \langle \mathbf{v}^k - \mathbf{v}^{k-1}, \mathbf{v}^k - \mathbf{w}^* \rangle + (\hat{\theta}^k)^2 \mathbb{E} \|\mathbf{v}^k - \mathbf{v}^{k-1}\|_2^2 \\ &= \mathbb{E} \|\mathbf{v}^k - \mathbf{w}^*\|_2^2 + \hat{\theta}^k \mathbb{E} (\|\mathbf{v}^k - \mathbf{w}^*\|_2^2 + \|\mathbf{v}^{k-1} - \mathbf{v}^k\|_2^2 - \|\mathbf{v}^{k-1} - \mathbf{w}^*\|_2^2) \\ &+ (\hat{\theta}^k)^2 \mathbb{E} \|\mathbf{v}^k - \mathbf{v}^{k-1}\|_2^2 \\ &= \mathbb{E} \|\mathbf{v}^k - \mathbf{w}^*\|_2^2 + \hat{\theta}^k \mathbb{E} (\|\mathbf{v}^k - \mathbf{w}^*\|_2^2 - \|\mathbf{v}^{k-1} - \mathbf{w}^*\|_2^2) + 2(\hat{\theta}^k)^2 \mathbb{E} \|\mathbf{v}^k - \mathbf{v}^{k-1}\|_2^2, \end{aligned}$$

where we used the following identity

$$(\mathbf{a} - \mathbf{b})^T (\mathbf{a} - \mathbf{b}) = \frac{1}{2} [\|\mathbf{a} - \mathbf{d}\|_2^2 - \|\mathbf{a} - \mathbf{c}\|_2^2 + \|\mathbf{b} - \mathbf{c}\|_2^2 - \|\mathbf{b} - \mathbf{d}\|_2^2].$$

Then, we have

$$\begin{aligned} \mathbb{E}\|\mathbf{v}^{k+1} - \mathbf{w}^*\|_2^2 &\leq \mathbb{E}\|\mathbf{v}^k - \mathbf{w}^*\|_2^2 - 2\gamma\mathbb{E}\langle \nabla f(\mathbf{w}^k), \mathbf{w}^k - \mathbf{w}^* \rangle + 2(\hat{\theta}^k)^2\mathbb{E}\|\mathbf{v}^k - \mathbf{v}^{k-1}\|_2^2 \\ &\quad + r^2R^2 + \hat{\theta}^k\mathbb{E}(\|\mathbf{v}^k - \mathbf{w}^*\|_2^2 - \|\mathbf{v}^{k-1} - \mathbf{w}^*\|_2^2). \end{aligned} \quad (58)$$

We then get that

$$\begin{aligned} 2\gamma\mathbb{E}(f(\mathbf{w}^k) - f(\mathbf{w}^*)) &\leq \mathbb{E}\|\mathbf{v}^k - \mathbf{w}^*\|_2^2 - \mathbb{E}\|\mathbf{v}^{k+1} - \mathbf{w}^*\|_2^2 \\ &\quad + \hat{\theta}^k(\mathbb{E}\|\mathbf{v}^k - \mathbf{w}^*\|_2^2 - \mathbb{E}\|\mathbf{v}^{k-1} - \mathbf{w}^*\|_2^2) + r^2R^2. \end{aligned} \quad (59)$$

Summing the inequality gives us the desired convergence result for convex optimization. \square

C.1 NUMERICAL VERIFICATION OF THE ASSUMPTIONS IN THEOREM 2

In this part, we numerically verify the assumptions in Theorem 2. In particular, we apply SRS GD with learning rate 0.1 to train LeNet⁴ for MNIST classification (we test on MNIST due to extremely large computational cost). We conduct numerical verification as follows: starting from a given point \mathbf{w}^0 , we randomly sample 469 mini-batches (note in total we have 469 batches in the training data) with batch size 128 and compute the stochastic gradient using each mini-batch. Next, we advance to the next step with each of these 469 stochastic gradients and get the approximated $\mathbb{E}f(\mathbf{w}^1)$. We randomly choose one of these 469 positions as the updated weights of our model. By iterating the above procedure, we can get $\mathbf{w}^1, \mathbf{w}^2, \dots$ and $\mathbb{E}f(\mathbf{w}^1), \mathbb{E}f(\mathbf{w}^2), \dots$ and we use these values to verify our assumptions in Theorem 2. We set restart frequencies to be 20, 40, and 80, respectively. Figure 6 top panels plot k vs. the cardinality of the set $\mathcal{A} := \{k \in \mathbb{Z}^+ | \mathbb{E}f(\mathbf{w}^{k+1}) \geq \mathbb{E}f(\mathbf{w}^k)\}$, and Figure 6 bottom panels plot k vs. $\sum_{k \in \mathcal{A}} (\mathbb{E}f(\mathbf{w}^{k+1}) - \mathbb{E}f(\mathbf{w}^k))$. Figure 6 shows that $\sum_{k \in \mathcal{A}} (\mathbb{E}f(\mathbf{w}^{k+1}) - \mathbb{E}f(\mathbf{w}^k))$ converges to a constant $\bar{R} < +\infty$. We also noticed that when the training gets plateaued, $\mathbb{E}(f(\mathbf{w}^k))$ still oscillates, but the magnitude of the oscillation diminishes as iterations goes, which is consistent with our plots that the cardinality of \mathcal{A} increases linearly, but \bar{R} converges to a finite number. These numerical results show that our assumption in Theorem 2 is reasonable.

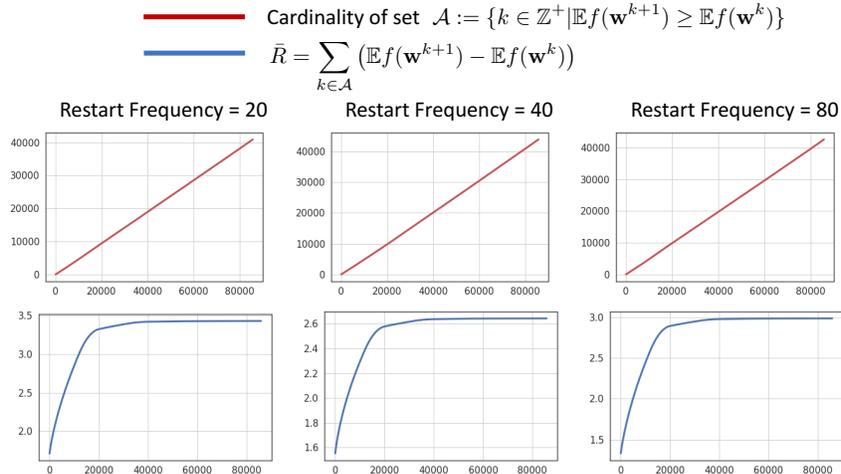


Figure 6: Cardinality of the set $\mathcal{A} := \{k \in \mathbb{Z}^+ | \mathbb{E}f(\mathbf{w}^{k+1}) \geq \mathbb{E}f(\mathbf{w}^k)\}$ (Top panels) and the value of $\bar{R} = \sum_{k \in \mathcal{A}} (\mathbb{E}f(\mathbf{w}^{k+1}) - \mathbb{E}f(\mathbf{w}^k))$ (Bottom panels). We notice that when the training gets plateaued, $\mathbb{E}(f(\mathbf{w}^k))$ still oscillates, but the magnitude of the oscillation diminishes as iterations goes, which is consistent with our plots that the cardinality of \mathcal{A} increases linearly, but \bar{R} converges to a finite number under different restart frequencies. These results confirm that our assumption in Theorem 2 is reasonable.

⁴We used the PyTorch implementation of LeNet at <https://github.com/activatedgeek/LeNet-5>.

D DATASETS AND IMPLEMENTATION DETAILS

D.1 CIFAR

The CIFAR10 and CIFAR100 datasets Krizhevsky et al. (2009) consist of 50K training images and 10K test images from 10 and 100 classes, respectively. Both training and test data are color images of size 32×32 . We run our CIFAR experiments on Pre-ResNet-110, 290, 470, 650, and 1001 with 5 different seeds He et al. (2016b). We train each model for 200 epochs with batch size of 128 and initial learning rate of 0.1, which is decayed by a factor of 10 at the 80th, 120th, and 160th epoch. The weight decay rate is 5×10^{-5} and the momentum for the SGD baseline is 0.9. Random cropping and random horizontal flipping are applied to training data. Our code is modified based on the Pytorch classification project Yang (2017),⁵ which was also used by Liu et al. Liu et al. (2020). We provide the restarting frequencies for the exponential and linear scheme for CIFAR10 and CIFAR100 in Table 7 below. Using the same notation as in the main text, we denote F_i as the restarting frequency at the i -th learning rate.

Table 7: Restarting frequencies for CIFAR10 and CIFAR100 experiments

	CIFAR10	CIFAR100
Linear schedule	$F_1 = 30, F_2 = 60, F_3 = 90, F_4 = 120$ ($r = 2$)	$F_1 = 50, F_2 = 100, F_3 = 150, F_4 = 200$ ($r = 2$)
Exponential schedule	$F_1 = 40, F_2 = 50, F_3 = 63, F_4 = 78$ ($r = 1.25$)	$F_1 = 45, F_2 = 68, F_3 = 101, F_4 = 152$ ($r = 1.50$)

D.2 IMAGENET

The ImageNet dataset contains roughly 1.28 million training color images and 50K validation color images from 1000 classes Russakovsky et al. (2015). We run our ImageNet experiments on ResNet-50, 101, 152, and 200 with 5 different seeds. Following He et al. (2016a;b), we train each model for 90 epochs with a batch size of 256 and decrease the learning rate by a factor of 10 at the 30th and 60th epoch. The initial learning rate is 0.1, the momentum is 0.9, and the weight decay rate is 1×10^{-5} . Random 224×224 cropping and random horizontal flipping are applied to training data. We use the official Pytorch ResNet implementation Paszke et al. (2019),⁶ and run our experiments on 8 Nvidia V100 GPUs. We report single-crop top-1 and top-5 errors of our models. In our experiments, we set $F_1 = 40$ at the 1st learning rate, $F_2 = 80$ at the 2nd learning rate, and F_3 is linearly decayed from 80 to 1 at the 3rd learning rate (see Table 8).

Table 8: Restarting frequencies for ImageNet experiments

	ImageNet
Linear schedule	$F_1 = 40, F_2 = 80, F_3$: linearly decayed from 80 to 1 in the last 30 epochs

D.3 TRAINING IMAGENET IN FEWER NUMBER OF EPOCHS:

Table 9 contains the learning rate and restarting frequency schedule for our experiments on training ImageNet in fewer number of epochs, i.e. the reported results in Table 6 in the main text. Other settings are the same as in the full-training ImageNet experiments described in Section D.2 above.

Additional Implementation Details: Implementation details for the ablation study of error rate vs. reduction in epochs and the ablation study of impact of restarting frequency are provided in Section F and G below.

D.4 DETAILS ON RESTARTING HYPER-PARAMETERS SEARCH

In our CIFAR10 and CIFAR100 experiments, for both linear and exponential schedule, we conduct hyperparameter searches over the restarting frequencies using our smallest model, Pre-ResNet-110,

⁵Implementation available at <https://github.com/bearpaw/pytorch-classification>

⁶Implementation available at <https://github.com/pytorch/examples/tree/master/imagenet>

Table 9: Learning rate and restarting frequency schedule for ImageNet short training, i.e. Table 6 in the main text.

	ImageNet
ResNet-50	Decrease the learning rate by a factor of 10 at the 30th and 56th epoch. Train for a total of 80 epochs. $F_1 = 60, F_2 = 105, F_3$: linearly decayed from 105 to 1 in the last 24 epochs
ResNet-101	Decrease the learning rate by a factor of 10 at the 30th and 56th epoch. Train for a total of 80 epochs. $F_1 = 40, F_2 = 80, F_3$: linearly decayed from 80 to 1 in the last 24 epochs
ResNet-152	Decrease the learning rate by a factor of 10 at the 30th and 51th epoch. Train for a total of 75 epochs. $F_1 = 40, F_2 = 80, F_3$: linearly decayed from 80 to 1 in the last 24 epochs
ResNet-200	Decrease the learning rate by a factor of 10 at the 30th and 46th epoch. Train for a total of 60 epochs. $F_1 = 40, F_2 = 80, F_3$: linearly decayed from 80 to 1 in the last 14 epochs

making choices based on final validation performance. The same chosen restarting frequencies are applied for all models including Pre-ResNet-110, 290, 470, 650, and 1001. In particular, we use 10,000 images from the original training set as a validation set. This validation set contains 1,000 and 100 images from each class for CIFAR10 and CIFAR100, respectively. We first train Pre-ResNet-110 on the remaining 40,000 training images and use the performance on the validation set averaged over 5 random seeds to select the initial restarting frequency F_1 and the growth rate r . Both F_1 and r are selected using grid search from the sets of $\{20, 25, 30, 35, 40, 45, 50\}$ and $\{1, 1.25, 1.5, 1.75, 2\}$, respectively. We then train all models including Pre-ResNet-110, 290, 470, 650, and 1001 on all 50,000 training images using the selected values of F_1 and r and report the results on the test set which contains 10,000 test images. The reported test performance is averaged over 5 random seeds. We also use the same selected values of F_1 and r for our short training experiments in Section 4.3.

For ImageNet experiments, we use linear scheduling and sweep over the initial restarting frequency F_1 and the growth rate r in the set of $\{20, 30, 40, 50, 60\}$ and $\{1, 1.25, 1.5, 1.75, 2\}$, respectively. We select the values of $F_1 = 40$ and $r = 2$ which have the highest final validation accuracy averaged over 5 random seeds. Same as in CIFAR10 and CIFAR100 experiments, we select F_1 and r using our smallest model, ResNet-50, and apply the same selected hyperparameter values for all models including ResNet-50, 101, 152, and 200. We also use the same selected values of F_1 and r for our short training experiments in Section 4.3. However, for ResNet-50, we observe that $F_1 = 60$ and $r = 1.75$ yields better performance in short training. All reported results are averaged over 5 random seeds.

E SRSGD VS. SGD AND SGD + NM ON IMAGENET CLASSIFICATION AND OTHER TASKS

E.1 COMPARING WITH SGD WITH NESTEROV MOMENTUM ON IMAGENET CLASSIFICATION

In this section, we compare SRSGD with SGD with Nesterov constant momentum (SGD + NM) in training ResNets for ImageNet classification. All hyper-parameters of SGD with constant Nesterov momentum used in our experiments are the same as those of SGD described in section D.2. We list the results in Table 10. Again, SRSGD remarkably outperforms SGD + NM in training ResNets for ImageNet classification, and as the network goes deeper the improvement becomes more significant.

E.2 LONG SHORT-TERM MEMORY (LSTM) TRAINING FOR PIXEL-BY-PIXEL MNIST

In this task, we examine the advantage of SRSGD over SGD and SGD with Nesterov Momentum in training recurrent neural networks. In our experiments, we use an LSTM with different numbers of hidden units (128, 256, and 512) to classify samples from the well-known MNIST dataset LeCun & Cortes (2010). We follow the implementation of Le et al. (2015) and feed each pixel of the image into the RNN sequentially. In addition, we choose a random permutation of $28 \times 28 = 784$ elements at the beginning of the experiment. This fixed permutation is applied to training and testing sequences. This task is known as permuted MNIST classification, which has become standard to measure the performance of RNNs and their ability to capture long term dependencies.

Table 10: Single crop validation errors (%) on ImageNet of ResNets trained with SGD + NM and SRSGD. We report the results of SRSGD with the increasing restarting frequency in the first two learning rates. In the last learning rate, the restarting frequency is linearly decreased from 70 to 1. For baseline results, we also include the reported single-crop validation errors He et al. (2016c) (in parentheses).

Network	# Params	SGD + NM		SRSGD		Improvement	
		top-1	top-5	top-1	top-5	top-1	top-5
ResNet-50	25.56M	24.27 ± 0.07	7.17 ± 0.07	23.85 ± 0.09	7.10 ± 0.09	0.42	0.07
ResNet-101	44.55M	22.32 ± 0.05	6.18 ± 0.05	22.06 ± 0.10	6.09 ± 0.07	0.26	0.09
ResNet-152	60.19M	21.77 ± 0.14	5.86 ± 0.09	21.46 ± 0.07	5.69 ± 0.03	0.31	0.17
ResNet-200	64.67M	21.98 ± 0.22	5.99 ± 0.20	20.93 ± 0.13	5.57 ± 0.05	1.05	0.42

Implementation and Training Details: For the LSTM model, we initialize the forget bias to 1 and other biases to 0. All weights matrices are initialized orthogonally except for the hidden-to-hidden weight matrices, which are initialized to be identity matrices. We train each model for 350 epochs with the initial learning rate of 0.01. The learning rate was reduced by a factor of 10 at epoch 200 and 300. The momentum is set to 0.9 for SGD with standard and Nesterov constant momentum. The restart schedule for SRSGD is set to 90, 30, 90. The restart schedule changes at epoch 200 and 300. In all experiments, we use batch size 128 and the gradients are clipped so that their L2 norm are at most 1. Our code is based on the code from the exponential RNN’s Github.⁷

Results: Our experiments corroborate the superiority of SRSGD over the two baselines. SRSGD yields much smaller test error and converges faster than SGD with standard and Nesterov constant momentum across all settings with different number of LSTM hidden units. We summarize our results in Table 11 and Figure 7.

Table 11: Test errors (%) on Permuted MNIST of trained with SGD, SGD + NM and SRSGD. The LSTM model has 128 hidden units. In all experiments, we use the initial learning rate of 0.01, which is reduced by a factor of 10 at epoch 200 and 300. All models are trained for 350 epochs. The momentum for SGD and SGD + NM is set to 0.9. The restart schedule in SRSGD is set to 90, 30, and 90.

Network	No. Hidden Units	SGD	SGD + NM	SRSGD	Improvement over SGD/SGD + NM
LSTM	128	10.10 ± 0.57	9.75 ± 0.69	8.61 ± 0.30	1.49/1.14
LSTM	256	10.42 ± 0.63	10.09 ± 0.61	9.03 ± 0.23	1.39/1.06
LSTM	512	10.04 ± 0.35	9.55 ± 1.09	8.49 ± 1.59	1.55/1.06

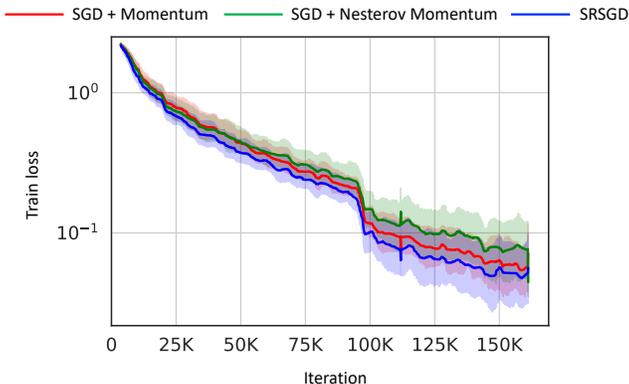


Figure 7: Training loss vs. training iterations of LSTM trained with SGD (red), SGD + NM (green), and SRSGD (blue) for PMNIST classification tasks.

E.3 WASSERSTEIN GENERATIVE ADVERSARIAL NETWORKS (WGAN) TRAINING ON MNIST

We investigate the advantage of SRSGD over SGD with standard and Nesterov momentum in training deep generative models. In our experiments, we train a WGAN with gradient penalty Gulrajani

⁷Implementation available at <https://github.com/Lezcano/expRNN>

et al. (2017) on MNIST. We evaluate our models using the discriminator’s loss, i.e. the Earth Moving distance estimate, since in WGAN lower discriminator loss and better sample quality are correlated Arjovsky et al. (2017).

Implementation and Training Details: The detailed implementations of our generator and discriminator are given below. For the generator, we set `latent_dim` to 100 and `d` to 32. For the discriminator, we set `d` to 32. We train each model for 350 epochs with the initial learning rate of 0.01. The learning rate was reduced by a factor of 10 at epoch 200 and 300. The momentum is set to 0.9 for SGD with standard and Nesterov constant momentum. The restart schedule for SRSGD is set to 60, 120, 180. The restart schedule changes at epoch 200 and 300. In all experiments, we use batch size 64. Our code is based on the code from the Pytorch WGAN-GP Github.⁸

```
import torch
import torch.nn as nn

class Generator(nn.Module):
    def __init__(self, latent_dim, d=32):
        super().__init__()
        self.net = nn.Sequential(
            nn.ConvTranspose2d(latent_dim, d * 8, 4, 1, 0),
            nn.BatchNorm2d(d * 8),
            nn.ReLU(True),

            nn.ConvTranspose2d(d * 8, d * 4, 4, 2, 1),
            nn.BatchNorm2d(d * 4),
            nn.ReLU(True),

            nn.ConvTranspose2d(d * 4, d * 2, 4, 2, 1),
            nn.BatchNorm2d(d * 2),
            nn.ReLU(True),

            nn.ConvTranspose2d(d * 2, 1, 4, 2, 1),
            nn.Tanh()
        )
    def forward(self, x):
        return self.net(x)

class Discriminator(nn.Module):
    def __init__(self, d=32):
        super().__init__()
        self.net = nn.Sequential(
            nn.Conv2d(1, d, 4, 2, 1),
            nn.InstanceNorm2d(d),
            nn.LeakyReLU(0.2),

            nn.Conv2d(d, d * 2, 4, 2, 1),
            nn.InstanceNorm2d(d * 2),
            nn.LeakyReLU(0.2),

            nn.Conv2d(d * 2, d * 4, 4, 2, 1),
            nn.InstanceNorm2d(d * 4),
            nn.LeakyReLU(0.2),

            nn.Conv2d(d * 4, 1, 4, 1, 0),
        )
    def forward(self, x):
        outputs = self.net(x)
        return outputs.squeeze()
```

Results: Our SRSGD is still better than both the baselines. SRSGD achieves smaller discriminator loss, i.e. Earth Moving distance estimate, and converges faster than SGD with standard and Nesterov constant momentum. We summarize our results in Table 12 and Figure 8. We also demonstrate the

⁸Implementation available at <https://github.com/arturml/pytorch-wgan-gp>

digits generated by the trained WGAN in Figure 9. By visually evaluation, we observe that samples generated by the WGAN trained with SRSGD look slightly better than those generated by the WGAN trained with SGD with standard and Nesterov constant momentum.

Table 12: Discriminator loss (i.e. Earth Moving distance estimate) of the WGAN with gradient penalty trained on MNIST with SGD, SGD + NM and SRSGD. In all experiments, we use the initial learning rate of 0.01, which is reduced by a factor of 10 at epoch 200 and 300. All models are trained for 350 epochs. The momentum for SGD and SGD + NM is set to 0.9. The restart schedule in SRSGD is set to 60, 120, and 180.

Task	SGD	SGD + NM	SRSGD	Improvement over SGD/SGD + NM
MNIST	0.71 ± 0.10	0.58 ± 0.03	0.44 ± 0.06	0.27/0.14

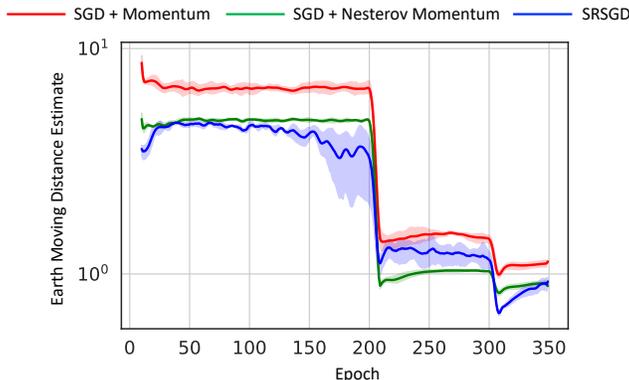


Figure 8: Earth Moving distance estimate (i.e. discriminator loss) vs. training epochs of WGAN with gradient penalty trained with SGD (red), SGD + NM (green), and SRSGD (blue) on MNIST.

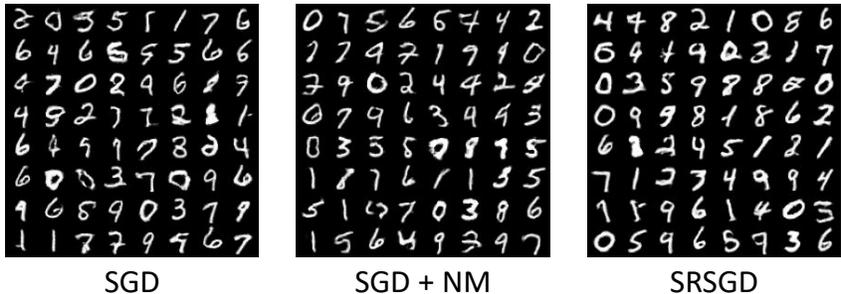


Figure 9: MNIST digits generated by WGAN trained with gradient penalty by SGD (left), SGD + NM (middle), and SRSGD (right).

F ERROR RATE VS. REDUCTION IN TRAINING EPOCHS

F.1 IMPLEMENTATION DETAILS

CIFAR10 (Figure 4, left, in the main text) and CIFAR100 (Figure 11 in this Appendix): Except for learning rate schedule, we use the same setting described in Section D.1 above and Section 4.1 in the main text. Table 13 contains the learning rate schedule for each number of epoch reduction in Figure 4 (left) in the main text and Figure 11 below.

ImageNet (Figure 4, right, in the main text): Except for the total number of training epochs, other settings are similar to experiments for training ImageNet in fewer number of epochs described in Section D.3. In particular, the learning rate and restarting frequency schedule still follow those in Table 9 above. We examine different numbers of training epochs: 90 (0 epoch reduction), 80 (10 epochs reduction), 75 (15 epochs reduction), 70 (20 epochs reduction), 65 (25 epochs reduction), and 60 (30 epochs reduction).

Table 13: Learning rate (LR) schedule for the ablation study of error rate vs. reduction in training epochs for CIFAR10 experiments, i.e. Figure 4 in the main text and for CIFAR100 experiments, i.e. Figure 11 in this Appendix.

#of Epoch Reduction	LR Schedule
0	Decrease the LR by a factor of 10 at the 80th, 120th and 160th epoch. Train for a total of 200 epochs.
15	Decrease the LR by a factor of 10 at the 80th, 115th and 150th epoch. Train for a total of 185 epochs.
30	Decrease the LR by a factor of 10 at the 80th, 110th and 140th epoch. Train for a total of 170 epochs.
45	Decrease the LR by a factor of 10 at the 80th, 105th and 130th epoch. Train for a total of 155 epochs.
60	Decrease the LR by a factor of 10 at the 80th, 100th and 120th epoch. Train for a total of 140 epochs.
75	Decrease the LR by a factor of 10 at the 80th, 95th and 110th epoch. Train for a total of 125 epochs.
90	Decrease the LR by a factor of 10 at the 80th, 90th and 100th epoch. Train for a total of 110 epochs.
100	Decrease the LR by a factor of 10 at the 80th, 90th and 95th epoch. Train for a total of 100 epochs.

Table 14: Classification test error (%) of SGD short training (100 epochs), SGD full training (200 epochs), SRSGD short training (100 epochs), and SRSGD full training (200 epochs) on CIFAR10. SGD short training yields much worse test errors than the others while SRSGD short training yields either comparable or even better results than SGD full training.

Network	SGD short training	SGD full training	SRSGD short training	SRSGD full training
Pre-ResNet-110	6.36 \pm 0.12	5.25 \pm 0.14	5.43 \pm 0.18	4.93 \pm 0.13
Pre-ResNet-290	5.81 \pm 0.10	5.05 \pm 0.23	4.83 \pm 0.11	4.37 \pm 0.15
Pre-ResNet-470	5.59 \pm 0.19	4.92 \pm 0.10	4.64 \pm 0.17	4.18 \pm 0.09

Table 15: Classification test error (%) of SGD short training (100 epochs), SGD full training (200 epochs), SRSGD short training (100 epochs), and SRSGD full training (200 epochs) on CIFAR100. SGD short training yields worse test errors than the others while SRSGD short training yields either comparable or even better results than SGD full training.

Network	SGD short training	SGD full training	SRSGD short training	SRSGD full training
Pre-ResNet-110	24.34 \pm 0.21	23.75 \pm 0.20	23.85 \pm 0.19	23.49 \pm 0.23
Pre-ResNet-290	22.70 \pm 0.25	21.78 \pm 0.21	21.77 \pm 0.43	21.49 \pm 0.27
Pre-ResNet-470	22.43 \pm 0.18	21.43 \pm 0.30	21.42 \pm 0.19	20.71 \pm 0.32

F.2 SHORT TRAINING ON CIFAR10/CIFAR100 USING SGD

For better comparison between SRSGD training using fewer epochs and SGD full training, we also conduct experiments with SGD training using fewer epochs on CIFAR10 and CIFAR100. Table 14 and 15 compares SRSGD short training using 100 epoch, SGD short training using 100 epochs, SRSGD full training using 200 epochs, and SGD full training using 200 epochs for Pre-ResNet-110, 290, and 470 on CIFAR10 and CIFAR100, respectively. The learning rate schedule for SGD short training using 100 epochs is the same as the learning rate schedule for SRSGD short training using 100 epoch given in Section 4 and in Table 13 above. In particular, for both SGD and SRSGD training using 100 epochs, we decrease the learning rate by a factor of 10 at the 80th, 90th, and 95th epoch. We observe that SGD short training has the worst performance compared to the others while SRSGD short training yields either comparable or even better results than SGD full training.

F.3 ADDITIONAL EXPERIMENTAL RESULTS

Figure 10 shows error rate vs. reduction in epochs for all models trained on CIFAR10 and ImageNet. It is a more complete version of Figure 4 in the main text. Table 16 and Table 17 provide detailed test errors vs. number of training epoch reduction reported in Figure 4 and Figure 10. We also conduct an additional ablation study of error rate vs. reduction in epochs for CIFAR100 and include the results in Figure 11 and Table 18 below.

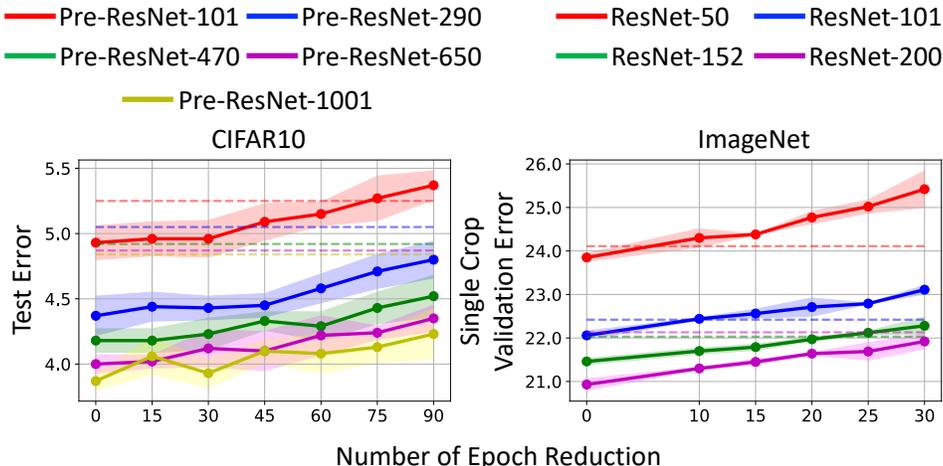


Figure 10: Test error vs. number of training epochs. Dashed lines are test errors of SGD trained in 200 epochs for CIFAR10 and 90 epochs for ImageNet. For CIFAR, SRS GD with fewer epochs achieves comparable results to SRS GD with 200 epochs. For ImageNet, training with less epochs slightly decreases the performance of SRS GD but still achieves comparable results to 200-epoch SGD.

Table 16: Test error vs. number of training epochs for CIFAR10

Network	110 (90 less)	125 (75 less)	140 (60 less)	155 (45 less)	170 (30 less)	185 (15 less)	200 (full trainings)
Pre-ResNet-110	5.37 \pm 0.11	5.27 \pm 0.17	5.15 \pm 0.09	5.09 \pm 0.14	4.96 \pm 0.14	4.96 \pm 0.13	4.93 \pm 0.13
Pre-ResNet-290	4.80 \pm 0.14	4.71 \pm 0.13	4.58 \pm 0.11	4.45 \pm 0.09	4.43 \pm 0.09	4.44 \pm 0.11	4.37 \pm 0.15
Pre-ResNet-470	4.52 \pm 0.16	4.43 \pm 0.12	4.29 \pm 0.11	4.33 \pm 0.07	4.23 \pm 0.12	4.18 \pm 0.09	4.18 \pm 0.09
Pre-ResNet-650	4.35 \pm 0.10	4.24 \pm 0.05	4.22 \pm 0.15	4.10 \pm 0.15	4.12 \pm 0.14	4.02 \pm 0.05	4.00 \pm 0.07
Pre-ResNet-1001	4.23 \pm 0.19	4.13 \pm 0.12	4.08 \pm 0.15	4.10 \pm 0.09	3.93 \pm 0.11	4.06 \pm 0.14	3.87 \pm 0.07

Table 17: Top 1 single crop validation error vs. number of training epochs for ImageNet

Network	60 (30 less)	65 (25 less)	70 (20 less)	75 (15 less)	80 (10 less)	90 (full trainings)
ResNet-50	25.42 \pm 0.42	25.02 \pm 0.15	24.77 \pm 0.14	24.38 \pm 0.01	24.30 \pm 0.21	23.85 \pm 0.09
ResNet-101	23.11 \pm 0.10	22.79 \pm 0.01	22.71 \pm 0.21	22.56 \pm 0.10	22.44 \pm 0.03	22.06 \pm 0.10
ResNet-152	22.28 \pm 0.20	22.12 \pm 0.04	21.97 \pm 0.04	21.79 \pm 0.07	21.70 \pm 0.07	21.46 \pm 0.07
ResNet-200	21.92 \pm 0.17	21.69 \pm 0.20	21.64 \pm 0.03	21.45 \pm 0.06	21.30 \pm 0.03	20.93 \pm 0.13

Table 18: Test error vs. number of training epochs for CIFAR100

Network	110 (90 less)	125 (75 less)	140 (60 less)	155 (45 less)	170 (30 less)	185 (15 less)	200 (full trainings)
Pre-ResNet-110	24.06 \pm 0.26	23.82 \pm 0.24	23.82 \pm 0.28	23.58 \pm 0.18	23.69 \pm 0.21	23.73 \pm 0.34	23.49 \pm 0.23
Pre-ResNet-290	21.96 \pm 0.45	21.77 \pm 0.21	21.67 \pm 0.37	21.56 \pm 0.33	21.38 \pm 0.44	21.47 \pm 0.32	21.49 \pm 0.27
Pre-ResNet-470	21.35 \pm 0.17	21.25 \pm 0.17	21.21 \pm 0.18	21.09 \pm 0.28	20.87 \pm 0.28	20.81 \pm 0.32	20.71 \pm 0.32
Pre-ResNet-650	21.18 \pm 0.27	20.95 \pm 0.13	20.77 \pm 0.31	20.61 \pm 0.19	20.57 \pm 0.13	20.47 \pm 0.07	20.36 \pm 0.25
Pre-ResNet-1001	20.27 \pm 0.17	20.03 \pm 0.13	20.05 \pm 0.22	19.74 \pm 0.18	19.71 \pm 0.22	19.67 \pm 0.22	19.75 \pm 0.11

G IMPACT OF RESTARTING FREQUENCY FOR IMAGENET AND CIFAR100

G.1 IMPLEMENTATION DETAILS

For the CIFAR10 experiments on Pre-ResNet-290 in Figure 5 in the main text, as well as the CIFAR100 and ImageNet experiments in Figure 14 and 15 in this Appendix, we vary the initial restarting frequency F_1 . Other settings are the same as described in Section D above.

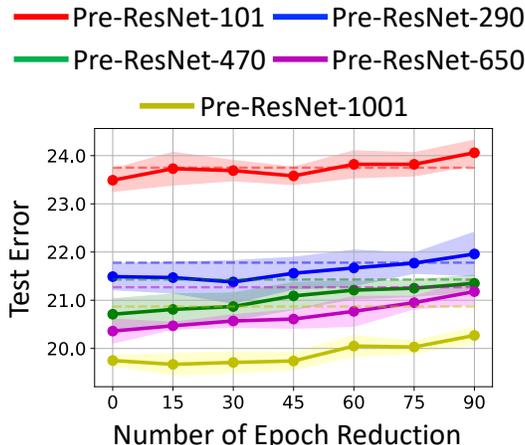


Figure 11: Test error vs. number of epoch reduction in CIFAR100 training. The dashed lines are test errors of the SGD baseline. For CIFAR100, SRSGD training with fewer epochs can achieve comparable results to SRSGD training with full 200 epochs. In some cases, such as with Pre-ResNet-290 and 1001, SRSGD training with fewer epochs achieves even better results than SRSGD training with full 200 epochs.

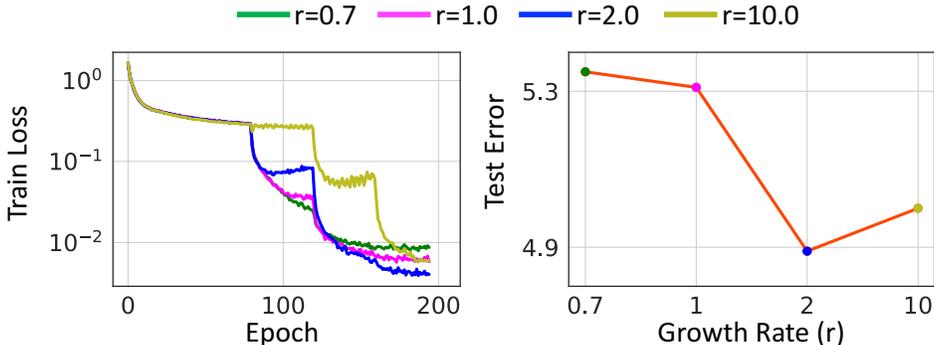


Figure 12: Training loss (left) and test error (right) of Pre-ResNet-110 trained on CIFAR10 with different growth rate r (linear schedule). Here, we fix the initial restarting frequency $F_1 = 30$ for all trainings. Increasing the restarting frequency during training yields better results than decreasing the restarting frequency, but increasing the restarting frequency too fast and too much also diminishes the performance of SRSGD.

G.2 IMPACT OF THE GROWTH RATE r

We do an ablation study for the growth rate r to understand its impact on the behavior of SRSGD. We choose a case study of training a Pre-ResNet-110 on CIFAR10 using SRSGD with a linear schedule scheme for the restarting frequency. We fix the initial restarting frequency $F_1 = 30$ and vary the growth rate r . We choose r from the set of $\{0.7, 1.0, 2.0, 10.0\}$. These values of r represent four different scenarios. When $r = 0.7$, the restarting frequency decreases every time the learning rate is reduced by a factor of 10. When $r = 1.0$, the restarting frequency stays constant during the training. When $r = 2.0$, the restarting frequency increases every time the learning rate is reduced by a factor of 10. Finally, when $r = 10.0$, it is similar to when $r = 2.0$, but the restarting frequency increases much faster and to larger values. Figure 12 summarizes the results of our ablation study. We observe that for CIFAR10, decreasing the restarting frequency or keeping it constant during training yields worse results than increasing the restarting frequency. However, increasing the restarting frequency too much also diminishes the performance of SRSGD.

G.3 ADDITIONAL EXPERIMENTAL RESULTS

To complete our study on the impact of restarting frequency in Section 5.2 in the main text, we examine the case of CIFAR100 and ImageNet in this section. We summarize our results in Figure 14 and 15 below. Also, Figure 13 is a more detailed version of Figure 5 in the main text.

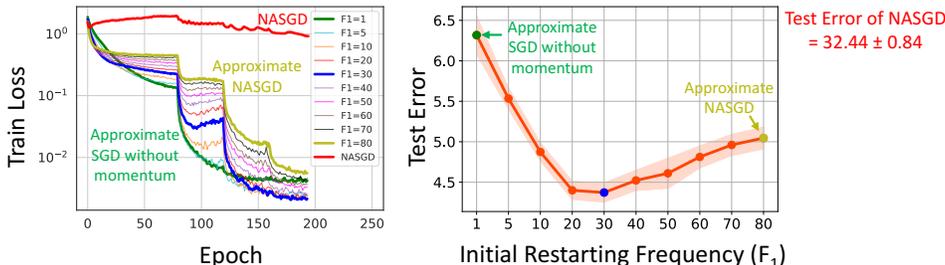


Figure 13: Training loss (left) and test error (right) of Pre-ResNet-290 trained on CIFAR10 with different initial restarting frequencies F_1 (linear schedule). SRS GD with small F_1 approximates SGD without momentum, while SRS GD with large F_1 approximates NASGD. The training loss curve and test accuracy of NASGD are shown in red and confirm the result of Theorem 1 that NASGD accumulates error due to the stochastic gradients.

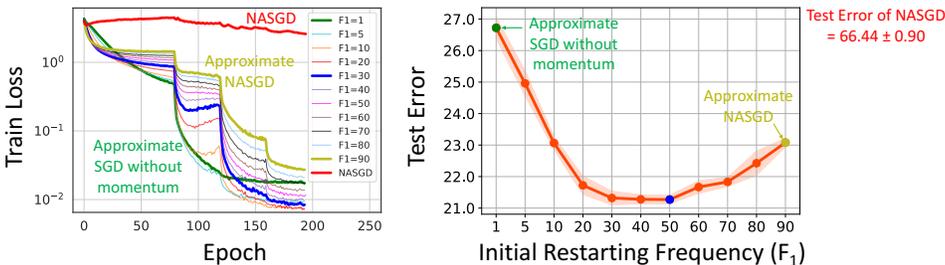


Figure 14: Training loss and test error of Pre-ResNet-290 trained on CIFAR100 with different initial restarting frequencies F_1 (linear schedule). SRS GD with small F_1 approximates SGD without momentum, while SRS GD with large F_1 approximates NASGD. The training loss curve and test accuracy of NASGD are shown in red and confirm the result of Theorem 1 that NASGD accumulates error due to the stochastic gradients.

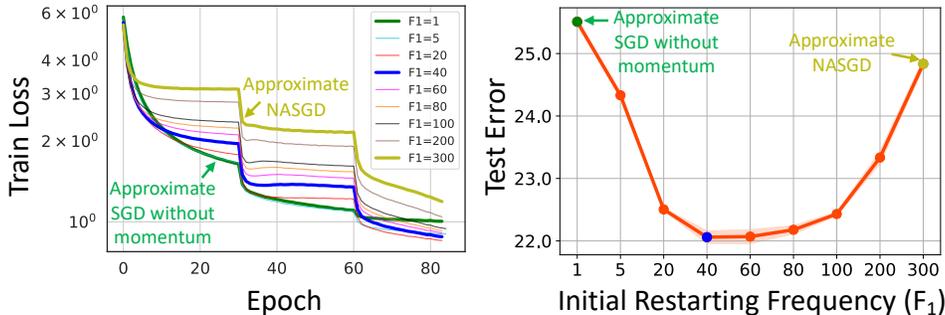


Figure 15: Training loss and test error of ResNet-101 trained on ImageNet with different initial restarting frequencies F_1 . We use linear schedule and linearly decrease the restarting frequency to 1 at the last learning rate. SRS GD with small F_1 approximates SGD without momentum, while SRS GD with large F_1 approximates NASGD.

H FULL TRAINING WITH LESS EPOCHS AT THE INTERMEDIATE LEARNING RATES

We explore SRSGD full training (200 epochs on CIFAR and 90 epochs on ImageNet) with less number of epochs at the intermediate learning rates and report the results in Table 19, 20, 21 and Figure 16, 17, 18 below. The settings and implementation details here are similar to those in Section F, but using all 200 epochs for CIFAR experiments and 90 epochs for ImageNet experiments.

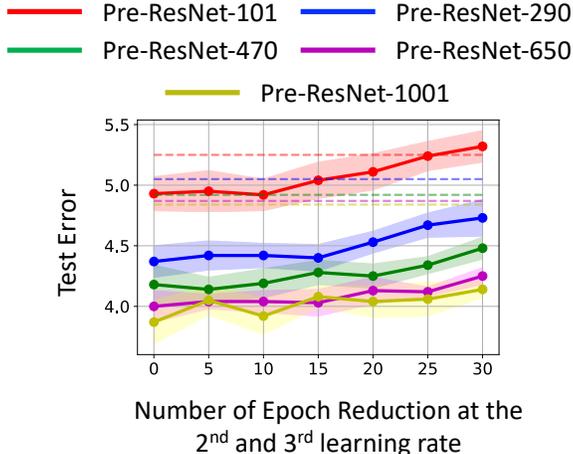


Figure 16: Test error when using new learning rate schedules with less training epochs at the 2nd and 3rd learning rate for CIFAR10. We still train in full 200 epochs in this experiment. On the x-axis, 10, for example, means we reduce the number of training epochs by 10 at each intermediate learning rate, i.e. the 2nd and 3rd learning rate. The dashed lines are test errors of the SGD baseline.

Table 19: Test error when using new learning rate schedules with less training epochs at the 2nd and 3rd learning rate for CIFAR10. We still train in full 200 epochs in this experiment. In the table, 80-90-100, for example, means we reduce the learning rate by factor of 10 at the 80th, 90th, and 100th epoch.

Network	80 - 90 - 100	80 - 95 - 110	80 - 100 - 120	80 - 105 - 130	80 - 110 - 140	80 - 115 - 150	80 - 120 - 160
Pre-ResNet-110	5.32 ± 0.14	5.24 ± 0.17	5.11 ± 0.13	5.04 ± 0.15	4.92 ± 0.15	4.95 ± 0.12	4.93 ± 0.13
Pre-ResNet-290	4.73 ± 0.13	4.67 ± 0.12	4.53 ± 0.10	4.40 ± 0.11	4.42 ± 0.09	4.42 ± 0.10	4.37 ± 0.15
Pre-ResNet-470	4.48 ± 0.16	4.34 ± 0.10	4.25 ± 0.12	4.28 ± 0.10	4.19 ± 0.10	4.14 ± 0.07	4.18 ± 0.09
Pre-ResNet-650	4.25 ± 0.13	4.12 ± 0.06	4.13 ± 0.09	4.03 ± 0.11	4.04 ± 0.11	4.04 ± 0.04	4.00 ± 0.07
Pre-ResNet-1001	4.14 ± 0.18	4.06 ± 0.12	4.04 ± 0.15	4.08 ± 0.09	3.92 ± 0.13	4.05 ± 0.14	3.87 ± 0.07

Table 20: Test error when using new learning rate schedules with less training epochs at the 2nd and 3rd learning rate for CIFAR100. We still train in full 200 epochs in this experiment. In the table, 80-90-100, for example, means we reduce the learning rate by factor of 10 at the 80th, 90th, and 100th epoch.

Network	80 - 90 - 100	80 - 95 - 110	80 - 100 - 120	80 - 105 - 130	80 - 110 - 140	80 - 115 - 150	80 - 120 - 160
Pre-ResNet-110	23.65 ± 0.14	23.96 ± 0.26	23.97 ± 0.31	23.53 ± 0.13	23.57 ± 0.36	23.68 ± 0.24	23.49 ± 0.23
Pre-ResNet-290	21.94 ± 0.44	21.71 ± 0.27	21.55 ± 0.40	21.44 ± 0.31	21.37 ± 0.45	21.47 ± 0.32	21.49 ± 0.27
Pre-ResNet-470	21.29 ± 0.11	21.21 ± 0.14	21.17 ± 0.18	20.99 ± 0.28	20.81 ± 0.22	20.80 ± 0.31	20.71 ± 0.32
Pre-ResNet-650	21.11 ± 0.24	20.91 ± 0.17	20.66 ± 0.33	20.52 ± 0.18	20.51 ± 0.16	20.43 ± 0.10	20.36 ± 0.25
Pre-ResNet-1001	20.21 ± 0.15	20.00 ± 0.11	19.86 ± 0.19	19.55 ± 0.19	19.69 ± 0.21	19.60 ± 0.17	19.75 ± 0.11

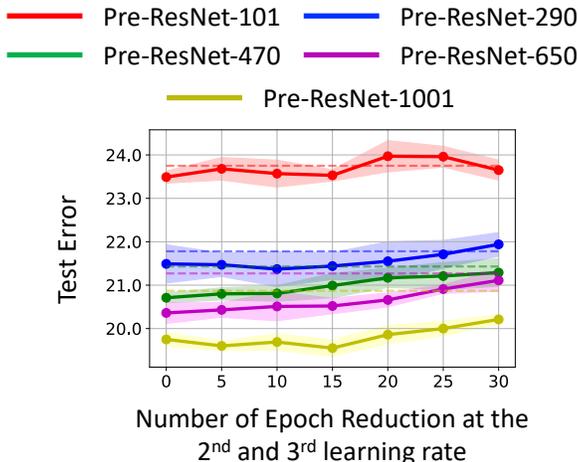


Figure 17: Test error when using new learning rate schedules with less training epochs at the 2nd and 3rd learning rate for CIFAR100. We still train in full 200 epochs in this experiment. On the x-axis, 10, for example, means we reduce the number of training epochs by 10 at each intermediate learning rate, i.e. the 2nd and 3rd learning rate. The dashed lines are test errors of the SGD baseline.

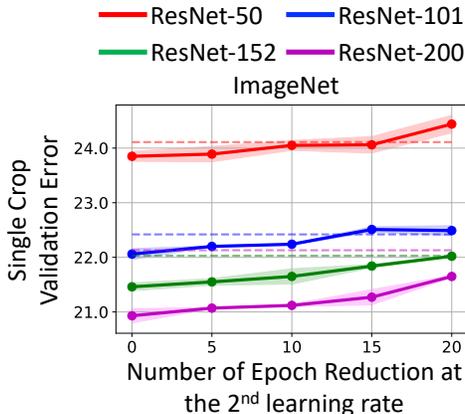


Figure 18: Test error when using new learning rate schedules with less training epochs at the 2nd learning rate for ImageNet. We still train in full 90 epochs in this experiment. On the x-axis, 10, for example, means we reduce the number of training epochs by 10 at the 2nd learning rate. The dashed lines are test errors of the SGD baseline.

Table 21: Top 1 single crop validation error when using new learning rate schedules with less training epochs at the 2nd learning rate for ImageNet. We still train in full 90 epochs in this experiment. In the table, 30-40, for example, means we reduce the learning rate by factor of 10 at the 30th and 40th epoch.

Network	30 - 40	30 - 45	30 - 50	30 - 55	30 - 60
ResNet-50	24.44 ± 0.16	24.06 ± 0.15	24.05 ± 0.09	23.89 ± 0.14	23.85 ± 0.09
ResNet-101	22.49 ± 0.09	22.51 ± 0.05	22.24 ± 0.01	22.20 ± 0.01	22.06 ± 0.10
ResNet-152	22.02 ± 0.01	21.84 ± 0.03	21.65 ± 0.14	21.55 ± 0.06	21.46 ± 0.07
ResNet-200	21.65 ± 0.02	21.27 ± 0.14	21.12 ± 0.02	21.07 ± 0.01	20.93 ± 0.13

I VISUALIZATION OF SRSGD’S TRAJECTORY

Here we visualize the training trajectory through bad minima of SRSGD, SGD with constant momentum, and SGD. In particular, we train a neural net classifier on a swiss roll data as in Huang et al. (2019) and find bad minima along its training. Each red dot in Figure 19 represents the trained model after each 10 epochs in the training. From each red dot, we search for nearby bad local minima,

which are the blue dots. Those bad local minima achieve good training error but bad test error. We plots the trained models and bad local minima using PCA Wold et al. (1987) and t-SNE Maaten & Hinton (2008) embedding. The blue color bar is for the test accuracy of bad local minima; the red color bar is for the number of training epochs.

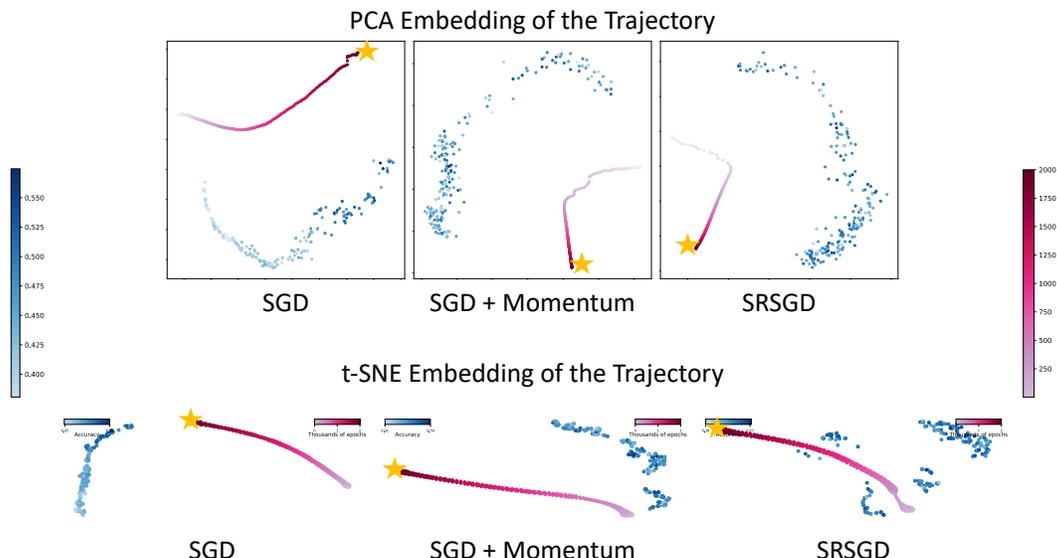


Figure 19: Trajectory through bad minima of SGD, SGD with constant momentum, and SRSGD during the training: we train a neural net classifier and plot the iterates of SGD after each ten epoch (red dots). We also plot locations of nearby “bad” minima with poor generalization (blue dots). We visualize these using PCA and t-SNE embedding. The blue color bar is for the test accuracy of bad local minima while the red color bar is for the number of training epochs. All blue dots for SGD with constant momentum and SRSGD achieve near perfect train accuracy, but with test accuracy below 59%. All blue dots for SGD achieves average train accuracy of 73.11% and with test accuracy also below 59%. The final iterate (yellow star) of SGD, SGD with constant momentum, and SRSGD achieve 73.13%, 99.25%, and 100.0% test accuracy, respectively.

(CONTINUED NEXT PAGE)

J SRSGD IMPLEMENTATION IN PYTORCH

```

import torch
from .optimizer import Optimizer, required

class SRSGD(Optimizer):
    """
    Scheduled Restart SGD.
    Args:
        params (iterable): iterable of parameters to optimize
            or dicts defining parameter groups.
        lr (float): learning rate.
        weight_decay (float, optional): weight decay (L2 penalty) (
            default: 0)
        iter_count (integer): count the iterations mod 200
    Example:
        >>> optimizer = torch.optim.SRSGD(model.parameters(), lr=0.1,
            weight_decay=5e-4, iter_count=1)
        >>> optimizer.zero_grad()
        >>> loss_fn(model(input), target).backward()
        >>> optimizer.step()
        >>> iter_count = optimizer.update_iter()
    Formula:
         $v_{t+1} = p_t - lr * g_t$ 
         $p_{t+1} = v_{t+1} + (iter\_count)/(iter\_count+3)*(v_{t+1} - v_t)$ 
    """
    def __init__(self, params, lr=required, weight_decay=0.,
                 iter_count=1, restarting_iter=100):
        if lr is not required and lr < 0.0:
            raise ValueError("Invalid learning rate: {}".format(lr))
        if weight_decay < 0.0:
            raise ValueError("Invalid weight decay value: {}".format(
                weight_decay))
        if iter_count < 1:
            raise ValueError("Invalid iter count: {}".format(iter_count))
        if restarting_iter < 1:
            raise ValueError("Invalid iter total: {}".format(
                restarting_iter))

        defaults = dict(lr=lr, weight_decay=weight_decay, iter_count=
            iter_count,
                        restarting_iter=restarting_iter)
        super(SRSGD, self).__init__(params, defaults)

    def __setstate__(self, state):
        super(SRSGD, self).__setstate__(state)

    def update_iter(self):
        idx = 1
        for group in self.param_groups:
            if idx == 1:
                group['iter_count'] += 1
                if group['iter_count'] >= group['restarting_iter']:
                    group['iter_count'] = 1
            idx += 1
        return group['iter_count'], group['restarting_iter']

    def step(self, closure=None):
        """
        Perform a single optimization step.
        Arguments: closure (callable, optional): A closure that
            reevaluates the model and returns the loss.
        """
        loss = None
        if closure is not None:

```

```

        loss = closure()

    for group in self.param_groups:
        weight_decay = group['weight_decay']
        momentum = (group['iter_count'] - 1.) / (group['iter_count'] + 2.)
        for p in group['params']:
            if p.grad is None:
                continue
            d_p = p.grad.data
            if weight_decay != 0:
                d_p.add_(-weight_decay, p.data)

            param_state = self.state[p]

            if 'momentum_buffer' not in param_state:
                buf0 = param_state['momentum_buffer'] = torch.clone(p.data).detach()
            else:
                buf0 = param_state['momentum_buffer']

            buf1 = p.data - group['lr'] * d_p
            p.data = buf1 + momentum * (buf1 - buf0)
            param_state['momentum_buffer'] = buf1

    iter_count, iter_total = self.update_iter()

    return loss

```

K SRSGD IMPLEMENTATION IN KERAS

```

import numpy as np
import tensorflow as tf
from keras import backend as K
from keras.optimizers import Optimizer
from keras.legacy import interfaces
if K.backend() == 'tensorflow':
    import tensorflow as tf

class SRSGD(Optimizer):
    """Scheduled Restart Stochastic gradient descent optimizer.
    Includes support for Nesterov momentum
    and learning rate decay.
    # Arguments
        learning_rate: float >= 0. Learning rate.
    """

    def __init__(self, learning_rate=0.01, iter_count=1, restarting_iter=40, **kwargs):
        learning_rate = kwargs.pop('lr', learning_rate)
        self.initial_decay = kwargs.pop('decay', 0.0)
        super(SRSGD, self).__init__(**kwargs)
        with K.name_scope(self.__class__.__name__):
            self.iterations = K.variable(0, dtype='int64', name='iterations')
            self.learning_rate = K.variable(learning_rate, name='learning_rate')
            self.decay = K.variable(self.initial_decay, name='decay')
            # for srsgd
            self.iter_count = K.variable(iter_count, dtype='int64', name='iter_count')
            self.restarting_iter = K.variable(restarting_iter, dtype='int64', name='restarting_iter')
        self.nesterov = nesterov

```

```

@interfaces.legacy_get_updates_support
@K.symbolic
def get_updates(self, loss, params):
    grads = self.get_gradients(loss, params)
    self.updates = [K.update_add(self.iterations, 1)]

    momentum = (K.cast(self.iter_count, dtype=K.dtype(self.decay)) -
                 1.) / (K.cast(self.iter_count, dtype=K.dtype(self.decay)) + 2.)

    lr = self.learning_rate
    if self.initial_decay > 0:
        lr = lr * (1. / (1. + self.decay * K.cast(self.iterations,
                                                  K.dtype(self.decay)
                                                  )))

    # momentum
    shapes = [K.int_shape(p) for p in params]

    moments = [K.variable(value=K.get_value(p), dtype=K.dtype(self.
                                                                decay), name='moment_' + str(i))
               for (i, p) in enumerate(params)]

    self.weights = [self.iterations] + moments + [self.iter_count]
    for p, g, m in zip(params, grads, moments):
        v = p - lr * g
        new_p = v + momentum * (v - m)
        self.updates.append(K.update(m, v))

        # Apply constraints.
        if getattr(p, 'constraint', None) is not None:
            new_p = p.constraint(new_p)

        self.updates.append(K.update(p, new_p))

    condition = K.all(K.less(self.iter_count, self.restarting_iter))
    new_iter_count = K.switch(condition, self.iter_count + 1, self.
                              iter_count - self.restarting_iter + 1)
    self.updates.append(K.update(self.iter_count, new_iter_count))

    return self.updates

def get_config(self):
    config = {'learning_rate': float(K.get_value(self.learning_rate))
            ,
            'decay': float(K.get_value(self.decay)),
            'iter_count': int(K.get_value(self.iter_count)),
            'restarting_iter': int(K.get_value(self.restarting_iter)
                                   )})
    base_config = super(SRSGD, self).get_config()
    return dict(list(base_config.items()) + list(config.items()))

```