SALR: Sharpness-Aware Learning Rate Scheduler for Improved Generalization

Xubo Yue, Maher Nouiehed, and Raed Al Kontar

***Abstract*— In an effort to improve generalization in deep learning and automate the process of learning rate scheduling, we propose SALR: a sharpness-aware learning rate update technique designed to recover flat minimizers. Our method dynamically updates the learning rate of gradient-based optimizers based on the local sharpness of the loss function. This allows optimizers to automatically increase learning rates at sharp valleys to increase the chance of escaping them. We demonstrate the effectiveness of SALR when adopted by various algorithms over a broad range of networks. Our experiments indicate that SALR improves generalization, converges faster, and drives solutions to significantly flatter regions.**

***Index Terms*— Deep learning, generalization, learning rate schedule, sharpness.**

I. Introduction

GENERALIZATION in deep learning has recently been an active area of research. The efforts to improve generalization over the past two decades have brought upon many cornerstone advances and techniques; be it dropout [1], batch-normalization [2], data-augmentation [3], weight decay [4], adaptive gradient-based optimization [5], architecture design and search [6], ensembles and their Bayesian counterparts [7], [8], amongst many others. Yet, recently, researchers have discovered that the concept of sharpness/flatness plays a fundamental role in generalization.

Though sharpness was first discussed in the context of neural networks in the early work of Hochreiter and Schmidhuber [9], it was only brought to the forefront of deep learning research after the seminal paper by Keskar et al. [10]. While trying to investigate decreased generalization performance when large batch sizes are used in stochastic gradient descent (SGD) [10], [11] noticed that this phenomenon can be justified by the ability of smaller batches to reach flat minimizers. Such flat minimizers, in turn, generalize well as they are robust to low precision arithmetic or noise in the parameter space [12], [13], [14].

Manuscript received 4 September 2022; revised 7 December 2022; accepted 27 March 2023. This work was supported by the National Science Foundation under Grant CMMI 1931950. *(Corresponding author: Raed Al Kontar.)*

Xubo Yue and Raed Al Kontar are with the Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: alkontar@umich.edu).

Maher Nouiehed is with the Department of Industrial Engineering and Management, American University of Beirut, Beirut 1107 2020, Lebanon.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2023.3263393.

Digital Object Identifier 10.1109/TNNLS.2023.3263393

Since then, the generalization ability of flat minimizers has been repeatedly observed in many recent works [8], [15], [16], [17], [18], [19]. Indeed, flat minimizers can potentially tie together many of the aforementioned approaches aimed at generalization. For instance: 1) higher gradient variance increases the probability of avoiding sharp regions [same can be said for SGD compared to gradient descent (GD)] [13]; 2) averaging over multiple hypotheses leads to wider optima in ensembles and Bayesian deep learning (BDL) [8]; and 3) regularization techniques such as dropout and over-parameterization can adjust the loss landscape into one that allows first-order methods to favor wide valleys [18], [20].

In this article, we study the problem of developing an algorithm that can converge to flat minimizers. Specifically, we introduce SALR: a sharpness-aware learning rate designed to explore the loss surface of an objective function and avoid undesired sharp local minima. SALR dynamically updates the learning rate based on the sharpness of the neighborhood of the current solution. The idea is simple: automatically increase the learning rates at relatively sharp valleys in an effort to escape them. One key feature of SALR is its ability to be adopted by any gradient-based method such as **Adagrad** [21], **Adam** [5] **and also into recent approaches toward escaping sharp valleys such as Entropy-SGD** [18]. Our contributions are summarized below.

1) Motivated by recent results on the improved generalization capability of flat minimizers, we propose SALR: a dynamic learning rate update mechanism that utilizes the sharpness of the underlying landscape. In particular, our mechanism increases the learning rate η in sharp regions to increase the chance of escaping them. Our dynamic learning rate schedule is based on a sharpness measure $S(\cdot, \cdot)$ that quantifies the sharpness of current parameter iterates using n_1 steps of stochastic gradient ascent (SGA) and n_2 steps of SGD (more details in Section III). Our framework can be adopted by a wide variety of gradient-based methods. We then show that GD-SALR (GD adopting our learning rate schedule) can escape strongly convex local minimizers.
2) We demonstrate the improved generalization achieved when adopting SALR on various optimization methods (SGD, Adam, and Entropy-SGD) applied to a wide range of applications (image classifications, text prediction, and fine-tuning) and using a variety of network structures and datasets.

3) **SALR circumvents the practical challenge of setting a heuristic learning rate schedule when training deep learning models.** Instead, SALR dynamically chooses to learn rates to recover flat solutions.

We note that the code that implements SALR can be found at the following GitHub link.

A. Related Work

From a theoretical perspective, the generalization of deep learning solutions has been explained through multiple lenses. One of which is uniform stability [22], [23], [24], [25], [26]. An algorithm is uniformly stable if for all data sets differing in only one element, nearly the same outputs will be produced [27]. Hardt et al. [24] show that SGD satisfies this property and derives a generalization bound for models learned with SGD. From a different viewpoint, [28], [29], [30], [31] attribute generalization to the complexity of the hypothesis space. Using measures like Rademacher complexity [32] and the Vapnik-Chervonenkis (VC) dimension [33], the former works show that deep hypothesis spaces are typically more advantageous in representing complex functions. Besides that, the importance of flatness on generalization has been theoretically highlighted through PAC-Bayes bounds [34], [35], [36]. These papers highlight the ability to derive non-vacuous generalization bounds based on the sharpness of a model class while arguing that relatively flat solutions yield tight bounds.

From an algorithmic perspective, approaches to recover flat minima are still limited. Most notably, [18] developed the Entropy-SGD algorithm. Entropy-SGD defines a local-entropy-based objective that smoothens the energy landscape based on its local geometry. This, in turn, allows SGD to attain flatter solutions. Indeed, this approach was motivated by earlier work in statistical physics [37], [38], which proves the existence of non-isolated solutions that generalize well in networks with discrete weights. Such non-isolated solutions correspond to flat minima in continuous settings. The authors then propose a set of approaches based on ensembles and replicas of the loss to favor wide solutions. Not too far, recent methods in BDL have also shown the potential to recover flat minima. BDL averages over multiple hypotheses weighted by their posterior probabilities (ensembles being a special case of BDL [8]). One example is the stochastic weighted averaging (SWA) algorithm proposed by Izmailov et al. [8]. SWA simply averages over multiple points along the trajectory of SGD to potentially find flatter solutions compared to SGD. Another example is the SWA-Gaussian (SWAG). SWAG defines a Gaussian posterior approximation over neural network weights. Afterward, samples are taken from the approximated distribution to perform Bayesian model averaging [39]. Besides Entropy-SGD and SWA, Wen et al. [40] propose a method SmoothOut to smooth out sharp minima by averaging over multiple perturbed copies of the landscape. More recently, Foret et al. [19] proposed a sharpness-aware minimization (SAM) method for finding a flat minimizer. In particular, SAM solves a min-max optimization problem that minimizes the maximum loss when parameters are allowed a small perturbation.

Another recent work that motivates our framework is the method proposed by Patel [41]. The aforementioned observations in [10] and [41] show that the learning rate lower-bound threshold for the divergence of batch SGD, run on quadratic optimization problems, increases for larger batch sizes. More generally, in non-convex settings, given a problem with N local minimizers, one can compute N lower bound thresholds for the local divergence of batch SGD. The number of minimizers for which batch SGD can converge is non-decreasing in the batch size. This is used to explain the tendency of small-batch SGD to converge to flatter minimizers compared to large-batch SGD. The former result links the choice of batch size and its effect on generalization to the choice of the learning rate. With the latter being a tunable parameter, to our knowledge, developing a dynamic choice of the learning rate that targets convergence to flat minimizers has not been studied. Here we note that the literature on dynamic rate schedulers abounds [42], [43], [44]. Yet, this work is the first to design dynamic rates to recover flat minimizers.

II. General Framework

This article proposes a framework that dynamically chooses a *SALR* to promote convergence to flat minimizers. More specifically, our proposed method locally approximates sharpness at the current iterate and dynamically adjusts the learning rate accordingly. In sharp regions, relatively large learning rates are attained to increase the chance of escaping that region. In contrast, our method returns a relatively small learning rate to guarantee convergence when the current iterate belongs to a flat region. Our framework can be adopted by any local search descent method and is detailed in Algorithm 1.

Algorithm 1 SALR Framework

```
Input: Starting point $\boldsymbol{\theta}_0$, initial learning rate $\eta_0$, number of iterations $K$.
for $k = 0, 1, \ldots, K$ do
    Estimate $\widehat{S}_k$, the local sharpness around the current iterate $\boldsymbol{\theta}_k$
    Set $\eta_k = \eta_0 \frac{\widehat{S}_k}{\text{Median}\{\widehat{S}_i\}_{i=1}^k}$
    Compute $\boldsymbol{\theta}_{k+1}$ using some local search descent method (Gradient Descent, Stochastic Gradient Descent, Adam, ...)
end for
Return $\boldsymbol{\theta}_K$
```

As detailed in Algorithm 1, at every iterate k, we compute the learning rate as a function of the local sharpness parameters $\{\widehat{S}_k\}_{i=1}^k$. The main intuition is to have the current learning rate to be an increasing function of the currently estimated sharpness. Since the scale of the sharpness at different points can vary when using different networks or datasets [12], we normalize our estimated sharpness by dividing by the median of the sharpness of previous iterates. For instance, in Fig. 1, despite having a similar sharpness measure, we consider the minimizer around $\boldsymbol{\theta} = 1$ to be sharp relative to the blue plot and flat relative to the red plot. Normalization resolves this issue by helping our sharpness measure attain scale-invariant properties.

One can think of the median of previous sharpness values as a global sharpness parameter the algorithm is trying to learn. When k is sufficiently large, the variation in the global sharpness parameter among different iterates will be minimal. From an algorithmic perspective, SALR exploits a neighborhood around the current iterate to dynamically compute a desired learning rate while simultaneously exploring the sharpness of the landscape to refine this global sharpness parameter.

III. Sharpness Measure

Several sharpness/flatness measures have been defined in recent literature [9], [10], [45]. For instance, Hochreiter and Schmidhuber [9] compute flatness by measuring the size of the connected region in the parameter space where the objective remains approximately constant. In a more recent paper, Rangamani et al. [45] proposed a scale-invariant flatness measure based on the quotient manifold. Computing such notions for complex non-convex landscapes can be intractable in practice. In addition to the cited results, Keskar et al. [10] quantify flatness by finding the difference between the maximum value of the loss function within a small neighborhood around a given point and the current value. More specifically, they define sharpness as follows:

$$\phi(\varepsilon, \boldsymbol{\theta}) \triangleq \frac{S(\varepsilon, \boldsymbol{\theta})}{1 + f(\boldsymbol{\theta})} \quad \text{and} \quad S(\varepsilon, \boldsymbol{\theta}) = \max_{\boldsymbol{\theta}' \in \mathbb{B}_\varepsilon(\boldsymbol{\theta})} f\left(\boldsymbol{\theta}'\right) - f(\boldsymbol{\theta}) \tag{1}$$

where $\mathbb{B}_\varepsilon(\boldsymbol{\theta})$ is a euclidean ball with radius ε centered at $\boldsymbol{\theta}$ and $1 + f(\boldsymbol{\theta})$ is a normalizing coefficient. One drawback of (1) is that the sharpness value around a maximizer is nearly zero. To resolve this issue, one can simply modify the sharpness measure in (1) as follows:

$$S(\varepsilon, \boldsymbol{\theta}) \triangleq \max_{\boldsymbol{\theta}' \in \mathbb{B}_\varepsilon(\boldsymbol{\theta})} f\left(\boldsymbol{\theta}'\right) - \min_{\boldsymbol{\theta}' \in \mathbb{B}_\varepsilon(\boldsymbol{\theta})} f\left(\boldsymbol{\theta}'\right). \tag{2}$$

It can be easily shown that if $\boldsymbol{\theta}$ is a local minimizer, (2) is equivalent to (1). Both measures defined in (1) and (2) require solving a possibly non-convex function which is in general NP-Hard. For computational feasibility, we provide a sharpness approximation by running n_1 gradient ascent and n_2 GD steps. The resulting solutions are used to approximate the maximization and minimization optimization problems. Here we note that our definition for sharpness does not include a normalizing coefficient, as $\text{median}\{\widehat{S}_i\}_{i=1}^k$ in Algorithm 1 plays this role. The details of the approximation are shown in Definition 1.

Definition 1: Given $\boldsymbol{\theta} \in \mathbb{R}^n$, iteration numbers n_1 and n_2, and step-size γ, we define the sharpness measure

$$\begin{aligned} \widehat{S}(\boldsymbol{\theta}) &\triangleq f\left(\boldsymbol{\theta}_{k,+}^{(n_2)}\right) - f(\boldsymbol{\theta}_k) + f(\boldsymbol{\theta}_k) - f\left(\boldsymbol{\theta}_{k,-}^{(n_1)}\right) \\ &= f\left(\boldsymbol{\theta}_{k,+}^{(n_2)}\right) - f\left(\boldsymbol{\theta}_{k,-}^{(n_1)}\right) \end{aligned}$$

where $\boldsymbol{\theta}_{k,-}^{(n_1)} = \boldsymbol{\theta}_k - \sum_{i=0}^{n_1-1}((\gamma \nabla f(\boldsymbol{\theta}_{k,-}^{(i)}))/\|\nabla f(\boldsymbol{\theta}_{k,-}^{(i)})\|)$, $\boldsymbol{\theta}_{k,+}^{(n_2)} = \boldsymbol{\theta}_k + \sum_{i=0}^{n_2-1} \gamma((\nabla f(\boldsymbol{\theta}_{k,+}^{(i)}))/\|\nabla f(\boldsymbol{\theta}_{k,+}^{(i)})\|)$ and $\boldsymbol{\theta}_{k,+}^{(0)} = \boldsymbol{\theta}_{k,-}^{(0)} = \boldsymbol{\theta}_k$.

Remark 1: In contrast to the measures defined in (2) and (1), Definition 1 does not require a ball radius ε. However, our definition requires specifying the step size γ and the number of ascent and descent iterations.

Remark 2: Running GD/gradient ascent with a fixed step size near a minimizer can return a very small sharpness value even if the minimizer is sharp. This is due to the small gradient norm around a minimizer. To resolve this issue, we normalize the gradient at every descent/ascent step. Moreover, normalizing by the norm of the gradient helps in understanding the radius of the ball containing the iterates $\{\boldsymbol{\theta}_{k,-}^{(j)}\}_{j=1}^{n_1}$ and $\{\boldsymbol{\theta}_{k,+}^{(j)}\}_{j=1}^{n_2}$.

Remark 3: According to Definition 1, larger gradient magnitudes yield larger sharpness values. Adam, a very popular method for training neural networks, tends to decrease the learning rate when the accumulated gradients are large; i.e., according to our definition of sharpness, the method returns a reduced learning rate in sharp regions. This can be one explanation for the common conception that Adam converges faster than SGD but generalizes worse [46], [47]. Our proposed method adopts a reverse behavior. More specifically, we aim at choosing a large learning rate in sharp regions. Our experiments indicate that this approach helps find flat local minima. Interestingly, our experiments further show that SALR can improve its performance of Adam when adopting our learning rate schedule strategy.

Fig. 2 shows the plots of the three different sharpness measures defined in this section when computed for a function $f(\theta) = 0.5\theta \sin(3\theta) + 1$. Notice that the blue plot corresponding to the sharpness measure $\phi(\cdot)$ attains a zero value at local maximizers compared to a positive value for the other two sharpness plots. Moreover, notice that the sharpness value in these three plots attains a small value near the local minimizer. This can be explained by our choice of radius $\varepsilon = 0.1$ which limits the neighborhood being exploited. Increasing the radius for $\phi(\varepsilon, \cdot)$ and $S(\varepsilon, \cdot)$ (increasing n_1 and n_2 for $\widehat{S}$) will provide higher values around the minimizer. We next show that using our sharpness measure in Definition 1, GD with SALR framework in Algorithm 1, denoted as GD-SALR, escapes sharp local minima.

IV. Theoretical Results

In this section, we focus on analyzing the convergence of vanilla GD when adopting our SALR framework in Algorithm 1. We show that GD-SALR escapes any given neighborhood of a sharp local minimum by choosing a sufficiently large step size. Throughout this section, we make the following assumptions that are standard for the convergence theory of GD methods.

Assumptions: The objective function f is twice continuously differentiable and L_0-Lipschitz continuous. The gradient function $\nabla f(\cdot)$ is L-Lipschitz continuous with Lipschitz constant L. Furthermore, the gradient norm is bounded, i.e., there exists a scalar constant $g_{\max} > 0$ such that $\|\nabla f(\boldsymbol{\theta})\| \leq g_{\max}$ for all $\boldsymbol{\theta}$.

The next theorem shows that GD with a sufficiently large step size escapes a given strongly convex neighborhood around a local minimum $\boldsymbol{\theta}^*$. The proof of the theorem is relegated to Appendix.

Theorem 1: Suppose that f is μ-strongly convex function in a neighborhood $\mathbb{B}_\delta(\boldsymbol{\theta}^*)$ around a local minimum $\boldsymbol{\theta}^*$, i.e.,

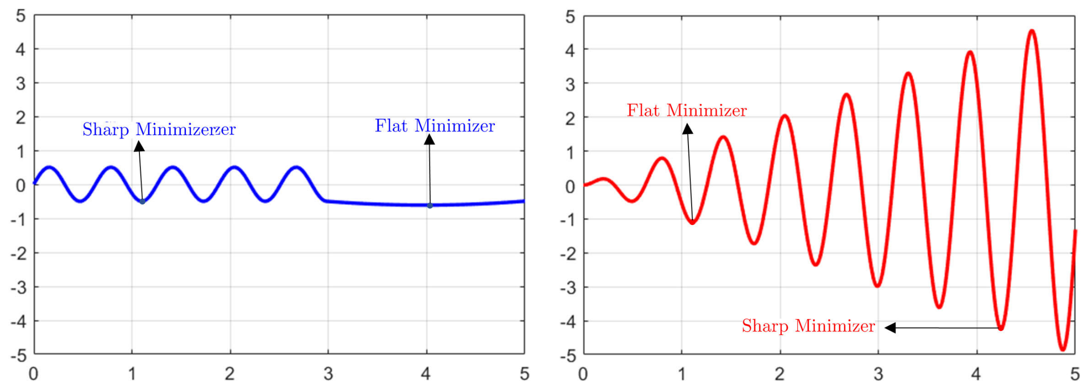


Fig. 1. Sharp/flat minimizers relative to the landscape.

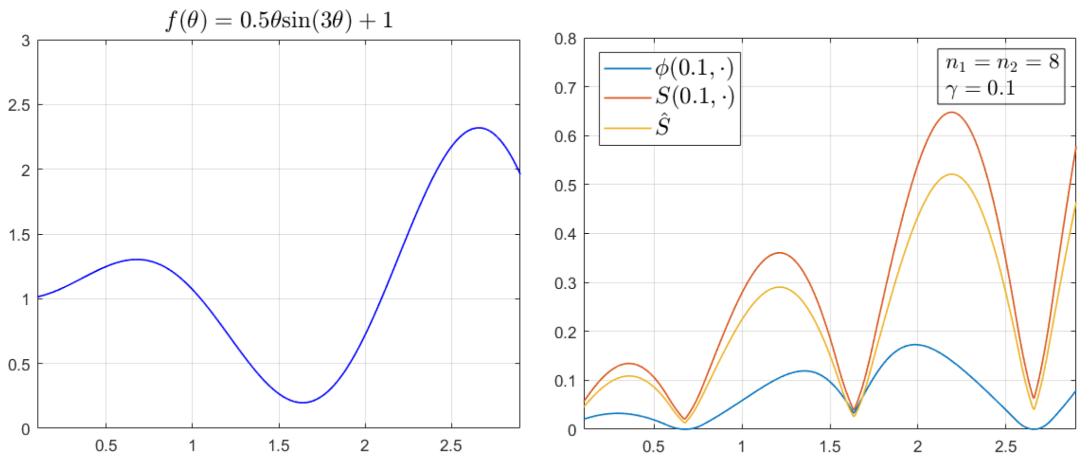


Fig. 2. Sharpness measure plots for ϕ, S, and $\widehat{S}$ on function f.

$\lambda_{\min}(\nabla^2 f(\boldsymbol{\theta})) \geq \mu$ for all $\boldsymbol{\theta} \in \mathbb{B}_\delta(\boldsymbol{\theta}^*) \triangleq \{\boldsymbol{\theta} \mid \|\boldsymbol{\theta} - \boldsymbol{\theta}^*\|_2 \leq \delta\}$. Running vanilla GD with $\boldsymbol{\theta}_0 \in \mathbb{B}_\delta(\boldsymbol{\theta}^*)$ and learning rate $\eta_k \geq ((2+\varepsilon)/\mu)$ for some fixed $\varepsilon > 0$, there exists $\widehat{k}$ with $\boldsymbol{\theta}_{\widehat{k}} \notin \mathbb{B}_\delta(\boldsymbol{\theta}^*)$.

Our next result shows that GD-SALR escapes sharp local minima by dynamically choosing a sufficiently large step size in a local strongly convex region. The proof is relegated to Appendix.

Theorem 2: Suppose that f is a μ-strongly convex function in a neighborhood $\mathbb{B}_\delta(\boldsymbol{\theta}^*)$ around a local minimum $\boldsymbol{\theta}^*$, i.e., $\lambda_{\min}(\nabla^2 f(\boldsymbol{\theta})) \geq \mu$ for all $\boldsymbol{\theta} \in \mathbb{B}_\delta(\boldsymbol{\theta}^*) \triangleq \{\boldsymbol{\theta} \mid \|\boldsymbol{\theta} - \boldsymbol{\theta}^*\|_2 \leq \delta\}$. Under our Assumptions, run GD-SALR (GD with step size choice according to Algorithm 1 and Definition 1) with

$$n_1 \geq \frac{a_1}{\left(\log\left(1 + \frac{\mu g_{\min}}{L g_{\max} - \mu g_{\min}}\right)\right)^2} - \frac{1}{a_1}$$

$$n_2 \geq \frac{a_2}{\left(\log\left(1 + \frac{\mu g_{\min}}{L g_{\max}}\right)\right)^2} - \frac{1}{a_2}, \quad \text{and} \quad \gamma = \frac{g_{\min}}{L}$$

where

$$a_1 = \frac{(2+\epsilon)L_0}{\eta_0(g_{\max}L - g_{\min}\mu)} + \frac{g_{\min}\mu}{2(g_{\max}L - g_{\min}\mu)}$$

and

$$a_2 = \frac{(2+\epsilon)L_0}{\eta_0 g_{\max} L} + \frac{\mu^2 g_{\min}}{2L^2 g_{\max}}, \quad \epsilon, \eta_0 > 0$$

and $g_{\min} > 0$ is a lower bound that satisfies

$$\max\left\{\left\|\nabla f\left(\boldsymbol{\theta}_{k,-}^{(n_1-1)}\right)\right\|, \|\nabla f(\boldsymbol{\theta}_k)\|\right\} \geq g_{\min}.$$

If $\delta > \max\{n_1, n_2\}\gamma$, then there exists $\widehat{k}$ with $\boldsymbol{\theta}_{\widehat{k}} \notin \mathbb{B}_\delta(\boldsymbol{\theta}^*)$.

Our theorem states that with sufficient ascent/descent steps and under local strong-convexity assumptions around local minimizers (i.e., local minimizer is sharp), GD-SALR can escape the neighborhood by choosing a large enough step size. Here, a higher μ value (strong convexity parameter) reflects sharper minimizers. As such, our results are local and do not require the convexity of the objective. Also, note that n_1 and n_2 are not necessarily increasing with μ. However, we only require that the number of descent and ascent steps performed by our algorithm need to be above the thresholds in our Theorem.

When the number of ascent/descent steps is fixed, GD-SALR will only be able to escape the sharper minima. Indeed, the overarching goal of SALR is to escape minimizers that are in sharper regions compared to the landscape. This is why our algorithm learns the sharpness landscape as it proceeds to learn the relatively sharper regions.

Remark 4: In the context of machine learning, our theorem shows that our algorithm can potentially escape sharp regions even when all the data are used (full-batch). For instance, Patel [41] shows that when using large batch sizes, we require a higher learning rate to escape sharp minima. This provides insight into the favorable empirical results presented in Section VI when running SGD-SALR. Moreover, our dynamic choice of high learning rates in sharp regions can potentially allow running SGD with larger batch sizes while still escaping sharp minimizers. This in turn provides an avenue for improved parallelism [48], [49].

Lastly, we note that in Theorem 2, we require to set $\gamma = (g_{\min}/L)$ which may be a challenging condition to satisfy in practice. Indeed, this is an intrinsic issue that is not unique to SALR but common in many deep learning optimization methods. For instance, similar conditions on the learning rate γ are needed for deep learning theory papers and many state-of-the-art algorithms such as noisy SGD, SGD with restarts, SAM, to name a few [20], [50], [51], [52], [53]. Such papers require the learning rate γ satisfies $\gamma = (1/L)$ or $\gamma \leq \mathcal{O}((1/L))$. In reality, we rarely know the value of L. However, an arbitrary large choice of learning rate has been shown to work well empirically. Furthermore, one can resort to cross-validation or grid-search to tune γ.

V. Stochastic Approximation of Sharpness

The concept of generalization is more relevant when solving problems arising in machine learning settings. Under the empirical risk minimization framework, the problem of training machine learning models can be mathematically formulated as the following optimization problem:

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^n} f(\boldsymbol{\theta}) \triangleq \frac{1}{m}\sum_{i=1}^{m} f_i(\boldsymbol{\theta}) \tag{3}$$

where f_i is a loss function parameterized with parameter $\boldsymbol{\theta}$ corresponding to data point $i \in \{1, 2, \ldots, m\}$. The most popular algorithm used to solve such optimization problems is SGD which iteratively updates the parameters using the following update rule:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta_k\left(\frac{1}{|B_k|}\sum_{i\in B_k}\nabla f_i(\boldsymbol{\theta}_k)\right)$$

where B_k is the batch sampled at iteration k and η_k is the learning rate. To apply our framework in stochastic settings, we provide a stochastic procedure for computing the sharpness measure at a given iterate. Details are provided in Algorithm V. By adopting Algorithm V, our framework can be applied to numerous popular algorithms like SGD, Adam, and Entropy-SGD.

The detailed implementation of SGD-SALR and ADAM-SALR is shown in Algorithms V and V, respectively.

Algorithm 2 Calculating Stochastic Sharpness at Iteration k

```
Input: batch size $B_k$, base learning rate $\gamma$, current iterate $\boldsymbol{\theta}_k$, iteration number $n_1, n_2$
Set $\boldsymbol{\theta}_{k,+}^{(0)} = \boldsymbol{\theta}_{k,-}^{(0)} = \boldsymbol{\theta}_k$
for $i = 0 : n_1 - 1$ do
    $\boldsymbol{\theta}_{k,-}^{(i+1)} = \boldsymbol{\theta}_{k,-}^{(i)} - \gamma\left(\frac{1}{|B_k|}\sum_{j\in B_k}\frac{\nabla f_j(\boldsymbol{\theta}_{k,-}^{(i)})}{\|\nabla f_j(\boldsymbol{\theta}_{k,-}^{(i)})\|}\right)$
end for
for $i = 0 : n_2 - 1$ do
    $\boldsymbol{\theta}_{k,+}^{(i+1)} = \boldsymbol{\theta}_{k,+}^{(i)} + \gamma\left(\frac{1}{|B_k|}\sum_{j\in B_k}\frac{\nabla f_j(\boldsymbol{\theta}_{k,+}^{(i)})}{\|\nabla f_j(\boldsymbol{\theta}_{k,+}^{(i)})\|}\right)$
end for
Set $\hat{S}_k = \frac{1}{|B_k|}\sum_{j\in B_k} f_j(\boldsymbol{\theta}_{k,+}^{(n_2)}) - \frac{1}{|B_k|}\sum_{j\in B_k} f_j(\boldsymbol{\theta}_{k,-}^{(n_1)})$
```

Algorithm 3 SGD-SALR Algorithm

```
Input: base learning rate $\eta_0$, number of iterations $K$, frequency $c$, initial weight $\boldsymbol{\theta}_0$ Set $\mathcal{S} = \emptyset$
for $k = 0 : K$ do
    if $k \mod c = 0$ then
        Calculate $\widehat{S}_k$ using Algorithm 2
    end if
    Compute $\mathcal{S} = \text{Median}(\{\widehat{S}_k\})$
    Set $\eta_k = \eta_0\frac{\widehat{S}_k}{\mathcal{S}}$   $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta_k\frac{1}{|B_k|}\sum_{j\in B_k}\nabla f_j(\boldsymbol{\theta}_k)$
end for
Set $\boldsymbol{\theta} = \boldsymbol{\theta}_K$
Return $\boldsymbol{\theta}$
```

Algorithm 4 ADAM-SALR Algorithm

```
Input: base learning rate $\eta_0$, exponential decay rates for the moment estimates $\beta_1, \beta_2 \in [0, 1)$, number of iterations $K$, frequency $c$, initial weight $\boldsymbol{\theta}_0$, perturbation $\epsilon$
Set $\mathcal{S} = \emptyset$
Set $m_0 = v_0 = 0$
for $k = 0 : K$ do
    if $k \mod c = 0$ then
        Calculate $\widehat{S}_k$
    end if
    Compute $\mathcal{S} = \text{Median}(\{\widehat{S}_k\})$
    Set $\eta_k = \eta_0\frac{\hat{S}_k}{\mathcal{S}}$
    $g = \frac{1}{|B_k|}\sum_{j\in B_k}\nabla f_j(\boldsymbol{\theta}_k)$
    $m_{k+1} = \beta_1 m_k + (1-\beta_1)g$
    $v_{k+1} = \beta_2 v_k + (1-\beta_2)g^2$
    $\widehat{m}_{k+1} = m_k/(1-\beta_1^{k+1})$
    $\widehat{v}_{k+1} = v_k/(1-\beta_2^{k+1})$
    $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta_k\widehat{m}_{k+1}/(\sqrt{\widehat{v}_{k+1}} + \epsilon)$
end for
Set $\boldsymbol{\theta} = \boldsymbol{\theta}_K$
Return $\boldsymbol{\theta}$.
```

VI. Empirical Results

In this section, we present experimental results on image classification, text prediction, and finetuning tasks. We show that our framework SALR can be adopted by many optimization methods and achieve notable improvements over a broad range of networks. We compare SALR with Entropy-SGD [18], SWA [8] and SAM [19]. Besides those benchmarks, we also use the conventional SGD and Adam [5] as baseline references. All aforementioned methods are trained with batch normalization [2] and dropout of probability 0.5 after each layer [1]. We replicate each experiment ten times to

TABLE I
CLASSIFICATION ACCURACY ON CIFAR10

Network	SGD	SWA	Entropy-SGD	Entropy-SGD-SALR	SAM	SGD-SALR
ResNet50	93.25 (0.03)	93.31 (0.06)	93.77 (0.08)	94.47 (0.12)	94.50 (0.08)	**94.94** (0.09)
All-CNN-BN	91.93 (0.01)	92.20 (0.01)	91.13 (0.01)	92.16 (0.01)	92.18 (0.03)	**92.45** (0.05)
ResNet101	95.11 (0.02)	95.56 (0.01)	95.51 (0.01)	95.87 (0.01)	95.92 (0.03)	**95.99** (0.00)
RegNetX	94.24 (0.02)	94.23 (0.01)	94.26 (0.01)	95.00 (0.01)	94.39 (0.05)	**95.01** (0.01)
Network	**Adam**	**SWA**	**Entropy-Adam**	**Entropy-Adam-SALR**	**SAM**	**Adam-SALR**
ResNet50	92.91 (0.07)	92.43 (0.06)	92.61 (0.11)	93.15 (0.09)	93.27 (0.10)	**93.61** (0.09)
All-CNN-BN	91.95 (0.01)	92.27 (0.01)	91.10 (0.01)	92.15 (0.01)	92.13 (0.03)	**92.35** (0.01)
ResNet101	95.00 (0.01)	95.57 (0.01)	95.56 (0.01)	96.03 (0.01)	**96.17** (0.08)	95.97 (0.00)
RegNetX	94.33 (0.01)	94.12 (0.01)	94.21 (0.01)	**95.06** (0.01)	95.04 (0.03)	95.02 (0.01)

TABLE II
SHARPNESS OF FINAL SOLUTIONS (CIFAR-10 AND SGD)

$\times 10^{-3}$	SGD	SWA	Entropy-SGD	Entropy-SGD SALR	SAM	SGD-SALR
ResNet50	8.19 (0.50)	7.51 (0.31)	3.55 (0.47)	3.67 (0.28)	3.45 (0.51)	**3.22** (0.63)
All-CNN-BN	11.02 (1.00)	10.65 (1.21)	**6.12** (0.88)	6.35 (0.84)	6.34 (0.55)	6.30 (0.91)
ResNet101	7.00 (0.87)	6.91 (0.22)	5.98 (0.60)	6.07 (0.82)	6.09 (0.65)	**5.53** (0.70)
RegNetX	9.56 (1.03)	9.66 (0.69)	9.41 (0.50)	8.77 (0.68)	8.69 (0.56)	**8.50** (0.71)

obtain the mean and standard deviation of testing errors. Moreover, to account for the overhead computation of our proposed method, we ensure all models have the *same total number of gradient calls*. We consider some typical networks such as WideResNet [54], ResNet [55], DenseNet [56], MobileNetV2 [57] and RegNetX [6].

A. Image Classification

1) CIFAR-10/100: We run our proposed SGD-SALR method detailed in Algorithm V for 40 epochs. We collect the sharpness measure every $c = 2$ iteration and set $n_1 = n_2 = 5$. The experimental settings for other benchmark models are as follows.

1) **SGD:** We run SGD for 200 epochs using decay learning rates.
2) **SWA:** The setting is the same as SGD. In the SWA stage, we switch to a cyclic learning rate schedule as suggested in [8].
3) **Entropy-SGD:** Following the setting in [18], we train Entropy-SGD for 40 epochs and set Langevin iterations $L_a = 5$. We set the learning rate of the outer loop to $\gamma = 1$ and it drops by a factor of 5 every four epochs. The learning rate of the SGLD is set to be $\eta' = 0.1$ with noise $\epsilon = 10^{-4}$. The initial γ_0 is 0.03 and increased by a factor of 1.001 after each parameter update.
4) **Entropy-SGD-SALR:** The setting is same as Entropy-SGD. However, we update the learning rate of Entropy-SGD using Algorithm 1.
5) **SAM** [19]: We run SAM for 100 epochs (in SAM, each iteration has two gradient calls). The choice of hyperparameter follows the guideline in the SAM paper. The results are reported in Table I. Overall, all methods have the same number of gradient calls (i.e., wall-clock times).

To illustrate the flexibility of our framework, we change the base optimizer SGD to Adam and re-run all the experiments under a similar setting as that of Table I. Results are reported in the same Table. Furthermore, in Table II, we report the sharpness measure of the final solution obtained by each optimization approach. We specifically report the sharpness measure $\widehat{S}_k$ without dividing by the median in Table II to ensure a fair comparison.

Under a similar setting, we test all models on CIFAR-100. The batch size is 256. Experimental results are reported in Table III. In this experiment, all methods also run with the same number of gradient calls.

2) ImageNet: Following the procedures in [55] and [58], we train SALR and all benchmark models on ImageNet using ResNet152 and DenseNet161. The batch size is chosen to be 4096. The best testing accuracies are reported in Table IV. Furthermore, we increase the number of training epochs and report results in the same Table.

As seen in Table IV, SALR can improve the performance of SGD even on the complex image classification task. Notably, as we increase the number of training epochs, the performance of SGD and SWA decreases due to model overfitting. However, SAM, Entropy-SGD, and SGD-SALR do not suffer from this issue. This further demonstrates the advantage of sharpness-aware algorithms.

B. Text Prediction

We train an LSTM network on the Penn Tree Bank (PTB) dataset for word-level text prediction. This dataset contains about one million words. Following the guideline

TABLE III
Classification Accuracy on CIFAR-100

Network	**SGD**	**SWA**	**SAM**	**Entropy-SGD**	**SGD-SALR**
ResNet18	79.20 (0.13)	79.63 (0.08)	81.30 (0.10)	81.17 (0.12)	**81.73** (0.10)
RegNetX	79.42 (0.05)	79.45 (0.08)	81.55 (0.09)	81.61 (0.10)	**82.00** (0.13)
MobileNetV2	81.22 (0.06)	81.18 (0.09)	82.07 (0.06)	**82.53** (0.05)	82.47 (0.09)
WideResNet-28-10	82.00 (0.10)	82.33 (0.09)	**83.52** (0.19)	82.37 (0.12)	83.12 (0.09)
PyramidNet	80.02 (0.08)	80.31 (0.09)	85.31 (0.17)	84.88 (0.13)	**85.69** (0.10)

TABLE IV
Classification Accuracy on ImageNet

Network	**SGD 200 epochs**	**SWA 200 epochs**	**SAM 100 epochs**	**Entropy-SGD 40 epochs**	**SGD-SALR 40 epochs**
ResNet152	78.83 (0.08)	79.21 (0.09)	80.00 (0.11)	79.44 (0.12)	**80.33 (0.10)**
DenseNet161	77.61 (0.07)	78.24 (0.13)	78.88 (0.07)	78.90 (0.10)	**79.10 (0.07)**
Network	**SGD 500 epochs**	**SWA 500 epochs**	**SAM 250 epochs**	**Entropy-SGD 100 epochs**	**SGD-SALR 100 epochs**
ResNet152	78.20 (0.06)	78.21 (0.06)	80.89 (0.11)	79.94 (0.11)	**81.15 (0.09)**
DenseNet161	77.13 (0.12)	77.21 (0.10)	79.25 (0.10)	78.99 (0.10)	**79.37 (0.11)**
Network	**SGD 800 epochs**	**SWA 800 epochs**	**SAM 400 epochs**	**Entropy-SGD 160 epochs**	**SGD-SALR 160 epochs**
ResNet152	78.03 (0.01)	78.08 (0.00)	81.59 (0.01)	81.05 (0.02)	**81.70** (0.03)
DenseNet161	77.00 (0.03)	77.01 (0.02)	79.33 (0.03)	**79.43** (0.01)	79.40 (0.01)

TABLE V
Perplexity on PTB/WP

PTB	**SGD**	**SWA**	**Entropy-SGD**	**SAM**	**SGD-SALR**
PTB-LSTM	78.4 (0.22)	78.1 (0.25)	72.15 (0.16)	72.10 (0.18)	**71.42 (0.14)**
WP-LSTM	1.223 (0.01)	1.220 (0.05)	1.095 (0.01)	1.091 (0.02)	**1.089 (0.02)**

in [1] and [2], we train PTB-LSTM with 66 million weights. SGD and SWA are trained with 55 epochs. Entropy-SGD ($L = 5$) and SALR ($c = 2$) are trained with 11 epochs. SAM is trained with 28 epochs. Overall, all methods have the same number of gradient calls.

Our second task is to train an LSTM to perform character-level text prediction using War and Peace (WP). We follow the procedures in [2] and [3]. We train SGD/SWA and Entropy-SGD/SALR with 50 and ten epochs, respectively. We report the perplexity on the test set in Table V.

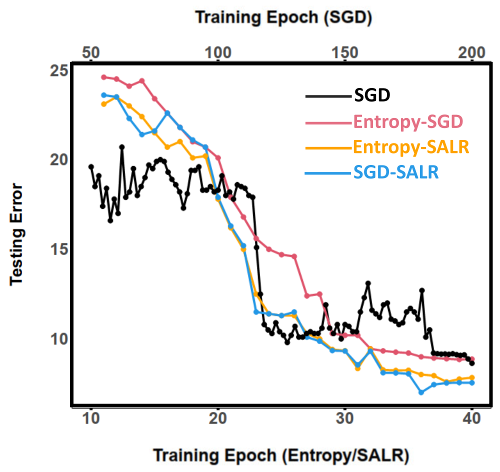
C. Fine-Tuning

We test our algorithm and other benchmarks on several finetuning tasks. We use EfficientNet-b7 and EfficientNet-L2 pre-trained on ImageNet. Weight parameters are initialized to the values provided by the publicly available checkpoints [59] while a new output layer with random weights is added to accommodate labels from a new dataset. Following the guideline in [19], we use $\eta_0 = 0.016$ (the learning rates for other benchmarks are tuned, see Appendix III), batch size of 1024 with weight decay $1e^{-5}$. SAM is trained with 5k steps, and Entropy-SGD and SGD-SALR are trained with 2k steps. For ImageNet, we train SAM with ten epochs and train Entropy-SGD/SGD-SALR with four epochs. The initial learning rate is set to be 0.1. Results are reported in Table VI.

As shown in Table VI, Entropy-SGD, SAM, and SGD-SALR can consistently improve the performance of SGD on several high-performance fine-tuning tasks. This further demonstrates the advantage of sharpness-aware algorithm.

D. Additional Analysis

Based on all aforementioned tables, we can obtain several important insights. First, methods adopting SALR show superior performance over their benchmarks. The increase in classification accuracy (or decrease in perplexity) is consistent across various datasets and a range of network structures. We also observed that SGD-SALR tends to outperform other SALR-based methods in most settings while achieving comparable results in others. Second, and more interestingly, this superior performance is achieved with five times fewer epochs compared to SGD, Adam, and SWA. The caveat however is that SALR, Entropy, and SAM, respectively, require $(n_1 + n_2)/c = 5$, $L_a = 5$ and two times more gradient calls at each iteration, hence making the total computational needs the same as Adam, SGD, and SWA. Third and as shown in Table II, it is clear that SALR drives solutions to significantly flatter regions. This highlights the effectiveness



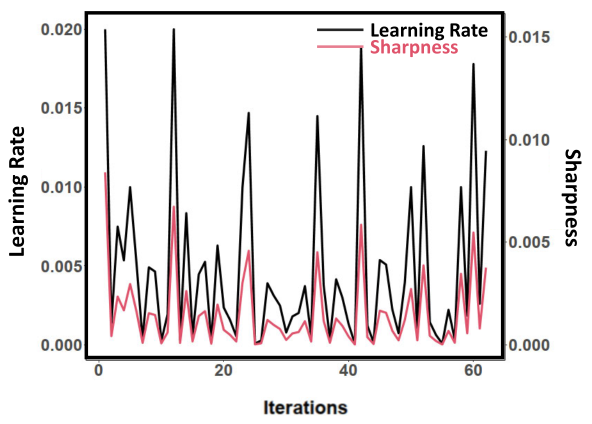


Fig. 3. (Left) All-CNN-BN: change of testing errors over epochs. (Right) Change of sharpness $\hat{S}_k$ and learning rate over iterations.

TABLE VI
TOP-1 TESTING ACCURACY FOR FINETUNING TASKS

EffNet-b7	**SGD**	**Entropy-SGD**	**SAM**	**SGD-SALR**
Flowers	98.83 (0.03)	99.01 (0.02)	**99.37** (0.02)	99.35 (0.01)
CIFAR-100	92.31 (0.06)	92.49 (0.05)	92.56 (0.06)	**92.60** (0.05)
Birdsnap	85.71 (0.17)	86.10 (0.15)	86.36 (0.18)	**86.38** (0.12)
ImageNet	84.69 (0.02)	84.88 (0.03)	84.86 (0.02)	**84.93** (0.02)

EffNet-L2	**SGD**	**Entropy-SGD**	**SAM**	**SGD-SALR**
FGVC-Aircraft	94.21 (0.07)	94.74 (0.05)	95.18 (0.06)	**95.41** (0.05)
Food101	96.00 (0.03)	**96.30** (0.02)	96.20 (0.01)	96.24 (0.01)
Birdsnap	89.70 (0.17)	89.64 (0.13)	**90.07** (0.14)	90.05 (0.11)
ImageNet	88.20 (0.05)	88.33 (0.03)	88.62 (0.03)	**88.75** (0.05)

TABLE VII
SENSITIVITY ANALYSIS ON γ

γ	5×10^{-4}	1×10^{-3}	2×10^{-3}	5×10^{-3}	0.01	0.02
Accuracy	85.63 (0.08)	85.69 (0.10)	85.45 (0.07)	85.72 (0.03)	85.68 (0.04)	85.27 (0.05)
Final Sharpness (10^{-3})	4.87 (0.29)	4.22 (0.31)	5.15 (0.28)	3.89 (0.18)	4.34 (0.17)	5.66 (0.20)

of dynamically adjusting learning rates based on the relative sharpness of the current iterate. To further demonstrate the advantage of SALR, we plot the testing error curves for SGD, Entropy-SGD, Entropy-SALR, and SALR in Fig. 3 (left). Interestingly, we can observe a smoother convergence when adopting SALR framework. To verify the behavior of our framework, we plot in Fig. 3 the dynamics of both the learning rate and sharpness when adopting SALR framework. The desired behavior can be clearly seen in this figure which shows a proportional relationship between learning rates and sharpness.

E. Sensitivity Analysis

To further study the effect of γ, we run sensitivity experiments on CIFAR-100 using PyramidNet (five repetitions for each γ). The parameter γ plays a role similar to ϵ in Definition 1. More specifically, reducing this parameter limits the set of reachable points. This implies decreasing the ball radius. In practice, the most reasonable approach is to first specify the ball radius ϵ, choose γ that is of the same order of ϵ and run gradient methods. As shown in Table VII, the impact of γ is rather marginal on the final accuracy if specified based on our recommendation.

VII. CONCLUSION

In this article, we introduce SALR scheduler that aims to recover flat minima. To demonstrate the effectiveness of SALR, we apply our framework over a wide range of network structures for image classification, text prediction, and finetuning tasks. Our empirical results indicate improved generalization performance when compared to SGD and many other benchmark optimization methods.

Designing learning rate updating mechanisms that utilize the structure of the underlying landscape can potentially improve training deep models. Our research is one step forward in that direction. Potential applications of our updating mechanism can be found in BDL [60], [61] and multimodal MCMC inference. Through SALR, the cyclical schedule can be set by exploiting sharpness information. In lieu of this, SALR may also find use in multimodal MCMC inference or defining stoppage criteria. A direct extension of our work can be seen in studying quasi-Newton approximations of the sharpness measure. Such methods can potentially provide more accurate approximations.

Besides that, and while recent research has shown the advantages of flat solutions, some papers show that sharp minimizers can also generalize well due to the lack of invariance of deep neural networks under transformation, permutation or other operations [12], [45], [62]. Those papers further hint that sharpness is a relative measure that depends on the overall landscape of the loss surface. Fortunately, SALR seeks flat solutions by considering the relative magnitude of historical sharpness information. Along this line, we see potential in investigating how perturbations to the loss surface can induce invariance and help algorithms more effectively recover solutions with good generalization power.

References

[1] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," in *Proc. 33rd Int. Conf. Mach. Learn.*, 2016, pp. 1050–1059.

[2] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

[3] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *J. Big Data*, vol. 6, no. 1, p. 60, 2019.

[4] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–19.

[5] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–15.

[6] I. Radosavovic, R. P. Kosaraju, R. Girshick, K. He, and P. Dollár, "Designing network design spaces," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2020, pp. 10428–10436.

[7] T. Garipov, P. Izmailov, D. Podoprikhin, D. P. Vetrov, and A. G. Wilson, "Loss surfaces, mode connectivity, and fast ensembling of DNNs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 8789–8798.

[8] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson, "Averaging weights leads to wider optima and better generalization," in *Uncertainty in Artificial Intelligence*. Arlington, VA, USA: AUAI Press, 2018.

[9] S. Hochreiter and J. Schmidhuber, "Flat minima," *Neural Comput.*, vol. 9, no. 1, pp. 1–42, 1997.

[10] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–16.

[11] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural Networks: Tricks of the Trade*. Berlin, Germany: Springer-Verlag, 2012, pp. 9–50.

[12] L. Dinh, R. Pascanu, S. Bengio, and Y. Bengio, "Sharp minima can generalize for deep nets," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 1019–1028.

[13] R. Kleinberg, Y. Li, and Y. Yuan, "An alternative view: When does SGD escape local minima?" in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 2698–2707.

[14] F. He, T. Liu, and D. Tao, "Why ResNet works? Residuals generalize," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 12, pp. 5349–5362, Dec. 2020.

[15] B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srebro, "Exploring generalization in deep learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5947–5956.

[16] P. Goyal et al., "Accurate, large minibatch SGD: Training imagenet in 1 hour," 2017, *arXiv:1706.02677*.

[17] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, "Visualizing the loss landscape of neural nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 6389–6399.

[18] P. Chaudhari et al., "Entropy-SGD: Biasing gradient descent into wide valleys," *J. Stat. Mech., Theory Exp.*, vol. 2019, no. 12, 2019, Art. no. 124018.

[19] P. Foret, A. Kleiner, H. Mobahi, and B. Neyshabur, "Sharpness-aware minimization for efficiently improving generalization," in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 1–20.

[20] Z. Allen-Zhu, Y. Li, and Z. Song, "A convergence theory for deep learning via over-parameterization," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 242–252.

[21] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, no. 7, pp. 2121–2159, 2011.

[22] L. Bottou and Y. Le Cun, "On-line learning for very large data sets," *Appl. Stochastic Models Bus. Ind.*, vol. 21, no. 2, pp. 137–151, 2005.

[23] L. Bottou and O. Bousquet, "The tradeoffs of large scale learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 161–168.

[24] M. Hardt, B. Recht, and Y. Singer, "Train faster, generalize better: Stability of stochastic gradient descent," in *Proc. 33rd Int. Conf. Mach. Learn.*, 2016, pp. 1225–1234.

[25] A. Gonen and S. Shalev-Shwartz, "Fast rates for empirical risk minimization of strict saddle problems," in *Proc. Conf. Learn. Theory*, 2017, pp. 1043–1063.

[26] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Rev.*, vol. 60, no. 2, pp. 223–311, 2018.

[27] O. Bousquet and A. Elisseeff, "Stability and generalization," *J. Mach. Learn. Res.*, vol. 2, pp. 499–526, Mar. 2002.

[28] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun, "The loss surfaces of multilayer networks," in *Proc. 18th Int. Conf. Artif. Intell. Statist.*, 2015, pp. 192–204.

[29] K. Kawaguchi, "Deep learning without poor local minima," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 586–594.

[30] T. Poggio, H. Mhaskar, L. Rosasco, B. Miranda, and Q. Liao, "Why and when can deep-but not shallow-networks avoid the curse of dimensionality: A review," *Int. J. Automat. Comput.*, vol. 14, no. 5, pp. 503–519, 2017.

[31] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*. Cambridge, MA, USA: MIT Press, 2018.

[32] M. Mohri and A. Rostamizadeh, "Rademacher complexity bounds for non-I.I.D processes," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 1097–1104.

[33] E. D. Sontag, "VC dimension of neural networks," *NATO ASI F Comput. Syst. Sci.*, vol. 168, pp. 69–96, Jan. 1998.

[34] G. Karolina Dziugaite and D. M. Roy, "Computing nonvacuous generalization bounds for deep (Stochastic) neural networks with many more parameters than training data," 2017, *arXiv:1703.11008*.

[35] B. Neyshabur, S. Bhojanapalli, and N. Srebro, "A PAC-Bayesian approach to spectrally-normalized margin bounds for neural networks," 2017, *arXiv:1707.09564*.

[36] H. Wang, N. Shirish Keskar, C. Xiong, and R. Socher, "Identifying generalization properties in neural networks," 2018, *arXiv:1809.07402*.

[37] C. Baldassi, A. Ingrosso, C. Lucibello, L. Saglietti, and R. Zecchina, "Subdominant dense clusters allow for simple learning and high computational performance in neural networks with discrete synapses," *Phys. Rev. Lett.*, vol. 115, no. 12, 2015, Art. no. 128101.

[38] C. Baldassi et al., "Unreasonable effectiveness of learning neural networks: From accessible states and robust ensembles to basic algorithmic schemes," *Proc. Nat. Acad. Sci. USA*, vol. 113, no. 48, pp. E7655–E7662, 2016.

[39] W. J. Maddox, P. Izmailov, T. Garipov, D. P. Vetrov, and A. G. Wilson, "A simple baseline for Bayesian uncertainty in deep learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 13153–13164.

[40] W. Wen et al., "SmoothOut: Smoothing out sharp minima to improve generalization in deep learning," 2018, *arXiv:1805.07898*.

[41] V. Patel, "The impact of local geometry and batch size on stochastic gradient descent for nonconvex problems," 2017, *arXiv:1709.04718*.

[42] C. Chinrungrueng and C. H. Sequin, "Optimal adaptive k-means algorithm with dynamic adjustment of learning rate," *IEEE Trans. Neural Netw.*, vol. 6, no. 1, pp. 157–169, Jan. 1995.

[43] D. Peng, Y. Zhang, Y. Xiang, and H. Zhang, "A globally convergent MC algorithm with an adaptive learning rate," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 2, pp. 359–365, Feb. 2012.

[44] Z. Wang and J. Zhang, "Incremental PID controller-based learning rate scheduler for stochastic gradient descent," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Oct. 26, 2022, doi: 10.1109/TNNLS.2022.3213677.

[45] A. Rangamani, N. H. Nguyen, A. Kumar, D. Phan, S. H. Chin, and T. D. Tran, "A scale invariant flatness measure for deep network minima," 2019, *arXiv:1902.02434*.

[46] N. Shirish Keskar and R. Socher, "Improving generalization performance by switching from Adam to SGD," 2017, *arXiv:1712.07628*.

[47] P. Zhou, J. Feng, C. Ma, C. Xiong, S. Hoi, and E. Weinan, "Towards theoretically understanding why SGD generalizes better than Adam in deep learning," 2020, *arXiv:2010.05627*.

[48] J. Dean et al., "Large scale distributed deep networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1223–1231.

[49] D. Das et al., "Distributed deep learning using synchronous stochastic gradient descent," 2016, *arXiv:1602.06709*.

[50] R. Ge, F. Huang, C. Jin, and Y. Yuan, "Escaping from saddle points—Online stochastic gradient for tensor decomposition," in *Proc. 28th Conf. Learn. Theory*, 2015, pp. 797–842.

[51] C. Jin, R. Ge, P. Netrapalli, S. M. Kakade, and M. I. Jordan, "How to escape saddle points efficiently," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 1724–1732.

[52] H. Yu, S. Yang, and S. Zhu, "Parallel restarted SGD with faster convergence and less communication: Demystifying why model averaging works for deep learning," in *Proc. AAAI Conf. Artif. Intell.*, 2019, vol. 33, no. 1, pp. 5693–5700.

[53] M. Andriushchenko and N. Flammarion, "Towards understanding sharpness-aware minimization," in *Proc. 39th Int. Conf. Mach. Learn.*, 2022, pp. 639–668.

[54] S. Zagoruyko and N. Komodakis, "Wide residual networks," 2016, *arXiv:1605.07146*.

[55] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.

[56] F. Iandola, M. Moskewicz, S. Karayev, R. Girshick, T. Darrell, and K. Keutzer, "DenseNet: Implementing efficient ConvNet descriptor pyramids," 2014, *arXiv:1404.1869*.

[57] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.

[58] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.

[59] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," 2019, *arXiv:1905.11946*.

[60] L. N. Smith, "Cyclical learning rates for training neural networks," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2017, pp. 464–472.

[61] R. Zhang, C. Li, J. Zhang, C. Chen, and A. Gordon Wilson, "Cyclical stochastic gradient MCMC for Bayesian deep learning," 2019, *arXiv:1902.03932*.

[62] F. Pittorino, A. Ferraro, G. Perugini, C. Feinauer, C. Baldassi, and R. Zecchina, "Deep networks on toroids: Removing symmetries reveals the structure of flat regions in the landscape geometry," 2022, *arXiv:2202.03038*.

Xubo Yue received the B.S. degree in biomedical sciences and applied mathematics from the University of Macau, Macau, in 2016, and the M.S. degree in biostatistics from the University of Michigan, Ann Arbor, MI, USA, in 2018, where he is currently pursuing the Ph.D. degree with the Department of Industrial and Operations Engineering.

His research focuses on Gaussian processes, federated learning, and Bayesian optimization.

Maher Nouiehed received the M.S. degree in operations research engineering from the University of Southern California, Los Angeles, CA, USA, in 2016, under the supervision of Prof. Sheldon Ross and the Ph.D. degree in industrial engineering from the University of Southern California, in 2019, under the supervision of Prof. Meisam Razaviyayn and Prof. Jong Shi-Pang.

He is currently an Assistant Professor with the Department of Industrial Engineering and Management, American University of Beirut (AUB), Beirut, Lebanon. He is interested in studying the computational and theoretical aspects of mathematical optimization problems that arise from machine learning and various fields of science and engineering.

Raed Al Kontar received the bachelor's degree in civil and environmental engineering and mathematics from the American University of Beirut, Beirut, Lebanon, in 2014, and the master's degree in statistics and the Ph.D. degree in industrial and systems engineering from the University of Wisconsin–Madison, Madison, WI, USA, in 2017 and 2018, respectively.

He is currently an Assistant Professor with the Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI, USA, and an Affiliate with the Michigan Institute for Data Science, University of Michigan. His research focuses on distributed and federated probabilistic modeling. His research is currently supported by the National Science Foundation (NSF) and the National Institutes of Health (NIH).

Dr. Al Kontar received the NSF CAREER Award in 2022.