Langevin algorithms for very deep Neural Networks with application to image classification

Pierre Bras

Langevin algorithms for very deep Neural Networks with application to image classification

Pierre Bras*
Laboratoire de Probabilités, Statistique et Modélisation
Sorbonne Université
pierre.bras@sorbonne-universite.fr

Abstract

Training a very deep neural network is a challenging task, as the deeper a neural network is, the more non-linear it is. We compare the performances of various preconditioned Langevin algorithms with their non-Langevin counterparts for the training of neural networks of increasing depth. For shallow neural networks, Langevin algorithms do not lead to any improvement, however the deeper the network is and the greater are the gains provided by Langevin algorithms. Adding noise to the gradient descent allows to escape from local traps, which are more frequent for very deep neural networks. Following this heuristic we introduce a new Langevin algorithm called Layer Langevin, which consists in adding Langevin noise only to the weights associated to the deepest layers. We then prove the benefits of Langevin and Layer Langevin algorithms for the training of popular deep residual architectures for image classification.

1 Introduction

Langevin algorithms are widely used for the training of neural networks in a Bayesian setting [WT11, VBB+20]. Adding a small exogenous noise adds regularization to the training and allows to quantify the degree of uncertainty on the parameters. In this paper, we consider Langevin algorithms directly used for stochastic optimization of neural networks in a non-Bayesian setting and compare their performances with non-Langevin stochastic gradient algorithms. As it was noted in [NVL+15, Ani19], adding gradient noise can in fact improve the learning. Similarly, noisy activation functions [GMDB16, SLH+19] may yield better learning for very deep neural networks. Indeed, the noise provides regularization and allows to escape from traps for the gradient descent such as local minima and saddle points [DPG+14]. Moreover, the deeper the neural network is, the more non-linear it is, thus increasing the number of such traps. Non-convex optimization through Langevin algorithms shares heuristics with simulated annealing which consists in sampling with respect to a Gibbs measure where the noise parameter gradually decreases to zero [vLA87, BP21].

Many advances in supervised learning were made possible using very deep neural networks, which are able to tackle much more difficult problems than shallow ones [KSH12, MPCB14, LBH15], in particular as it comes to image classification [SLJ+15, SZ15, HZRS16, HLVDMW17]. Still, deep neural networks which consist in a succession of dense layers are considerably more difficult to train [GB10, DPG+14] and may run into vanishing gradient problems [Hoc91, Han18]. Without proper adaptation or training, they show poor performance. To cope with this issue, highway networks [SGS15] and residual networks [HZRS16] were introduced. Their many successive layers behaves either as a dense layer or as the identity function, allowing the gradient information to propagate trough the successive layers.

*https://www.lpsm.paris/pageperso/brasp/

Preprint. Under review.

We compare the benefits of preconditioned Langevin algorithms [LCCC16] for various architectures and depths of neural networks and we proceed to side-to-side comparison of Langevin algorithms with their respective non-Langevin counterparts. The purpose of our experiments is to compare different methods on the same model architecture, not to achieve state-of-the-art results. For shallow networks, there is no benefit in using Langevin algorithms as it only adds noise to the gradient descent and brings a less accurate estimation of the minimum. However, we observe that the deeper the network is, the greater are the gains provided by Langevin algorithms.

Since the most important non-linearities of the network are contained in the deepest layers, we introduce a new optimization method that we call Layer Langevin algorithm, which consists in training the network by adding Langevin noise only to the training of some layers and not to the other layers. In particular, we choose the Langevin layers to be the k first (deepest) layers for some integer k. We then highlight the possibilities of training acceleration using Langevin and Layer Langevin methods on deep residual networks [HZRS16] for image classification.

Our code for the numerical experiments is available at `https://github.com/Bras-P/deep-layer-langevin`. It includes in particular ready-to-use Langevin optimizers and Layer Langevin optimizers as instances of the TensorFlow `Optimizer` base class and a demonstration notebook.

2 Very deep neural networks

Training of very deep neural networks is a significantly more challenging task than for shallow networks [GB10, DPG+14]. Let us write the output of a neural network with K layers and with weights $\theta = (\theta^1, \dots, \theta^K)$ as

$$\psi_\theta(x) = \varphi^K_{\theta^K} \circ \cdots \circ \varphi^1_{\theta^1}(x), \tag{1}$$

where $\varphi^1, \dots, \varphi^K$ are activation function and where $\varphi^k_{\theta^k} : x \mapsto \varphi^k(\theta^k \cdot x)$ at every unit. Denoting

$$\Phi_k(x) := \varphi^k_{\theta^k} \circ \cdots \circ \varphi^1_{\theta^1}(x) \tag{2}$$

for $1 \leq k \leq K$ and $\Phi_0(x) := x$, then the gradient reads for $1 \leq k \leq K$:

$$\nabla_{\theta^k} \psi_\theta(x) = \left(\nabla_{\theta^K} \varphi_{\theta^K} \circ \Phi_{K-1}(x)\right) \cdot \cdots \cdot \left(\nabla_{\theta^k} \varphi_{\theta^k} \circ \Phi_{k-1}(x)\right). \tag{3}$$

Thus heuristically, the deeper the layer is, the more the gradient with respect to the parameters of this layer has annealing points, since more factors appear in (3), hinting that deep layers show more non-linearities and local traps.

3 Langevin algorithms for the training of deep neural networks

3.1 Experimental setting

In our experiments we use the following datasets. The MNIST dataset [LBBH98] is composed of 28×28 grayscale images of handwritten digits (from 0 to 9). 60.000 images are used for training and 10.000 images are used for test. The CIFAR-10 and the CIFAR-100 datasets [KH09] consist in RGB images of size 32×32 belonging to 10 and 100 different classes respectively. For both datasets 50.000 images are used for training and 10.000 images are used for test.

The neural networks are trained using preconditioned Langevin algorithms with per-dimension adaptive stepsize [LCCC16] with different choices of preconditioner. That is, for a preconditioner rule (P_n) the Langevin update reads

$$g_{n+1} = \nabla_\theta V(\theta_n; \mathcal{D}_{n+1}) \tag{4}$$

$$\theta_{n+1} = \theta_n - \gamma_{n+1} P_{n+1} \cdot g_{n+1} + \sigma \sqrt{\gamma_{n+1}} \mathcal{N}(0, P_{n+1}), \tag{5}$$

where $\sigma \in (0, \infty)$ controls the amount of injected noise, (γ_n) is the non-increasing learning rate sequence, V denotes the objective function and where $\nabla_\theta V(\theta_n; \mathcal{D}_n)$ stands for the mean gradient computed on a subset $\mathcal{D}_n$ of the dataset. The corresponding preconditioned non-Langevin algorithm follows the same update as in (5) without Gaussian noise. In our experiments we use the RMSprop [DHS11, LCCC16], the Adam [KB15] and the Adadelta [Zei12] preconditioners and we call the Langevin version of these algorithms as L-RMSprop, L-Adam and L-Adadelta respectively. The

preconditioner rules are given in Algorithms 1, 2, 3 respectively. Note that depending on the algorithm version, in the update (5) the gradient g_{n+1} can be replaced by an averaged gradient over the past iterations as this in the case in Adam (Algorithm 2) i.e. momentum gradient is used. While comparing some preconditioned method with its Langevin counterpart, we ensure that both training procedures start with the same initial weights.

Algorithm 1 RMSprop update

Parameters: $\alpha, \lambda > 0$

$\mathrm{MS}_{n+1} = \alpha\,\mathrm{MS}_n + (1-\alpha) g_{n+1} \odot g_{n+1}$

$P_{n+1} = \mathrm{diag}\left(\mathbb{1} \oslash \left(\lambda \mathbb{1} + \sqrt{\mathrm{MS}_{n+1}}\right)\right)$

$\theta_{n+1} = \theta_n - \gamma_{n+1} P_{n+1} \cdot g_{n+1}$

Algorithm 2 Adam update

Parameters: $\beta_1, \beta_2, \lambda > 0$

$M_{n+1} = \beta_1 M_n + (1-\beta_1) g_{n+1}$

$\mathrm{MS}_{n+1} = \beta_2\,\mathrm{MS}_n + (1-\beta_2) g_{n+1} \odot g_{n+1}$

$\widehat{M}_{n+1} = M_{n+1} / (1-\beta_1^{n+1})$

$\widehat{\mathrm{MS}}_{n+1} = \mathrm{MS}_{n+1} / (1-\beta_2^{n+1})$

$P_{n+1} = \mathrm{diag}\left(\mathbb{1} \oslash \left(\lambda \mathbb{1} + \sqrt{\widehat{\mathrm{MS}}_{n+1}}\right)\right)$

$\theta_{n+1} = \theta_n - \gamma_{n+1} P_{n+1} \cdot \widehat{M}_{n+1}.$

Algorithm 3 Adadelta update

Parameters: $\beta_1, \beta_2, \lambda > 0$

$\mathrm{MS}_{n+1} = \beta_1\,\mathrm{MS}_n + (1-\beta_1) g_{n+1} \odot g_{n+1}$

$P_{n+1} = \mathrm{diag}\left(\widehat{MS}_n + \lambda \mathbb{1} \oslash \left(\lambda \mathbb{1} + \sqrt{\widehat{\mathrm{MS}}_n}\right)\right)$

$\theta_{n+1} = \theta_n - \gamma_{n+1} P_{n+1} \cdot g_{n+1}.$

$\widehat{MS}_{n+1} = \beta_2\,\mathrm{MS}_n + (1-\beta_2)(\theta_{n+1} - \theta_n) \odot (\theta_{n+1} - \theta_n).$

3.2 Plain and convolutional networks

We first train fully connected feedforward neural networks on the MNIST dataset. The networks are composed of 3, 20, 30 and 40 hidden dense layers respectively with 64 units each and with ReLU activation, followed by one dense output layer. The results are given in Figure 1. We observe that for shallow neural networks, Langevin algorithms do not outperform their respective non-Langevin counterparts; they add noise to the gradient descent thus giving a less accurate estimate of the minimum value. In particular and as noted in the footnote in [MO17], we could not reproduce the good results from [LCCC16] for plain networks with two hidden layers. However, the deeper the network is, the greater the gains induced by Langevin algorithms compared with their respective non-Langevin counterparts are. We also display the value of the loss function on the training set in order to highlight that the better performances of the Langevin algorithms are not due to some overfitting effect. Langevin algorithms indeed show improvements on 20-layer deep networks; beyond 30-layer deep networks, the gains are significant. The training of 40-layer deep networks with non-Langevin algorithms may run into the vanishing gradient problem, whereas such problem is avoided by Langevin algorithms. In the latter case of very deep training, preconditioned Langevin algorithms not only add noise preventing the vanishing of the gradient, they also help starting up the training in the right directions. To obtain better results with Langevin algorithms, we recommend using a small coefficient σ, empirically ranging from $1\,\mathrm{e}{-3}$ to $5\,\mathrm{e}{-5}$.

We then perform simulations in a similar setup on convolutional architectures that are more adapted to image recognition [JKRL09] followed by a large number of hidden dense layers. More specifically, we train neural networks composed of two convolutional layers with 4×4 kernel size and 32 channels for each; 2×2 max-pooling is used after each convolutional layer. These layers are followed by respectively 10 and 30 hidden dense layers with 64 units each and by one dense output layer. Since the images in the CIFAR-10 dataset do not have a good resolution, we cannot expect a very high accuracy on the test set. Instead, we focus on comparing different algorithms on the same model architecture. The results are given in Figure 2 and we make similar observations: Langevin algorithms show improvements with 10 hidden dense layers and for 30 dense layers, non-Langevin algorithms run into vanishing gradient issues which is not the case for Langevin algorithms.

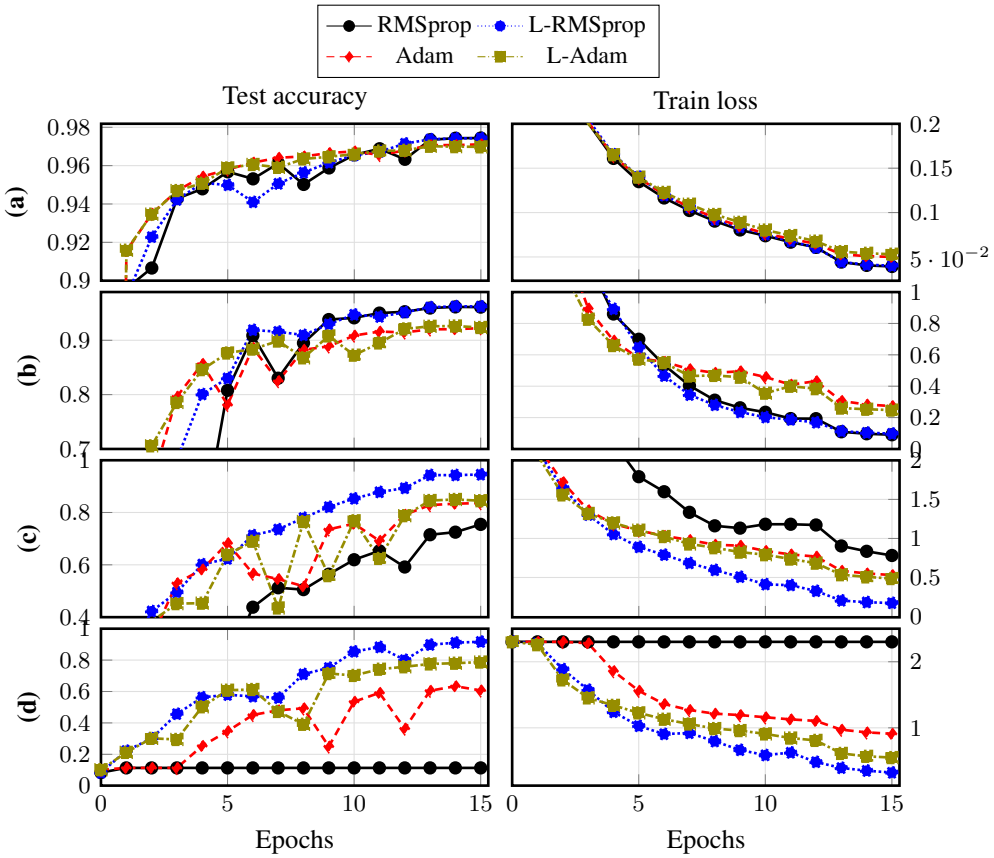

Figure 1: Training of neural networks of various depths on the MNIST dataset using Langevin algorithms compared with their non-langevin counterparts. (a): 3 hidden layers, (b): 20 hidden layers, (c): 30 hidden layers, (d): 40 hidden layers. The batch size is 512. The schedules are $\gamma_n = 1\,\mathrm{e}-3$ and $\sigma = 5\,\mathrm{e}-4$ for epochs 1 to 12 and $\gamma_n = 1e-4$ and $\sigma = 0$ beyond.

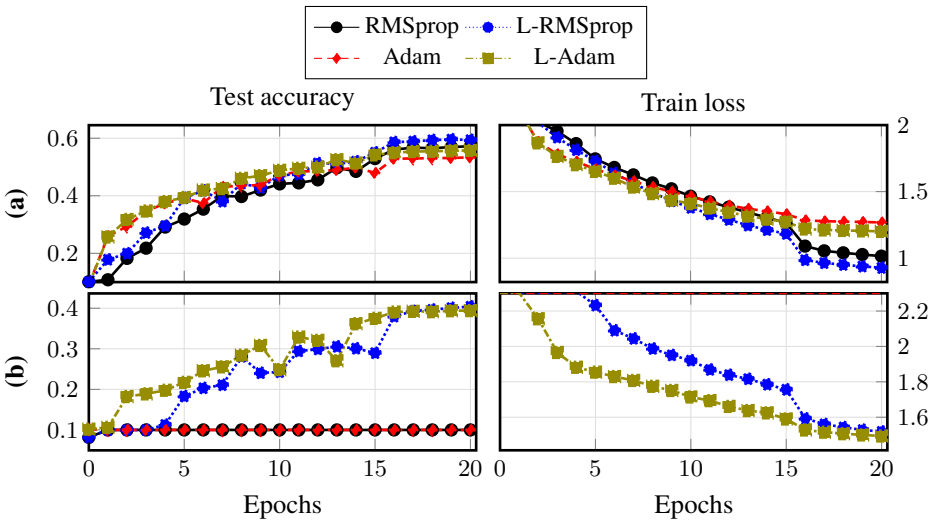

Figure 2: Training of convolutional neural networks on the CIFAR-10 dataset. (a): 10 hidden dense layers, (b): 30 hidden layers. The batch size is 512. The schedules are $\gamma_n = 1\,\mathrm{e}-3$ and $\sigma = 2e-4$ for epochs 1 to 15 and $\gamma_n = 1e-4$ and $\sigma = 0$ beyond.

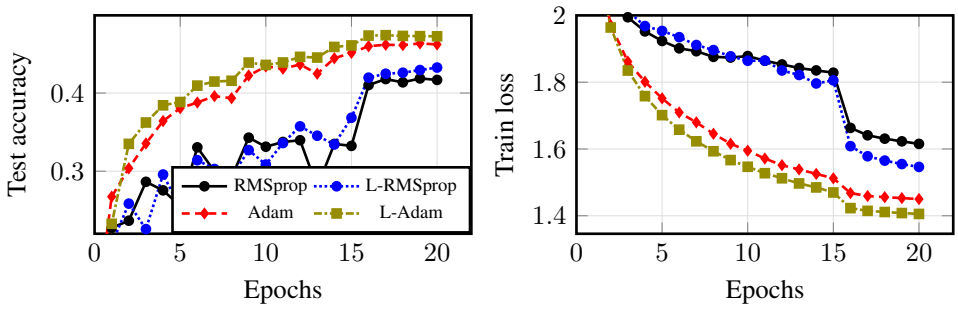

Figure 3: Training of a highway neural network with 80 dense hidden layers. The schedules are $\gamma_n = 1\,\mathrm{e}-3$ and $\sigma = 1e-4$ for epochs 1 to 15 and $\gamma_n = 1e-4$ and $\sigma = 0$ beyond.

3.3 Highway networks

We now perform the same simulations on Highway networks, in a setting very similar to [SGS15]. Comparably to residual networks, the output of a highway layer is a convex combination of the output of a dense layer and the output of a identity layer; the parameter controlling the convex combination is itself trainable. For a layer with weights (θ_D, θ_T), the output reads

$$y = T_{\theta_T}(x) \cdot D_{\theta_D}(x) + (1 - T_{\theta_T}(x)) \cdot x, \tag{6}$$

where T and D are dense layers and where T has sigmoid output.

We observe that Langevin algorithms become effectively faster than non-Langevin algorithms only from a larger depth than for plain networks. In Figure 3 we plot the results for the training of a network composed of 80 dense hidden layers with 64 units each and ReLU activation on the CIFAR-10 dataset, showing the possibilities of acceleration through Langevin algorithms, even in a residual (highway) architecture.

4 Layer Langevin algorithm

We introduce a new Langevin algorithm for stochastic optimization of deep neural networks that we call Layer Langevin algorithm. Choosing a preconditioner rule P, some weights are updated following the Langevin rule while the other weights are updated following the non-Langevin rule. Denoting $\theta_n^{(i)}$ the ith weight at step n, we have for every i:

$$\theta_{n+1}^{(i)} = \theta_n^{(i)} - \gamma_{n+1}[P_{n+1} \cdot g_{n+1}]^{(i)} + \mathbb{1}_{i\in\mathcal{J}}\sigma\sqrt{\gamma_{n+1}}\big[\mathcal{N}(0, P_{n+1})\big]^{(i)}, \tag{7}$$

where $\mathcal{J}$ is a subset of weight indices and where P_n denotes the preconditioner. To simplify the choice of $\mathcal{J}$, we choose $\mathcal{J}$ as the subset of indexes of weights belonging to some layers. However, a finer control over the subset $\mathcal{J}$ remains possible. To implement this method in practice, we simply assign before the training an attribute equals to $\mathbb{1}_{i\in\mathcal{J}}$ to every trainable variable of the network.

We compare the performances of Layer Langevin algorithms with the Adam preconditioner for different choices of the subset of layers. The results are given in Figure 4 for the training of a dense network with 30 hidden dense layers on the MNIST dataset in a setting similar to Figure 1. For some optimizer *Name*, we denote LL-*Name* $p\%$ the corresponding Layer Langevin algorithm where the subset $\mathcal{J}$ is the first $p\%$ layers of the network. We observe that we obtain significant gains in comparison with the vanilla Langevin algorithm and that the best performances are obtained when choosing the subset $\mathcal{J}$ as being the first ℓ layers for some $\ell \in \mathbb{N}$, in particular all the layers of the network except the few last ones.

5 Application to deep architectures for image recognition

We now test the Layer Langevin algorithm to speed up the training of neural networks with very deep architectures that are popular in image recognition. VGG (Visual Geometry Group) network [SZ15] consists in a large number of successive 2D convolutional layers with ReLU activation; the size of

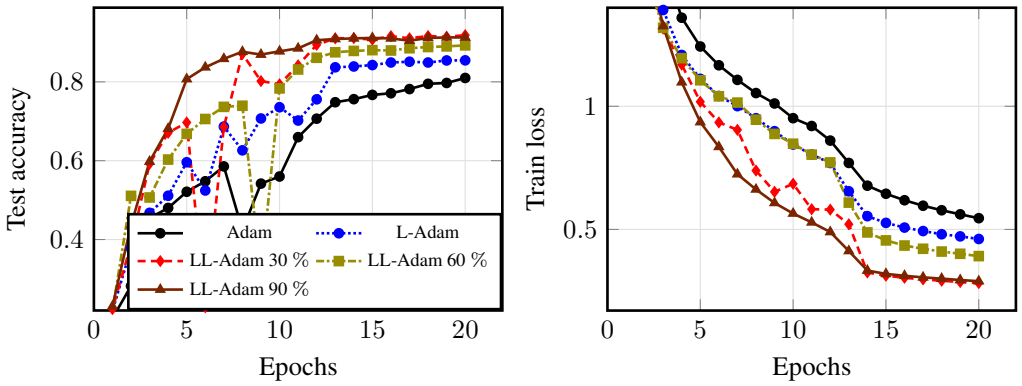

Figure 4: Layer Langevin method comparison on a dense neural network with 30 hidden layers. The schedules are $\gamma_n = 1\,\mathrm{e}-3$ and $\sigma = 5e-4$ for epochs 1 to 13 and $\gamma_n = 1e-4$ and $\sigma = 0$ beyond.

	Adam	LL-Adam	RMSprop	LL-RMSprop	Adadelta	LL-Adadelta
CIFAR-10	76.95 %	77.39 %	84.29 %	85.14 %	75.23 %	75.74 %
CIFAR-100	45.33 %	45.41 %	55.15 %	55.68 %	42.28 %	43.84 %

Table 1: Final test accuracy values obtained in Figures 5.

the image is gradually reduced using 2×2 pooling layers. However its performances are limited by the difficulty of training very deep networks. To cope with this issue, residual network (ResNet) [HZRS16] adds residual connections to the VGG architecture. For H_ℓ some layer composed of convolutions, activations and batch normalizations, the output is $x_{\ell+1} = H_\ell(x_\ell) + x_\ell$ instead of simply $H_\ell(x_\ell)$, so that the residual layer behaves in part as the identity layer. Similarly to highway networks, residual connections improve the flow of gradient inside the network.

We train on the CIFAR-10 dataset a ResNet architecture composed of 2 blocks with 5 residual layers each; each block is followed by a size reduction layer. This architecture is given as ResNet-20 in [HZRS16, Section 4.2]. We apply usual data augmentation to both CIFAR-10 and CIFAR-100 datasets [LXG+15, HZRS16]: 4 pixels are padded on each side and a 32×32 crop is randomly sampled from the padded image or its horizontal flip. The results are given in figure 5. Experiments show that Layer Langevin algorithms (in this case LL-Adam 30%) yield improvements in comparison with non-Langevin methods, even on residual architectures adapted to very deep learning. The train loss is also plotted, showing that the better performances of Layer Langevin is not only due to some overfitting effect.

Acknowledgments and Disclosure of Funding

I would like to thank Gilles Pagès for insightful discussions.

References

[Ani19] Chandrasekaran Anirudh Bhardwaj. Adaptively Preconditioned Stochastic Gradient Langevin Dynamics. *arXiv e-prints*, page arXiv:1906.04324, June 2019.

[BP21] Pierre Bras and Gilles Pagès. Convergence of Langevin-Simulated Annealing algorithms with multiplicative noise. *arXiv e-prints*, page arXiv:2109.11669, September 2021.

[DHS11] John Duchi, Elad Hazan, and Yoram Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

[DPG+14] Yann N. Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and Attacking the Saddle Point Problem in High-Dimensional Non-Convex Optimization. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 2933–2941, Cambridge, MA, USA, 2014. MIT Press.

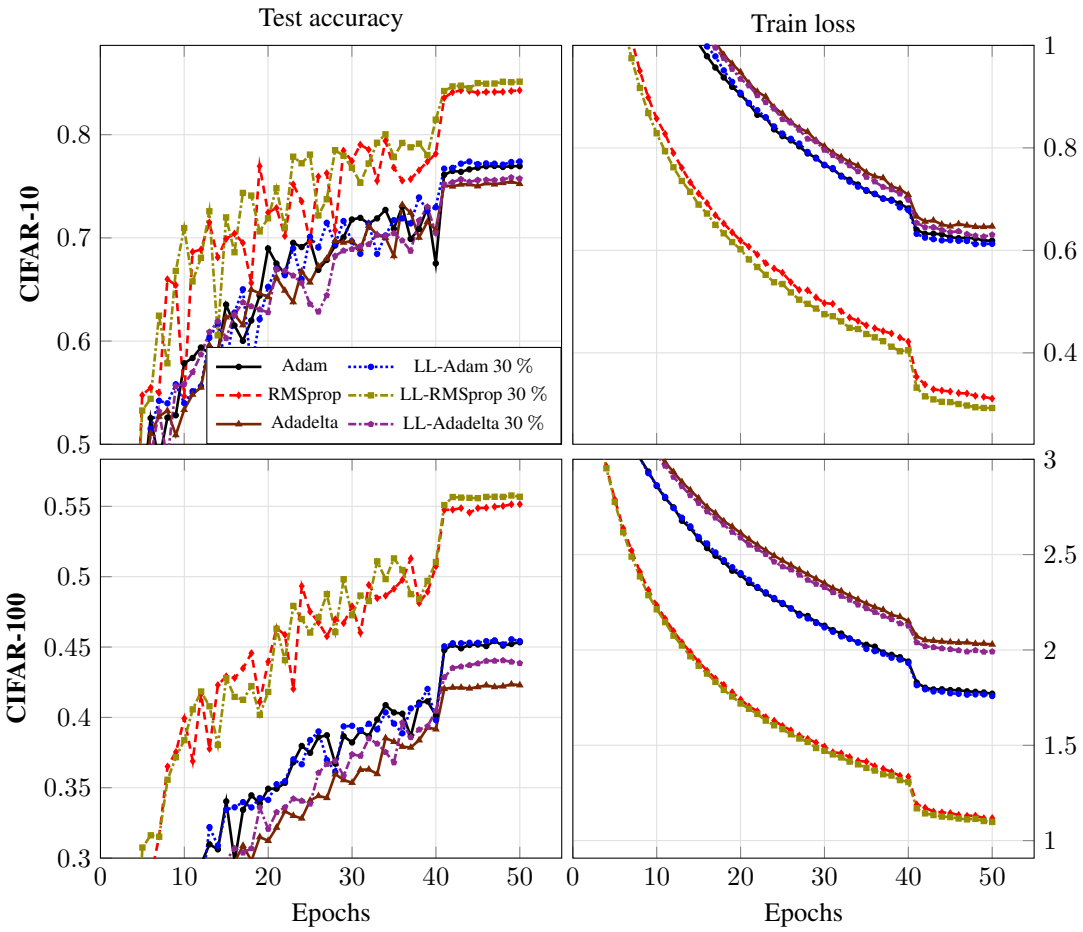

Figure 5: Layer Langevin method comparison for the training of ResNet-20. The initial number of channels is 16. The schedules are $\gamma_n = 1\,\mathrm{e}{-3}$ ($2\,\mathrm{e}{-1}$ for Adadelta) and $\sigma = 5\,\mathrm{e}{-4}$ ($5\,\mathrm{e}{-3}$ for Adadelta) for epochs 1 to 40 γ_n is divided by 10 and σ is set to 0 beyond.

[GB10] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[GMDB16] Caglar Gulcehre, Marcin Moczulski, Misha Denil, and Yoshua Bengio. Noisy activation functions. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 3059–3068. JMLR.org, 2016.

[Han18] Boris Hanin. Which neural net architectures give rise to exploding and vanishing gradients? In *NeurIPS*, pages 580–589, 2018.

[HLVDMW17] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, 2017.

[Hoc91] Sepp Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. diploma thesis, institut für informatik, lehrstuhl prof. brauer, technische universität münchen, 04 1991.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[JKRL09] Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th International Conference on Computer Vision*, pages 2146–2153, 2009.

[KB15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[KH09] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical Report 0, University of Toronto, Toronto, Ontario, 2009.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[LBBH98] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 2015.

[LCCC16] Chunyuan Li, Changyou Chen, David Carlson, and Lawrence Carin. Preconditioned stochastic gradient langevin dynamics for deep neural networks. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 1788–1794. AAAI Press, 2016.

[LXG+15] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-Supervised Nets. In Guy Lebanon and S. V. N. Vishwanathan, editors, *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*, pages 562–570, San Diego, California, USA, 09–12 May 2015. PMLR.

[MO17] Gaétan Marceau-Caron and Yann Ollivier. Natural Langevin Dynamics for Neural Networks. *arXiv e-prints*, page arXiv:1712.01076, December 2017.

[MPCB14] Guido Montúfar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 2924–2932, Cambridge, MA, USA, 2014. MIT Press.

[NVL+15] Arvind Neelakantan, Luke Vilnis, Quoc V. Le, Ilya Sutskever, Lukasz Kaiser, Karol Kurach, and James Martens. Adding Gradient Noise Improves Learning for Very Deep Networks. *arXiv e-prints*, page arXiv:1511.06807, November 2015.

[SGS15] Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[SLH+19] Kumar Shridhar, Joonho Lee, Hideaki Hayashi, Purvanshi Mehta, Brian Kenji Iwana, Seokjun Kang, Seiichi Uchida, Sheraz Ahmed, and Andreas Dengel. ProbAct: A Probabilistic Activation Function for Deep Neural Networks. *arXiv e-prints*, page arXiv:1905.10761, May 2019.

[SLJ+15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[SZ15] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[VBB+20] Laurent Valentin Jospin, Wray Buntine, Farid Boussaid, Hamid Laga, and Mohammed Bennamoun. Hands-on Bayesian Neural Networks – a Tutorial for Deep Learning Users. *arXiv e-prints*, page arXiv:2007.06823, July 2020.

[vLA87] P. J. M. van Laarhoven and E. H. L. Aarts. *Simulated annealing: theory and applications*, volume 37 of *Mathematics and its Applications*. D. Reidel Publishing Co., Dordrecht, 1987.

[WT11] Max Welling and Yee Whye Teh. Bayesian Learning via Stochastic Gradient Langevin Dynamics. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 681–688. Omnipress, 2011.

[Zei12] Matthew D. Zeiler. ADADELTA: An Adaptive Learning Rate Method. *arXiv e-prints*, page arXiv:1212.5701, December 2012.