Training Deep Neural Networks by optimizing over nonlocal paths in hyperparameter space

Vlad Pushkarov
Technion
Israel Institute of Technology
vladpush@icloud.com

Jonathan Efroni
Technion
Israel Institute of Technology
jonathan.efroni@gmail.com

Mykola Maksymenko
SoftServe Inc.
Austin, TX, USA
mmaks@softserveinc.com

Maciej Koch-Janusz
Institute for Theoretical Physics
Swiss Federal Institute of Technology
ETH Zurich
maciejk@ethz.ch

Abstract

Hyperparameter optimization is a practical issue, and an interesting theoretical problem in training of deep architectures. Despite recent advances commonly used methods almost universally involve training multiple and decoupled copies of the model, in effect sampling the hyperparameter space. We show that at a negligible additional cost, results can be improved by sampling *nonlocal paths* instead of points in hyperparameter space. To this end we interpret hyperparameters as controlling the level of correlated noise in training, which can be mapped to an effective temperature. The usually independent instances of the model are coupled and allowed to exchange their hyperparameters throughout the training using the parallel tempering technique of statistical physics. Each simulation corresponds then to a unique path in the joint hyperparameter/model-parameter space. We provide empirical tests, in particular for dropout and learning rate optimization. We observed faster training and improved resistance to overfitting, and a systematic decrease in the absolute validation error, improving over benchmark results.

1 Introduction

The remarkable performance of machine learning models was paid for with an increased model complexity. Part of it is due to architectural choices, but an important aspect is the growth of the number of *hyperparameters*, whose different nature is laid bare by the distinct nature of training, involving a double loop of optimizations. The inner loop finds the best weights – with hyperparameters held fixed – while the outer loop is responsible for finding the optimal value of hyperparameters.

Many HO approaches were developed [Bergstra et al., 2011]; the simplest ones are grid- or random searches, but their weakness is poor scaling and failure to reuse information obtained from previous parameter choices. A family of methods addressing the latter issue arise in the Bayesian framework: the choice of new values to be evaluated is informed by the performance of the model with the previous assignments [Snoek et al., 2012]. High computational cost of evaluation inspires additional strategies, e.g. bandit methods, to leverage cheaper, but cruder results obtained with partial datasets. All of the above HO strategies, however, share a key characteristic: HO is decoupled from that of the weights, and amounts to a more or less clever point sampling of the hyperparameter space, using the validation dataset only. The PBT genetic HO algorithm [Jaderberg et al., 2017] overcomes this issue

Second Workshop on Machine Learning and the Physical Sciences (NeurIPS 2019), Vancouver, Canada.

by optimizing in both weights and hyperparameter space, however, in a greedy way. It cannot be thus guaranteed to explore space in an unbiased way, and may explicitly depend on initial conditions .

Here, we propose a radically different strategy: coupling the usually independent copies of the model at different hyperparameter values *during* the training. The copies are allowed to exchange their hyperparameters as the optimization of the weights progresses, tracing out a *highly nonlocal* trajectory in the product of hyperparameter and weight spaces. The trained model is thus not characterized by a point value of the hyperparameters, but rather by a path or history. The exchange procedure is based on the physical technique of parallel tempering [Swendsen and Wang, 1986], which depends on a mapping of the hyperparameter values to an effective "temperature" of the model we introduce. We demonstrate empirically, that this HO approach – which leverages the training dataset – has desirable properties: the models are more resilient to overfitting, achieve smaller overall errors, and achieve them faster. The method is parallelizable, and the computational overhead over usual grid searches is negligible. We show numerical tests of the approach on neural nets trained on EMNIST and CIFAR-10 datasets. The discussed examples of hyperparameters include the learning rate and drop-out rate, among others.

2 Hyperparameters as controls of smoothness of potential landscape

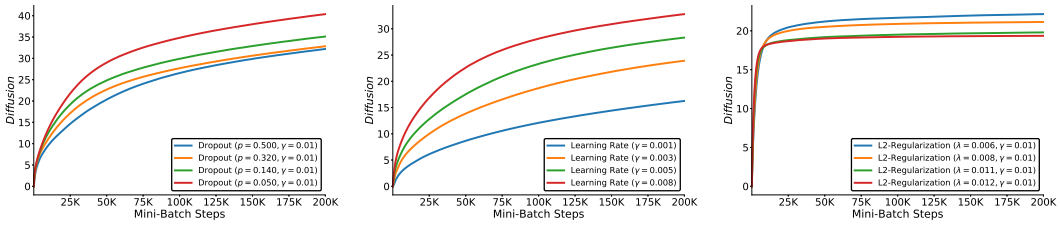

Figure 1: Diffusion curves for model weights (calculated as Euclidean distance between weights vector at initial time t_0 and the vector at epoch t) for replicas of the model at different rate of Dropout, learning rate and L2 regularization. See discussion in the main text for details.

Direct noise injection, whether to the inputs, weights or gradients, aids generalization [Sietsma and Dow, 1991, Neelakantan et al., 2015]. Theoretically, the effect is analogous to introducing particular regularizations [Bishop, 1995, An, 1996] smoothing the cost function. This is also true for non-white noise introduced "indirectly", *e.g* by mini-batches. We will benefit from an intuitive – if not always rigorous – picture relating various kinds of noise to a parameter controlling *effective* smoothness of the potential landscape, similar in spirit to temperature. Common hyperparameters, such as the learning rate, dropout or batch size, can be treated on the same footing. In the case of Langevin noise, the temperature analogy is exact [Seung et al., 1992]: the gradients are corrupted by white noise ξ with a variance $\langle \xi_i(t)\xi_j(t')\rangle = 2T\delta_{ij}\delta_{tt'}$, resulting in a Langevin equation, at long-times converging to a Gibbs probability distribution $P(\mathbf{W}) = \frac{1}{Z}\exp\left(-\beta\mathcal{L}\left(\mathbf{W}\right)\right)$, from which weights $\mathbf{W}$ can be sampled. The inverse temperature β is set by the variance of ξ, and as a multiplicative factor, controls the scale of variation of the potential landscape. Strong noise, i.e. large variance, results in small β which "flattens" the landscape of $\mathcal{L}\left(\mathbf{W}\right)$. Conversely, weak noise results in large β, amplifying potential differences. Crucially, this extends also to cases where the noise is not white: Though, strictly speaking, the variance of the noise is not the physical temperature anymore, it still performs an analogous function: it controls the effective roughness/smoothness of the landscape, as experienced by the random walker subject to such dynamics. Higher variance facilitates diffusion in the high-dimensional space, improving ergodicity (see Fig. 1).

We consider examples of hyperparameters from this perspective, and examine their influence on diffusion curves. In the case of SGD dynamics the noise variance can be computed explicitly [Zhang et al., 2018]. Even though the noise is correlated, the variance scales as inverse batch size: $|N_b|^{-1}$, and thus smaller batch size smoothens the potential. Similarly, dropout regularization increases the variance of neural outputs at training by a factor of inverse dropout retention rate p^{-1} [Hinton et al., 2012], therefore amplifying noise and improving diffusion, as seen in Fig. 1. For the learning rate γ, its magnitude is directly related to the size of discretization step of the Langevin equation, and by dimensional analysis it is analogous to increasing the noise variance (thus temperature) by a factor of γ. The case of L2 is different: an increased L2 diminishes the magnitude of the potential landscape

variations, and so the initial diffusion is faster (see Fig. 1), however at later times the diffusion is suppressed; the plateau value reached is the lower the stronger the regularization. Since, therefore, the models do not diffuse at different rates, L2 does not satisfy the basic requirements of our procedure, and we do not expect systematic improvements.

As generically there are many hyperparameters, and results such as above may not be available, it is imperative to be able to systematically identify whether they are indeed related to effective landscape smoothness. This can be tested exactly as above, with a *diffusion experiment*, i.e. a few inexpensive SGD training runs at different hyperparamater values. Training the model with different such hyperparameters can thus be intuitively thought of as minimizing the objective at different intensities of noise, or, equivalently, in effective landscapes of varying degree of smoothness. We emphasize that this notion of effective smoothness is a joint effect of the geometry of the cost function *and* the dynamics imposed by the training algorithm (with its parameter choices). We will use this observation to construct a hyperparameter parallel tempering procedure.

3 Replica exchange of hyperparameters

Algorithm 1 Training with replica exchange

INPUT: Number of replicas M, Inverse "temperature" (hyperparameters) $\beta = (\beta_1, \beta_2, \ldots, \beta_M)$; Number of steps for initialization ΔN_i; Number of SGD steps between exchanges ΔN_e; Exchange normalization parameter C; Number of steps T

OUTPUT: Weight configurations $\mathbf{W} = (\mathbf{W_1}, \mathbf{W_2}, \ldots, \mathbf{W_M})$ of the replicas,

Initialization: $\forall k \in M$, initialize weights $\mathbf{W}_k$ for each replica and set $t = 0$.
$\forall k \in M$, perform SGD for ΔN_i steps. Update $t \leftarrow t + 1$ at each step.
Repeat:
    $\forall k \in M$, perform SGD for ΔN_e steps to update $\mathbf{W}_k$. Set $t \leftarrow t + 1$ at each step.
    Let $\mathcal{L}_t = (\mathcal{L}(\mathbf{W}_1^t), \mathcal{L}(\mathbf{W}_2^t), ..., \mathcal{L}(\mathbf{W}_k^t)$ be validation losses at time t.
    Randomly select a pair (m, n) of replicas with adjacent temperatures.
    if $\Delta = C(\beta_m - \beta_n)[\mathcal{L}(\mathbf{W}_m) - \mathcal{L}(\mathbf{W}_n)] \leq 0$ **then**
        swap β_m and β_n
    else
        swap β_m and β_n with probability $\exp(-\Delta)$.
    Update α, the acceptance ratio. Finish if $t > T$.

Our approach is to allow the model to change and *exchange* hyperparameters during training, optimizing over paths in the combined weight and hyperparameter space. Inspired by problems in statistical physics we use the parallel tempering (PT) [Swendsen and Wang, 1986]. In PT, multiple Markov Chain Monte-Carlo (MCMC) simulations, or replicas, are run in parallel at different temperatures, defining the levels of uncertainty in the objective function, i.e. the energy. The temperatures are arranged in a ladder on which the states of the replicas are swapped with Metropolis-Hastings acceptance criteria, ensuring the system satisfies detailed balance not only for each chain individually, but also between the chains.The lower temperature chains' ergodicity is radically improved by temporarily performing MC moves at higher temperatures, where the landscape looks flatter. This results in a *non-greedy* and *nonlocal* exploration. Indeed, PT is efficient in systems with broken ergodicity, where configuration space is partitioned into separate regions with low probability for inter-region transitions [Earl and Deem, 2005], e.g. in spin-glass simulations and protein folding [Fukunishi et al., 2002]. It was also applied in training of Boltzmann machines [Desjardins et al., 2010], and ML algorithms used in materials science [Chao et al., 2017, Mazaheri et al., 2019].

The resulting Algorithm 1 is shown in the table above. It accepts M replicas of the system, a vector β of inverse temperatures, number of initialization and SGD steps between exchanges. Each of the M copies is first run for ΔN_e steps, till they achieve relative equilibrium. Later, exchanges are proposed every ΔN_e steps. The constant C in the acceptance ratio is responsible for normalization of the exponent and may be required when the values of hyperparameter are very low, or too high.

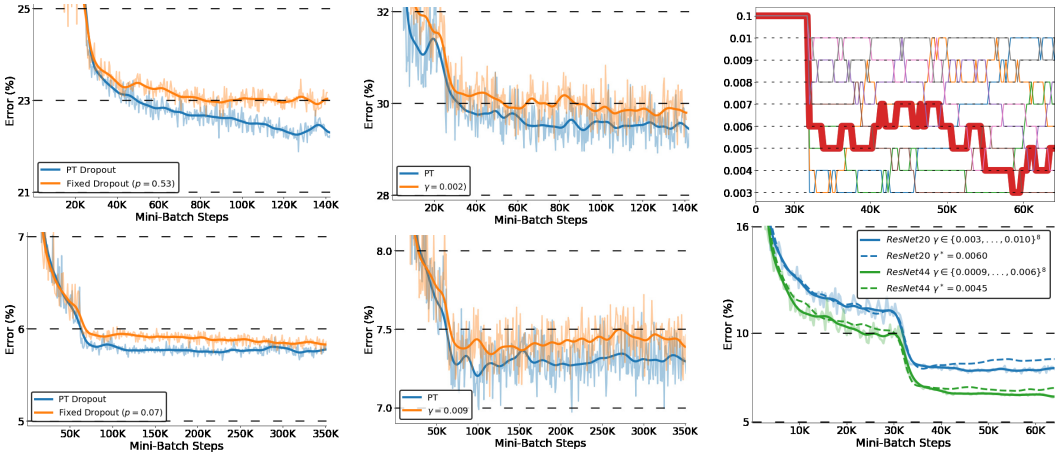

Figure 2: Columns L, C: Test error curves for LeNet architecture for PT training over Dropout p and learning rate γ . In the upper/lower row results for EMNIST / CIFAR-10. The dark lines are running averages. Column R: (Top) Exchanges between replicas differing in learning rates, for the ResNet architecture. Shaded in red: the path in hyperparameter space taken by the best simulation. (Bottom) Test error curves for the ResNets. Dashed lines correspond to the best annealed learning rate, while solid lines show the results with replica exchanges.

4 Empirical results

To validate the approach we conducted a series of experiments. First, on small LeNet-like models, we investigated the effects of various types of hyperparameter-induced "noise" on weight diffusion and tested the idea of hyperparameter replica exchange. We then moved to deep ResNets [He et al., 2016] to verify applicability of our parallel-tempering-based algorithm to large scale models.

The experiments were performed on EMNIST-letters [Cohen et al., 2017] and CIFAR-10 [Krizhevsky and Hinton, 2009] datasets. EMNIST-letters consists of 124800 training images and 20800 testing images of shape 28×28 pixels, with 26 classes. CIFAR-10 has 50000 training images and 10000 testing images, with 10 classes. Each image has shape 32×32 pixels, and 3 channels. The validation splits for both datasets were generated by random sampling of a training dataset taking 10% out. All models were trained with a mini-batch size of 128.

The LeNet-like models are summarized in tables 1 and 2, and the results are shown in Fig. 2. Replicas differing by a "temperature" defined by Dropout and learning rate are evaluated. The performance (classification error) of the best independent replica, with a fixed hyperparameter value, is compared against a PT solution, where replica swaps are introduced. In all of the cases, for both datasets, the best PT path achieves a significantly lower error rate. We also observed increased resilience to overfitting in our small-scale simulations, though this requires more careful study. It is worth noting, that for EMNIST an error rate comparable to the best untempered results is achieved in a much smaller number of epochs. Overall, the training is at least as fast as for the independent simulations.

Layer	Size	Activation
Input	$32 \times 32 \times 1$	-
Convolution	$28 \times 28 \times 6$	tanh
Avg Pooling	$14 \times 14 \times 6$	tanh
Convolution	$10 \times 10 \times 16$	tanh
Avg Pooling	$5 \times 5 \times 16$	tanh
Convolution	$1 \times 1 \times 120$	tanh
Fully Connected	84	tanh
Fully Connected	26	RBF

Table 1: Architecture for EMNIST

Layer	Size	Activation
Input	$32 \times 32 \times 1$	-
Convolution	$28 \times 28 \times 6$	relu
Max Pooling	$14 \times 14 \times 6$	-
Convolution	$10 \times 10 \times 16$	relu
Max Pooling	$5 \times 5 \times 16$	-
Convolution	$1 \times 1 \times 120$	relu
Fully Connected	84	relu
Fully Connected	10	softmax

Table 2: Architecture for CIFAR-10

To showcase the flexibility of the method, and potential for improvements in already efficient and optimized models, we benchmark it on the residual architectures for CIFAR-10, following [He et al., 2016]. We use weight decay of 0.0001 and momentum of 0.9, apply Batch Normalization, as in the original paper, and compare PT learning rates to a fixed one after the initial learning rate annealing. The training begins with learning rate 0.1, annealed once at step 32K; the total amount of training steps is 64K. In Fig. 2R test error curves for ResNet20 and ResNet44 are shown, along with a visualisation of replica hyperparameter exchanges. Introduction of the exchanges consistently reduces the minimal testing error for both architectures. With eight replicas we obtain an improvement of 1.04% for ResNet20, and 1.66% for ResNet44.

We expect larger gains with more replicas. The hyperparameter value of the the best individually trained model (found by grid-search) is included in the parameter ladder for PT, together with a small number of "suboptimal" values. PT improves on all of them, leveraging the replicas to explore the parameter space nonlocally, as seen in the non-trivial replica trajectories throughout the training, in particular the trajectory of the ultimately best one.

5 Conclusions

We introduced a new approach to improve model optimization based on coupling previously independent simulations at different hyperparameter values via parallel tempering (PT). The method is very general: it applies to any hyperparameter which admits interpretation as a temperature-like quantity, in the very weak sense of facilitating weight diffusion during training. This diffusion test is a simple experiment which can be performed at little cost to establish, whether any given parameter is related to *effective* landscape smoothness. We show that this is the case for Dropout, learning rate, but also Batch Normalization. The method is parallelizable, and similar in cost to standard grid search. Experiments performed on LeNet (with CIFAR and EMNIST datasets) and ResNet (with CIFAR) architectures, showed consistently lower test error. In particular, we obtain improvement over the benchmark results for ResNet20 and ResNet44 on the CIFAR dataset. We discuss potential generalizations.

References

Guozhong An. The effects of adding noise during backpropagation training on a generalization performance. *Neural computation*, 8(3):643–674, 1996.

James S. Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2546–2554. Curran Associates, Inc., 2011. URL `http://papers.nips.cc/paper/4443-algorithms-for-hyper-parameter-optimization.pdf`.

Chris M. Bishop. Training with noise is equivalent to tikhonov regularization. *Neural Computation*, 7(1):108–116, 1995. doi: 10.1162/neco.1995.7.1.108. URL `https://doi.org/10.1162/neco.1995.7.1.108`.

Patrick Chao, Tahereh Mazaheri, Bo Sun, Nicholas B. Weingartner, and Zohar Nussinov. The stochastic replica approach to machine learning: Stability and parameter optimization, 2017.

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. Emnist: an extension of mnist to handwritten letters. *arXiv preprint arXiv:1702.05373*, 2017.

Guillaume Desjardins, Aaron Courville, Yoshua Bengio, Pascal Vincent, and Olivier Delalleau. Tempered markov chain monte carlo for training of restricted boltzmann machines. 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 145–152. PMLR, 13–15 May 2010. URL `http://proceedings.mlr.press/v9/desjardins10a.html`.

David J. Earl and Michael W. Deem. Parallel tempering: Theory, applications, and new perspectives. *Phys. Chem. Chem. Phys.*, 7:3910–3916, 2005. doi: 10.1039/B509983H. URL `http://dx.doi.org/10.1039/B509983H`.

Hiroaki Fukunishi, Osamu Watanabe, and Shoji Takada. On the hamiltonian replica exchange method for efficient sampling of biomolecular systems: Application to protein structure prediction. *The Journal of Chemical Physics*, 116(20):9058–9067, 2002. doi: 10.1063/1.1472510. URL `https://doi.org/10.1063/1.1472510`.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012. URL http://arxiv.org/abs/1207.0580.

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M. Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, Chrisantha Fernando, and Koray Kavukcuoglu. Population based training of neural networks. *CoRR*, abs/1711.09846, 2017. URL http://arxiv.org/abs/1711.09846.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

T. Mazaheri, Bo Sun, J. Scher-Zagier, A. S. Thind, D. Magee, P. Ronhovde, T. Lookman, R. Mishra, and Z. Nussinov. Stochastic replica voting machine prediction of stable cubic and double perovskite materials and binary alloys. *Phys. Rev. Materials*, 3:063802, Jun 2019. doi: 10.1103/PhysRevMaterials.3.063802. URL https://link.aps.org/doi/10.1103/PhysRevMaterials.3.063802.

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*, art. arXiv:1511.06807, Nov 2015.

H. S. Seung, H. Sompolinsky, and N. Tishby. Statistical mechanics of learning from examples. *Phys. Rev. A*, 45:6056–6091, Apr 1992. doi: 10.1103/PhysRevA.45.6056. URL https://link.aps.org/doi/10.1103/PhysRevA.45.6056.

Jocelyn Sietsma and Robert J.F. Dow. Creating artificial neural networks that generalize. *Neural Networks*, 4(1):67 – 79, 1991. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(91)90033-2. URL http://www.sciencedirect.com/science/article/pii/0893608091900332.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'12, pages 2951–2959, USA, 2012. Curran Associates Inc. URL http://dl.acm.org/citation.cfm?id=2999325.2999464.

Robert H. Swendsen and Jian-Sheng Wang. Replica monte carlo simulation of spin-glasses. *Phys. Rev. Lett.*, 57:2607–2609, Nov 1986. doi: 10.1103/PhysRevLett.57.2607. URL https://link.aps.org/doi/10.1103/PhysRevLett.57.2607.

Yao Zhang, Andrew M. Saxe, Madhu S. Advani, and Alpha A. Lee. Energy-entropy competition and the effectiveness of stochastic gradient descent in machine learning. *Molecular Physics*, 116 (21-22):3214–3223, 2018. doi: 10.1080/00268976.2018.1483535. URL https://doi.org/10.1080/00268976.2018.1483535.