Parameter Re-Initialization through Cyclical Batch Size Schedules

Norman Mu[*] **Zhewei Yao**[*] **Amir Gholami** **Kurt Keutzer** **Michael W. Mahoney**
{thenorm, zheweiy, amirgh, keutzer and mahoneymw}@berkeley.edu

Abstract

Optimal parameter initialization remains a crucial problem for neural network training. A poor weight initialization may take longer to train and/or converge to sub-optimal solutions. Here, we propose a method of weight re-initialization by repeated annealing and injection of noise in the training process. We implement this through a cyclical batch size schedule motivated by a Bayesian perspective of neural network training. We evaluate our methods through extensive experiments on tasks in language modeling, natural language inference, and image classification. We demonstrate the ability of our method to improve language modeling performance by up to 7.91 perplexity and reduce training iterations by up to 61%, in addition to its flexibility in enabling snapshot ensembling and use with adversarial training.

1 Introduction

Despite many promising empirical results at using stochastic optimization methods to train highly non-convex modern deep neural networks, we still lack theoretically robust practical methods which are able to escape saddle points and/or sub-optimal local minima and converge to parameters that retain high testing performance. This lack of understanding leads to practical training challenges.

Stochastic Gradient Descent (SGD) is currently the de-facto optimization method for training deep neural networks (DNNs). Through extensive hyper-parameter tuning, SGD can avoid poor local optima and achieve good generalization ability. One important hyper-parameter that can significantly affect SGD performance is the weight initialization. For instance, initializing the weights to all zeros or all ones leads to extremely poor performance [24]. Different approaches have been proposed for weight initialization such as Xavier, MSRA, Ortho, LSUV [5, 7, 19, 17]. These are mostly agnostic to the model architecture and the specific learning task.

Our work explores the idea of adapting the weight initialization to the optimization dynamics of the specific learning task at hand. From the Bayesian perspective, improved weight initialization can be viewed as starting with a better prior, which leads to a more accurate posterior and thus better generalization ability. This problem has been explored extensively in Bayesian optimization. For example, in the seminal works [9, 18], an adaptive prior is implemented via Markov Chain Monte Carlo (MCMC) methods. Motivated by these ideas, we incorporate an "adaptive initialization" for neural network training (see section 2 for details), where we use cyclical batch size schedules to control the noise (or temperature) of SGD. As argued in [22], both learning rate and batch size can be used to control the noise of SGD but the latter has an advantage in that it allows more parallelization opportunity [4]. The idea of using batch size to control the noise in a simple cyclical schedule was recently proposed in [11]. Here, we build upon this work by studying different cyclical annealing strategies for a wide range of problems. Additionally, we discuss how this can be combined with a new adversarial regularization scheme recently proposed in [25], as well as prior work [10] in order to obtain ensembles of models at no additional cost. In summary, our contributions are as follows:

[*]Equal contribution

Presented in Systems for ML Workshop in 32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

- We explore different cyclical batch size (CBS) schedules for training neural networks inspired by Bayesian statistics, particularly adaptive MCMC methods. The CBS schedule leads to multiple perplexity improvement (up to 7.91) in language modeling and minor improvements in natural language inference and image classification. Furthermore, we show that CBS schedule can alleviate problems with overfitting and sub-optimal parameter initialization.
- Additionally, CBS schedules require up to $3\times$ fewer SGD iterations due to larger batch sizes, which allows for more parallelization opportunity. This reflects the benefit of cycling the batch size instead of the learning rate as in prior work [21, 10]
- We showcase the flexibility of CBS schedules for use with additional techniques. We propose a simple but effective ensembling method that combines models saved during different cycles at no additional training cost. In addition, we show that CBS schedule can be combined with other approaches such as the recently proposed adversarial regularization [25] to yield further classification accuracy improvement of 0.26%.

Related Work

[5] introduced Xavier initialization, which keeps the variance of input and output of all layers within a similar range in order to prevent vanishing or exploding values in both the forward and backward passes. Building off this idea, [7] explored a new strategy known as MSRA to keep the variance constant for all convolutional layers. [19] proposed an orthogonal initialization (Ortho) to achieve faster convergence, and more recently, [17] combined ideas from previous work and showed that a unit variance orthogonal initialization is beneficial for deep models.

[11, 22, 3] show that the noise of SGD is controlled by the ratio of learning rate to batch size. The authors argued that the SGD algorithm can be derived through Euler-Maruyama discretization of a Stochastic Differential Equation (SDE). The SDE dynamics are governed by a "noise scale" $g \approx \epsilon N/B$ for ϵ the learning rate, N the training dataset size, and B the batch size. They conclude that a higher noise scale prevents SGD from settling into sharper minima. This result supports a prior empirical observation [12] that under certain mild assumptions such as $N \gg B$, the effect of dividing the learning rate by a constant factor is equivalent to that of multiplying the batch size by the same constant factor. In related work, [21] applied this understanding and used batch size as a knob to control the noise, and empirically showed that the baseline performance could be matched. [25] further explored how to use second-order information and adversarial training to control the noise for training large batch size. [14, 15] showed using a statistical mechanics argument that many other hyper-parameters in neural network training, e.g. data quality, can also act as temperature knobs.

2 Methods

The goal of neural network optimization is to solve an empirical risk minimization, with a loss function of the form:

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N} l(x_i, \theta), \tag{1}$$

where θ is the model parameters, X is the training dataset and $l(x, \theta)$ is the loss function. Here $N = |X|$ is the cardinality of the training set. In SGD, a mini-batch, $B \subset \{1, 2, ..., N\}$ is used to compute an (unbiased) gradient, i.e., $g_t = \frac{1}{|B|}\sum_{x\in B} \nabla_\theta l(x, \theta_t)$, and this is typically used to optimize (1) in the form:

$$\theta_{t+1} = \theta_t - \eta_t g_t, \tag{2}$$

where η_t is the learning rate (step size) at iteration t, and commonly annealed during training.

By Bayes' Theorem, given the input data, X, a prior distribution on the model parameters, $P(\theta)$, and a likelihood function, $P(X|\theta)$, the posterior distribution, $P(\theta|X)$, is:

$$P(\theta|X) \propto P(\theta)P(X|\theta). \tag{3}$$

From this Bayesian perspective, the goal of the neural network training is to find the Maximum-A-Posteriori (MAP) point for a given prior distribution. Note that in this context weight initialization and prior distribution are similar, that is a better prior distribution would lead to more informative posterior. In general, it may be difficult to design a better prior given only data and a model architecture. Additionally, the high dimensionality of the NN's parameter space renders various approaches such as adaptive priors intractable (e.g. adaptive MCMC algorithms [18, 9]). Hence, we look into an adaptive weight "re-initialization" strategy. We start with an input prior (weight initialization) and compute an approximate MAP point by annealing the noise in SGD. Once we compute the MAP point, we use it as a new initialization of the neural network weights, and restart the noise annealing schedule. We then iteratively repeat this process through the training process.

One approach to controlling the level of noise in SGD is via the learning rate, which is the approach used in [10, 20]. However, as discussed in [11, 21, 3], the batch size can also be used to control SGD noise. The motivation for this is that larger batch sizes allow for parallel execution which can accelerate training. We implement weight re-initialization through cyclical batch size schedules. The SGD training process is divided into one or more cycles, and in single cycle we gradually increase the batch size to decrease noise. As the noise level of SGD is annealed, θ will approaches a local minima i.e., an approximate MAP point of $P(\theta|X)$. Then at the beginning of the subsequent cycle we drop the batch size back down to the initial value, which increases the noise in SGD and "re-initializes" the neural network parameters using the previous estimate. Several CBS schedules are shown in Fig. 1.

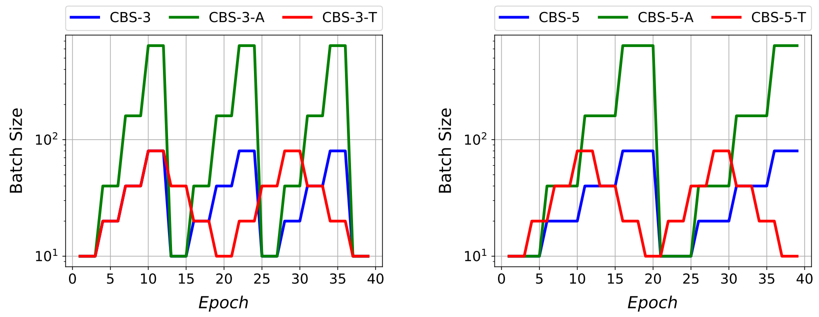

Figure 1: *Illustration of 6 different CBS schedules, with initial batch size of 10; see Appendix A for details.*

3 Results

We perform a variety of experiments across different tasks and neural network architectures in natural language processing as well as image classification. We report our experimental findings on language tasks in section 3.1, and image classification in section 3.2. We illustrate that CBS schedules can alleviate sub-optimal initialization in section 3.3. We follow the baseline training method for each task (for details please see Appendix A). Alongside testing/validation performance, we also report the number of training iterations (lower values are preferred).

3.1 Language Results

Language modeling is a challenging problem due to the complex and long-range interactions between distant words [16]. One hope is that large/deep models might be able to capture these complex interactions, but large models easily overfit on these tasks and exhibit large gaps between training set and testing set performance. CBS schedules effectively help us avoid overfitting, and in addition snapshot ensembling enables even greater performance.

We evaluate a large variety of CBS schedules to positive results as shown in Table 1. Results are measured in perplexity, a standard figure of merit for evaluating the quality of language models by measuring its prediction of the empirical distribution of words (lower perplexity value is better). As we can see, the best performing CBS schedules result in significant improvements in perplexity (up to 7.91) over the baseline schedules and also offer reductions in the number of SGD training iterations (up to 33%). For example, CBS schedules achieve improvement of 7.91 perplexity improvement on WikiText 2 via CBS-1-T and reduce the SGD iterations from 164k to 111k via the CBS-1-A schedule. Notice that almost all CBS schedules outperform the baseline schedule.

Fig. 2 shows the training and testing perplexity of the L2 model on PTB and WikiTest 2 as trained via the baseline schedule along with our best CBS schedule (from Table 1). Notice the cyclical spikes in training and testing perplexity. The peaks occur during decreases in batch size, i.e., increases in noise scale, which could help to escape sub-optimal local minima, and the troughs occur during increases in batch size, i.e., decreases with noise scale.

Table 1: *Testing perplexity and number of parameter updates of L1 and L2 models on Penn Tree Bank (PTB) and WikiText 2 (WT2) datasets. The best perplexity and lowest number of updates are* ***bolded***.

	L1 on PTB		L1 on WT2		L2 on PTB		L2 on WT2	
Schedule	Per.	# Iters	Per.	# Iters	Per.	# Iters	Per.	# Iters
BL[2]	83.13	52k	96.41	116k	79.34	73k	99.69	164k
CBS-10	80.49	49k	94.93	111k	79.37	83k	95.43	187k
CBS-5	80.78	49k	**94.31**	111k	78.61	73k	94.32	164k
CBS-1	81.56	49k	94.52	111k	77.56	69k	**91.78**	156k
CBS-10-A	**80.28**	35k	95.91	79k	81.47	65k	95.28	146k
CBS-5-A	82.03	35k	95.23	79k	79.48	53k	93.63	118k
CBS-1-A	84.41	35k	95.66	79k	81.32	49k	93.19	111k
CBS-10-T	80.49	49k	94.93	111k	79.42	83k	94.39	187k
CBS-5-T	80.94	53k	94.9	120k	78.95	63k	94.68	142k
CBS-1-T	81.82	46k	95.38	104k	**77.39**	65k	93.78	147k

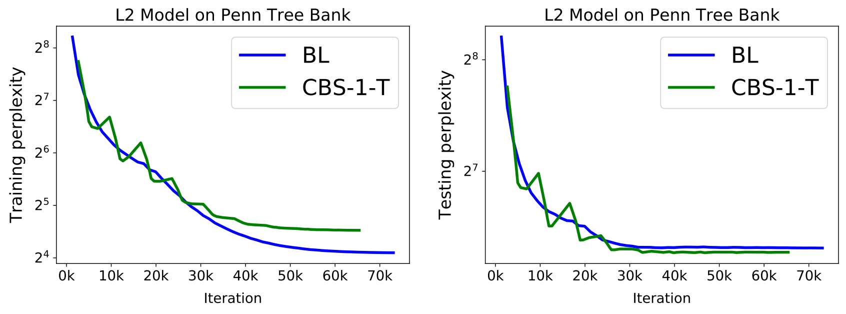

Figure 2: *Training (left) and testing (right) perplexity as a function of iterations for the L2 model on PTB.*

In order to support our claim that CBS schedules are especially useful for counteracting overfitting, we conducted additional language modeling experiments on models L1', L2' with PTB and WT2 which use significantly lower dropout (0.2 and 0.3) than the original L1, L2 models (0.5 and 0.65). Because these models heavily overfit the training data, we report both the final testing perplexity as well as the best testing perplexity achieve during training. As seen in Table 5 (in Appendix B), with L2' CBS yields improvements of a staggering 60.3 on final testing perplexity and 36.2 on best testing perplexity. CBS yields smaller improvements on L1' of 26.0 and 25.3, which are still much larger than the improvement achieved by CBS on L1 and L2.

As mentioned above the goal of every cycle is to get an approximate MAP point. A very interesting idea proposed in [10] is to ensemble these MAP points by saving snapshots of the model at the end of every cycle. We follow that strategy with the only difference that we use a batch size cycle instead of cyclical learning rate proposed in [10] due to higher parallelization opportunities for the former. We perform experiments on snapshot ensembling with the L2 model with the respective best performing CBS schedules on PTB and WikiText 2 (CBS-1-T and CBS-1), as well as the fixed batch size baseline. The CBS ensembles on PTB and WikiText 2 result in test set perplexity of 76.14 and 88.47, outperforming baseline ensembles on both datasets (76.52, 89.99 respectively) and CBS single models (77.39, 91.78 respectively).

To further explore the properties of cyclical batch size schedules, we also evaluate these schedules on natural language inference tasks, as shown in Table 2. In our experiments, CBS schedules do not yield large performance improvements on models like E1 which exhibit smaller disparities between training and testing performance. This is in line with our limitation in that CBS is more effective for models which tend to overfit. On the other hand, we see a large reduction in training iterations by up to 62% which is due to higher effective batch size used in CBS than baseline.

[2][28] reports testing perplexity of 82.7 and 78.4 for L1 and L2 respectively on PTB, which we could not reproduce. The best perplexity and lowest number of updates are **bolded**.

3.2 Image Classification Results

We also test our CBS schedules on Cifar-10 and ImageNet. Table. 3 reports the testing accuracy and the number of training iterations for different models on Cifar-10. We see that the CBS schedules match baseline performance, but the number of training iterations used in CBS schedules is up to 2× fewer.

As seen in Fig. 3, the training curves of CBS schedules also exhibit the aforementioned cyclical spikes both in training loss and testing accuracy. Similarly in the previously discussed language experiments, these spikes correspond to cycles in the CBS schedules and can be thought of as re-initializations of the neural network weights. We observe that CBS achieves similar performance to the baseline.

Table 2: *Validation accuracy and number of parameter updates of E1 on MultiNLI and SNLI datasets. The best accuracy and lowest number of updates are* ***bolded****.*

	MultiNLI		SNLI	
Strategy	Acc.	# Iters	Acc.	# Iters
BL	72.87	123k	**86.86**	172k
CBS-1	**73.17**	64k	86.73	90k
CBS-2	73.07	71k	86.56	99k
CBS-1-A	72.23	**48k**	86.26	**67k**
CBS-2-A	72.04	57k	85.83	80k

Fig. 4 shows the results of ResNet50 on ImageNet. The baseline trains in $450k$ iterations and reaches 76.134% validation accuracy. With CBS, the final validation accuracy is 76.336%, trained in $262k$ parameter updates. CBS outperforms the baseline on both training loss and validation accuracy.

We offer further support for the hypothesis that CBS schedules are more effective for overfitting neural networks with experiments on model C4, which achieves 94.35% training accuracy and 55.55% testing accuracy on Cifar-10. With CBS-15, we see 90.71% training accuracy and 56.44% testing accuracy, which is a larger improvement than that offered by CBS on convolutional models on Cifar-10.

We also explore combining CBS with the recent adversarial regularization proposed by [25]. Combining CBS-15 on C2 with this strategy improves accuracy to 94.82%. This outperforms other schedules shown in Table 3. Applying snapshot ensembling on C3 trained with CBS-15-2 leads to improved accuracy of 93.56% as compared to 92.58%. After ensembling ResNet50 on Imagenet with snapshots from the last two cycles, the performance increases to 76.401% from 75.336%.

Table 3: *Accuracy and number of parameter updates of different models on Cifar-10. The best accuracy and lowest number of iterations are* ***bolded****.*

AlexNet-like (C1)			WResNet (C2)			ResNet18 (C3)		
Strategy	Acc.	# Iters	Strategy	Acc.	# Iters	Strategy	Acc.	# Iters
Baseline	86.94	35k	Baseline	94.53	78k	Baseline	**92.71**	63k
CBS-10-3	86.83	20k	CBS-15	94.46	40k	CBS-10	92.47	**32k**
CBS-15-2	86.87	26k	CBS-10-3	**94.56**	45k	CBS-5-3	92.45	37k
CBS-5-3	**87.03**	20k	CBS-5-3	94.44	45k	CBS-15-2	92.58	48k
CBS-5-3-A	86.75	**15k**	CBS-5-3-A	94.34	**33k**	CBS-15-2-A	92.27	39k

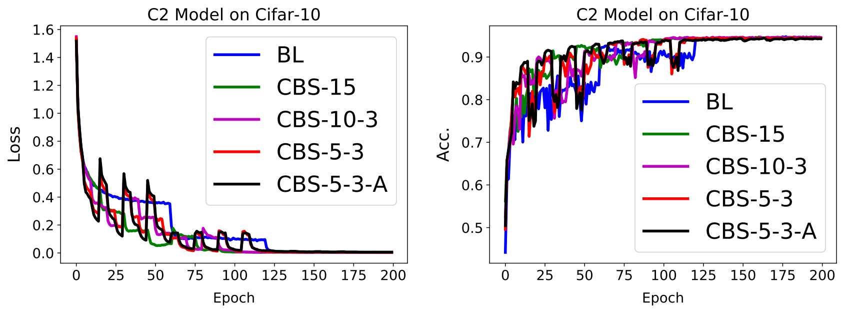

Figure 3: *C2 model (WResNet) on Cifar-10. Training set loss (left), and testing set accuracy (right), evaluated as a function of epochs*

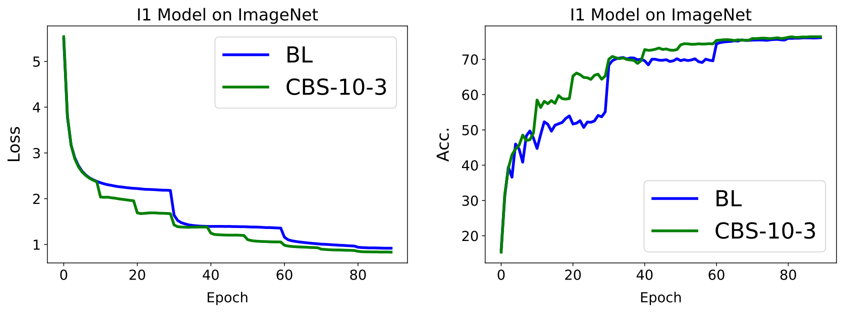

Figure 4: *I1 model (ResNet50) on ImageNet. Training set loss (left), and testing set accuracy (right), evaluated as a function of epochs.*

3.3 Sub-optimal Initialization

Various effective initialization methods [5, 7, 19, 17] have been proposed previously; however, when presented with new architectures and new tasks, initialization still needs to be explored empirically and often the final performance varies greatly with different initializations. In this section, we test if CBS schedules can alleviate the problem of sub-optimal initialization.

We test a Gaussian initialization with mean 0 and standard deviation 0.1 on an AlexNet-like model (C1). The baseline (BL) training follows the same setting as described in Appendix A and achieves final accuracy 84.27%. For CBS, we use cycle width of 10 with 3 steps. In particular, CBS$_1$ denotes a constant learning rate, and achieves final accuracy 85.41%. CBS$_2$ decays the learning rate by a factor of 5 at epoch 75 and achieves final accuracy 84.95%. We keep learning rate high during training because a high noise level helps θ escape sub-optimal local minima. Notice that all CBS methods achieve better generalization performance than the baseline.

4 Conclusions

In this work we explored different cyclical batch size (CBS) schedules for training neural networks. We framed the motivation behind CBS schedules through the lens of Bayesian statistical methods, in particular adaptive MCMC algorithms, which seek out better estimates of the posterior starting with a (poor) prior distribution. In the context of neural network training, this translates to re-initialization of the weights via cycling between large and small batch sizes which control the noise in SGD. We show empirical results which find this cyclical batch size schedule can significantly outperform fixed batch size baselines, especially in networks prone to overfitting or initialized poorly, on the tasks of language modeling, natural language inference, and image classification with LSTMs, CNNs, and ResNets. In our language modeling experiments, we see that a wide variety of CBS schedules outperform the baseline by up to 7.91 perplexity and up to 33% fewer training iterations. For natural language inference and image classification tasks, we observe a reduction in the number of training iterations of up to 61%, which translates directly into reduced runtime. Finally, we demonstrate the flexibility of CBS as a building block for ensembling and adversarial training methods. Ensembling on language modeling yields improvements of up to 11.22 perplexity over the baseline and on image classification, an improvement of up to 1.07% accuracy. Adversarial training in conjunction with CBS gives a bump in image classification accuracy of 0.26%.

Limitations

We believe that it is very important for every work to state its limitations (in general, but in particular in this area). We performed an extensive variety of experiments on different tasks in order to comprehensively test the algorithm. The primary limitation of our work is that cyclical batch size schedules introduce another hyper-parameter that requires manual tuning. We note that this is also true for cyclical learning rate schedules, and hope to address this using second order methods [25] as part of future work. Furthermore, for well initialized models which are not prone to overfitting, single snapshot CBS achieves similar performance to the baseline, although the cyclical ensembling provides a modicum of improvement.

References

[1] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2015.

[2] Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. Enhanced lstm for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1657–1668, 2017.

[3] Aditya Devarakonda, Maxim Naumov, and Michael Garland. Adabatch: Adaptive batch sizes for training deep neural networks. *arXiv preprint arXiv:1712.02029*, 2017.

[4] Amir Gholami, Ariful Azad, Peter Jin, Kurt Keutzer, and Aydin Buluc. Integrated model, batch and domain parallelism in training neural networks. *ACM Symposium on Parallelism in Algorithms and Architectures(SPAA'18)*, 2018.

[5] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.

[6] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples (2014). *arXiv preprint arXiv:1412.6572*, 2014.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034, 2015.

[8] 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.

[9] Zixi Hu, Zhewei Yao, and Jinglai Li. On an adaptive preconditioned crank–nicolson mcmc algorithm for infinite dimensional bayesian inference. *Journal of Computational Physics*, 332:492–503, 2017.

[10] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. Snapshot ensembles: Train 1, get m for free. *arXiv preprint arXiv:1704.00109*, 2017.

[11] Stanis**l**aw Jastrz**k**ebski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. Three factors influencing minima in sgd. *arXiv preprint arXiv:1711.04623*, 2017.

[12] Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv:1404.5997*, 2014.

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

[14] Charles H Martin and Michael W Mahoney. Rethinking generalization requires revisiting old ideas: statistical mechanics approaches and complex learning behavior. *arXiv preprint arXiv:1710.09553*, 2017.

[15] Charles H Martin and Michael W Mahoney. Implicit self-regularization in deep neural networks: Evidence from random matrix theory and implications for learning. *arXiv preprint arXiv:1810.01075*, 2018.

[16] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

[17] Dmytro Mishkin and Jiri Matas. All you need is a good init. *arXiv preprint arXiv:1511.06422*, 2015.

[18] Gareth O Roberts and Jeffrey S Rosenthal. Examples of adaptive mcmc. *Journal of Computational and Graphical Statistics*, 18(2):349–367, 2009.

[19] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

[20] Leslie N Smith. Cyclical learning rates for training neural networks. In *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*, pages 464–472. IEEE, 2017.

[21] Samuel L Smith, Pieter-Jan Kindermans, and Quoc V Le. Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*, 2017.

[22] Samuel L Smith and Quoc V Le. A Bayesian perspective on generalization and Stochastic Gradient Descent. *arXiv preprint arXiv:1710.06451*, 2018.

[23] Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics, 2018.

[24] Peng Xu, Farbod Roosta-Khorasan, and Michael W Mahoney. Second-order optimization for non-convex machine learning: An empirical study. *arXiv preprint arXiv:1708.07827*, 2017.

[25] Zhewei Yao, Amir Gholami, Kurt Keutzer, and Michael Mahoney. Large batch size training of neural networks with adversarial training and second-order information. *arXiv preprint arXiv:1810.01021*, 2018.

[26] Zhewei Yao, Amir Gholami, Qi Lei, Kurt Keutzer, and Michael W Mahoney. Hessian-based analysis of large batch training and robustness to adversaries. *arXiv preprint arXiv:1802.08241*, 2018.

[27] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[28] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

A Training Details

Here we catalogue details regarding all tasks, datasets, models, batch schedules, and other hyper-parameters used in our experiments. In all experiments, we try to copy as many hyper-parameters from the original papers as possible.

Tasks: We train networks to perform the following supervised learning tasks:

- **Image classification.** The network is trained to classify the content of images within a fixed set of object classes.
- **Language modeling.** The network is trained to predict the last token in a sequence of English words.
- **Natural Language Inference.** The network is trained to classify the relationship between pairs of English sentences such as that of entailment, contradiction, or neutral.

Datasets: We train networks on the following datasets.

- **Cifar (image classification).** The two Cifar (i.e., Cifar-10/Cifar-100) datasets [13] contain 50k training images and 10k testing images, and 10/100 label classes.
- **ImageNet (image classification).** The ILSVRC 2012 classification dataset consists of 1000 label classes, with a total of 1.2 million training images and 50,000 validation images. During training, we crop the image to 224×224.
- **PTB (language modeling).** The Penn Tree Bank dataset consists of preprocessed and tokenized sentences from the Wall Street Journal. The training set is 929k words, the validation set 73k words, and test set 82k words. The total vocabulary size is 10k, and all words outside the vocabulary are replaced by a placeholder token.
- **WikiText 2 (language modeling).** The Wikitext 2 dataset is modeled after the Penn Tree Bank dataset and consists of preprocessed and tokenized sentences from Wikipedia. The training set is 2089k words, the validation set 218k words, and the test set 246k words. The total vocabulary size is 33k, and all words outside the vocabulary are replaced by a placeholder token.
- **SNLI (natural language inference).** The SNLI dataset [1] consists of pairs of sentences annotated with one of three labels regarding textual entailment information: contradiction, neutral, or entailment. The training set contains 550k pairs, and the validation set contains 10k pairs.
- **MultiNLI (natural language inference.** The MultiNLI dataset [23] is modeled after the SNLI dataset and contains a training set of 393k pairs and a validation set of 20k pairs.

Model Architecture. We implement the following neural network architectures.

- **C1.** AlexNet-like on Cifar-10 dataset as in [26][C1], trained on the task of image classification. We train for 200 epochs with an initial learning rate 0.02 which we decay by a factor of 5 at epoch 30, 60. In particular, we use initial learning rate 0.05 for cyclic scheduling.
- **C2.** WResNet 16-4 on Cifar-10 dataset [27], trained on the task of image classification. We train for 200 epochs with an initial learning rate 0.1 which we decay by a factor of 5 at epoch 60, 120, and 180.
- **C3.** ResNet20 on Cifar-10 dataset [8]. We train it for 160 epochs with initial learning rate 0.1, and decay a factor of 5 at epoch 80, 120. In particular, we use initial learning rate 0.05 for cyclic scheduling.
- **C4.** MLP3 network from [15]. The network consists of 3 fully connected layers with 512 units each and ReLU activations. As a baseline, we train this network with vanilla SGD for 240 epochs with a batch size of 100 and an initial learning rate of 0.1, which is decayed by a factor of 10 at 150 and 225 epochs.
- **I1.** ResNet50 on ImageNet dataset [8], trained on the task of image classification for 90 epochs with initial learning rate 0.1 which we decay by a factor of 10 at epoch 30, 60 and 80.
- **L1.** Medium Regularized LSTM [28], trained on the task of language modeling. We use 50% dropout on non-recurrent connections and train for 39 epochs with initial learning rate of 20, decaying by a factor of 1.2 every epoch after epoch 6. We set a backpropagation-through-time limit of 35 steps and clip the max gradient norm at 0.25.
- **L2.** Large Regularized LSTM [28], trained on the task of language modeling. We use 65% dropout on non-recurrent connections and train for 55 epochs with initial learning rate of 20, decaying by a factor of 1.15 every epoch after epoch 14. We set a backpropagation-through-time limit of 35 steps and clip the max gradient norm at 0.5.

- **L1', L2'** Identical to L1, L2 except for lower dropout: 0.2, 0.3 respectively. Leads to significant overfitting, evidenced by test perplexity curve in Fig. 5.
- **E1.** ESIM [2]. We train the base ESIM model without the tree-LSTM, as in [23], on the task of natural language inference with ADAM for 10 epochs on MultiNLI and also SNLI.

Training Schedules: We use the following batch size schedules

- **BL.** Use a fixed small batch size as specified in the original paper introducing the model or as is standard.
- **CBS-k(-n).** Use a Cyclical Batch Size schedule, where k is the width of each step measured in epochs and n is the integer number of steps per cycle. When n is not specified it refers to the default value of 4. At the beginning of each cycle the batch size is initialized to the base batch size, and after each step it is then doubled.
- **CBS-k(-n)-A**. Use an aggressive Cyclical Batch Size schedule, which is equivalent to the original CBS schedule except after every step the batch size is quadrupled.
- **CBS-k(-n)-T**. Use a triangular Cyclical Batch Size schedule, which is modeled after the triangular schedule. Each cycle consists of n steps doubling the batch size after each step, then $n - 2$ symmetrical steps halving the batch size after each step.

In all language modeling CBS experiments, we use an initial batch size of 10, that is, half the baseline batch size as reported in the respective papers of each baseline model tested. The intuition behind starting with a smaller batch size is to introduce additional noise to help models escape sub-optimal local minima.

For adversarial training used in image classification, we use FGSM method [6] to generate adversarial examples. Adversarial training is implemented for the first half training epochs.

B Additional Results

This section shows additional experiment results.

Table 4: *Test perplexity of L1, L2 snapshot ensembled models on Penn Tree Bank (PTB) and WikiText-2 (WT2) datasets. Each CBS ensemble model is trained on its best-performing CBS schedule and each baseline ensemble model is snapshotted at the same epochs as its corresponding CBS ensemble model.*

	L1 PTB	L1 WT2	L2 PTB	L2 WT2
BL Single	83.13	96.41	79.34	99.69
BL Ens.	82.22	97.18	76.52	89.99
CBS Ens	81.51	94.27	76.14	88.47

Table 5: *Final testing perplexity and best testing perplexity of low-dropout L1' and L2' models on Penn Tree Bank (PTB) and WikiText 2 (WT2) datasets. The best perplexity values are **bolded**.*

	L1' on PTB		L1' on WT2		L2' on PTB		L2' on WT2	
Schedule	Final	Best	Final	Best	Final	Best	Final	Best
BL	132.4	108.4	132.2	129.8	231.7	106.9	177.3	136.6
CBS-10-A	119.4	105.4	119.0	115.7	177.2	97.8	**145.1**	116.7
CBS-5-A	115.1	95.6	116.6	111.4	257.0	**88.3**	178.5	106.9
CBS-1-A	**106.4**	**93.6**	**113.5**	**104.5**	**171.4**	88.7	147.1	**100.4**

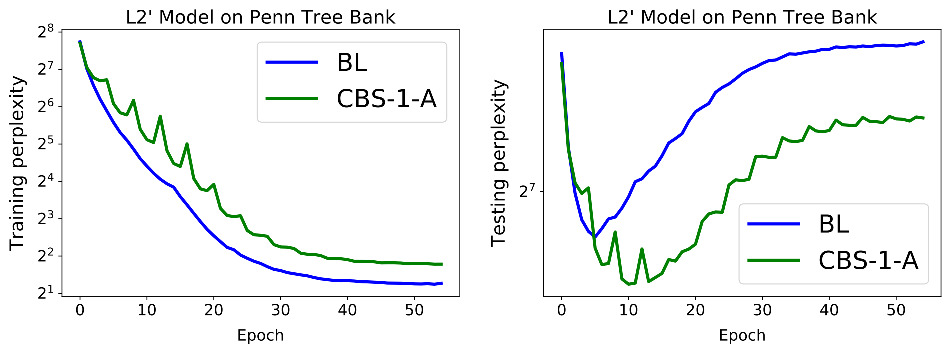

Figure 5: *Training (left) and testing (right) perplexity as a function of epoch for overfitting L2' model on Penn Tree Bank.*

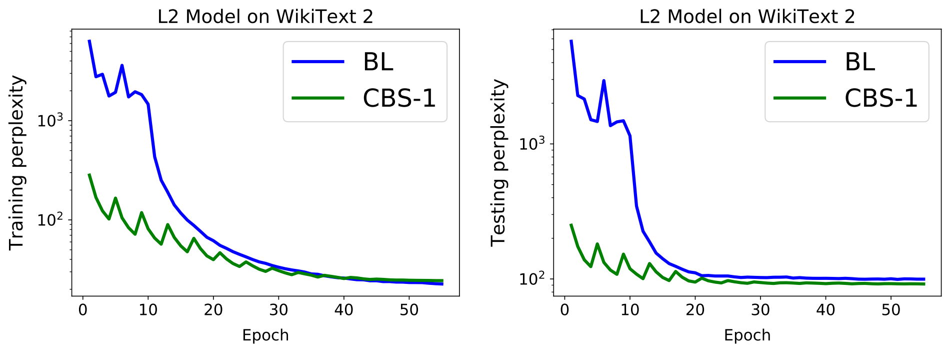

Figure 6: *Training (left) and testing (right) perplexity as a function of epoch for L2 model on WikiText 2.*