Surrogate Minimization: An Optimization Algorithm for Training Large Neural Networks with Model Parallelism

Reza Asad REZA_ASAD@SFU.CA
Simon Fraser University
Reza Babanezhad BABANEZHAD@GMAIL.COM
Samsung AI, Montreal
Issam Laradji ISSAM.LARADJI@GMAIL.COM
ServiceNow Research
Nicolas Le Roux NICOLAS@LE-ROUX.NAME
Microsoft Research
Sharan Vaswani VASWANI.SHARAN@GMAIL.COM
Simon Fraser University

Abstract

Optimizing large memory-intensive neural networks requires distributing its layers across multiple GPUs (referred to as *model parallelism*). We develop a framework that allows decomposing a neural network layer-wise and training it by optimizing layer-wise local losses in parallel. By using the resulting framework with GPipe [12], an effective pipelining strategy for model parallelism, we propose the Surrogate Minimization (`SM`) algorithm. `SM` allows for multiple parallel updates to the layer-wise parameters of a distributed neural network and consequently improves the GPU utilization of GPipe. Our framework ensures that the sum of local losses is a global upper-bound on the neural network loss, and can be minimized efficiently. Under mild technical assumptions, we prove that `SM` requires $O(1/\epsilon)$ iterations in order to guarantee convergence to an ϵ-neighbourhood of a stationary point of the neural network loss. Finally, our experimental results on MLPs demonstrate that `SM` leads to faster convergence compared to competitive baselines.

1. Introduction

As the size of neural networks continues to grow, models are too large to fit on a single GPU. This necessitates the use of splitting the model across machines or multiple GPUs and is referred to as *model parallelism* or *pipeline parallelism*. Distributing the model naively across GPUs and training it using standard optimization methods such as (stochastic) gradient descent results in only one GPU being active at a given time while rendering the other GPUs idle. The major challenge in effective model parallelism is to reduce the 'GPU idle time' during forward and backward passes. Gpipe [12] is a common technique that shrinks the "bubble of idle time" by partitioning a mini-batch into micro-batches. The partitioning is done in a way that a worker can reduce its wait time by immediately processing the next micro-batch (see Figure 1 for an illustration).

There are numerous algorithmic approaches that achieve model parallelism through forward, backward, and update unlocking of a large neural network [3, 10, 13, 14, 17, 27]. Forward locking refers to the constraint that one can not find the activation of a layer without visiting its previous layers. Similarly, backward locking is the same constraint during backpropagation as computing the gradients of a layer depends on the gradients computed from the upstream layers. In addition, update locking refers to the problem that updating the parameters of a layer would require a complete

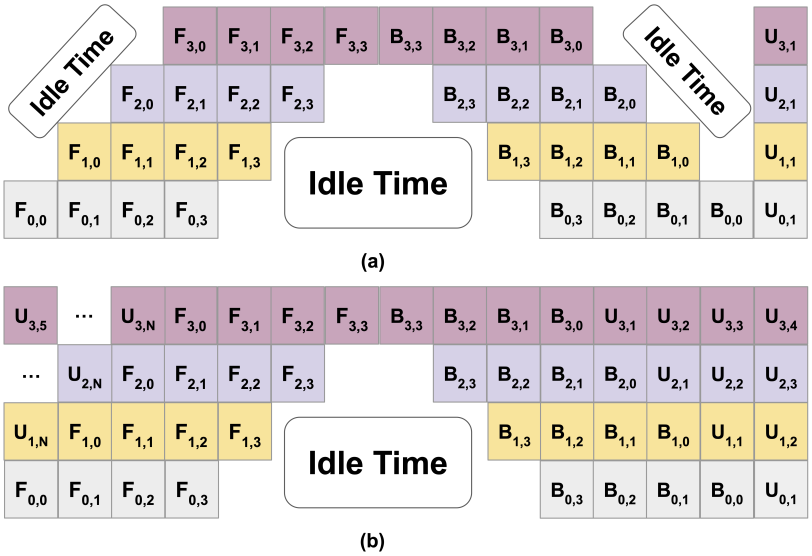
Idle Time
$F_{3,0}$ $F_{3,1}$ $F_{3,2}$ $F_{3,3}$ $B_{3,3}$ $B_{3,2}$ $B_{3,1}$ $B_{3,0}$
Idle Time
$U_{3,1}$
$F_{2,0}$ $F_{2,1}$ $F_{2,2}$ $F_{2,3}$ $B_{2,3}$ $B_{2,2}$ $B_{2,1}$ $B_{2,0}$ $U_{2,1}$
$F_{1,0}$ $F_{1,1}$ $F_{1,2}$ $F_{1,3}$
Idle Time
$B_{1,3}$ $B_{1,2}$ $B_{1,1}$ $B_{1,0}$ $U_{1,1}$
$F_{0,0}$ $F_{0,1}$ $F_{0,2}$ $F_{0,3}$ $B_{0,3}$ $B_{0,2}$ $B_{0,1}$ $B_{0,0}$ $U_{0,1}$
(a)
$U_{3,5}$ ... $U_{3,N}$ $F_{3,0}$ $F_{3,1}$ $F_{3,2}$ $F_{3,3}$ $B_{3,3}$ $B_{3,2}$ $B_{3,1}$ $B_{3,0}$ $U_{3,1}$ $U_{3,2}$ $U_{3,3}$ $U_{3,4}$
... $U_{2,N}$ $F_{2,0}$ $F_{2,1}$ $F_{2,2}$ $F_{2,3}$ $B_{2,3}$ $B_{2,2}$ $B_{2,1}$ $B_{2,0}$ $U_{2,1}$ $U_{2,2}$ $U_{2,3}$
$U_{1,N}$ $F_{1,0}$ $F_{1,1}$ $F_{1,2}$ $F_{1,3}$
Idle Time
$B_{1,3}$ $B_{1,2}$ $B_{1,1}$ $B_{1,0}$ $U_{1,1}$ $U_{1,2}$
$F_{0,0}$ $F_{0,1}$ $F_{0,2}$ $F_{0,3}$ $B_{0,3}$ $B_{0,2}$ $B_{0,1}$ $B_{0,0}$ $U_{0,1}$
(b)

Figure 1: We observe the need for algorithms that can optimize the layers of a large neural network (distributed across GPUs). To this end, we propose modifying GPipe's [12] pipeline parallelism framework (Figure (a)). For both Figures, the x-axis represents the training time for one epoch, and the graph is repeated across epochs. Here, $\mathbf{F_{i,j}}$, $\mathbf{B_{i,j}}$ and $\mathbf{U_{i,j}}$ represent the forward, backward, and update operation by worker i on micro-batch j. In Figure (a), we identify additional bubbles of idle time before and after each update phase. Figure (b) represents GPipe [12] using our optimizer. In this setup, one can do multiple updates to the parameters of the layers of a network in parallel and remove the idle time.

forward pass. We categorize the works addressing such problems as parametric or non-parametric approaches. [3, 14] are examples of parametric approaches that use a set of auxiliary networks to address the update locking problem. However, adding auxiliary networks introduces complexity and hyper-parameters, making the resulting implementation difficult. Furthermore, it has been shown that approaches like [14] do not lead to good performance for deeper networks [13]. On the other hand, [13] is an example of a non-parametric approach that uses delayed gradients to address the backward locking problem. However, as the number of splits in the model increases the negative impact of using delayed gradients on performance is more severe [13]. Another approach to achieving model parallelism is to use the Alternating Direction Method of Multipliers (ADMM) [23, 24]. In this framework, the objective of a large neural network can be split into sub-problems where the sub-problems are solved in parallel. As a gradient-free approach, ADMM does not suffer from the vanishing gradient problem. However, it is shown that, as the number of layers increases, ADMM suffers from a severe loss in accuracy while offering speed up compared to SGD [25]. Tunable Subnetwork Splitting Method (TSSM) [25] is a follow-up work that suggests a framework for fine-tuning the gain from the speed of ADMM while maintaining good accuracy. However, for TSSM, all experiments are done using one GPU, and no theoretical convergence guarantees are provided.

In this paper, we propose an alternative framework that allows decomposing the neural network and trains it by optimizing layer-wise surrogates (referred to as local losses) in parallel. Our decomposition ensures that the sum of local losses is a global upper-bound on the neural network loss, i.e. layer-wise surrogates majorize the function, and therefore descent is guaranteed [4, 8, 16]. Using our framework in conjunction with GPipe helps eliminate additional bubbles of idle time prior to and after each update phase. Figure 1 illustrates the difference between using vanilla GPipe [12] vs. our modification. The proposed framework motivates us to develop an optimization algorithm **Surrogate Minimization** (SM) that optimizes each layer-wise local loss *in parallel* resulting in multiple parametric updates for each layer. We note that the idea of constructing layer-wise local losses has been explored before. LocoProp [1] proposes two mechanisms for constructing local losses that allow for multiple parallel updates of the neural network parameters. However, the resulting algorithm introduces two sensitive hyper-parameters per layer. This makes implementing LocoProp [1] for deep neural networks difficult. In contrast, we demonstrate that the local losses in SM can be minimized using conjugate gradient without introducing additional tunable hyper-parameters.

The organization of this paper is as follows. In section 2 we formally describe our problem and assumptions. In section 3 we state the mechanism for constructing our local losses through Proposition 1 and provide a pseudo-code for running SM in the most general setting. Furthermore, in Theorem 1 of that section, we show that SM results in convergence to a stationary point of the neural network loss at an $\mathcal{O}(\frac{1}{T})$ rate. Finally, we share empirical results on MLPs in section 4 and demonstrate the effectiveness of using SM compared to other baselines.

2. Problem Formulation

Consider a neural network with m layers where $\theta^{(i)}$ corresponds to the parameters of layer i of the network. We define $z^{(i)}$ as the post-activation of the i-th layer where $z^{(0)} = X$ corresponds to the input, and $z^{(i)} = f(z^{(i-1)}, \theta^{(i-1)})$ with f as a smooth activation function[1]. Given an output vector y, we aim to minimize the loss $h(\theta) := \ell(z^{(m)})$ w.r.t $\theta = [\theta^{(0)}, \theta^{(1)}, \dots, \theta^{(m-1)}]$. Throughout, we consider ℓ to be a convex loss, for example, equal to the squared loss for regression or the cross-entropy loss for classification. We will assume that ℓ is $L_z^{(m)}$ smooth w.r.t $z^{(m)}$. This property is satisfied for standard losses – $L_z^{(m)} = 1$ for the squared loss and $L_z^{(m)} = \frac{1}{4}$ for the logistic loss. Though ℓ is convex, we note that $h(\theta)$ is a potentially non-convex function.

In the deterministic setting, GD is the standard method to optimize $h(\theta)$. Specifically, in each iteration $t \in [T]$, GD updates the vector θ as: $\theta_{t+1} = \theta_t - \alpha_t \nabla h(\theta)$, where α_t is the (potentially varying) step-size at iteration t. Another common optimization algorithm is Newton's method which makes use of the Hessian $\nabla^2 h(\theta_t)$ at iteration t and corresponds to the update rule: $\theta_{t+1} = \theta_t - [\nabla^2 h(\theta_t)]^{-1} \nabla h(\theta_t)$. Compared to these approaches, SM updates the parameters of each layer of a neural network in parallel for N steps. This capability amortizes the expensive cost of forward/backward operations for a large neural network. Furthermore, SM utilizes the per-layer local losses to form a block diagonal Hessian (using finite differences [21]) and hence avoids storing the full Hessian in memory (Please see Section 3 for details).

Our method can be thought of as an extension to surrogate optimization proposed by Lavington et al. [18]. This method is designed for applications where computing the loss $\ell(z)$ is expensive. For example, in reinforcement learning (RL), computing $\nabla \ell(z^{(m)})$ requires expensive interactions with the environment. Methods such as GD or Newton's method require computing $\nabla \ell(z)$ for each update to the parameters, and are hence computationally expensive. To alleviate this problem, the SSO algorithm in Lavington et al. [18] uses the gradient of ℓ w.r.t $z^{(m)}$ to form a surrogate loss. The surrogate loss is a global upper-bound on $h(\theta)$ and optimizing the surrogate can be done efficiently

[1] For notational convenience, we assume that f is the same activation function across layers. All our results hold when f is different across layers.

for multiple steps without re-computing $\nabla\ell(z^{(m)})$. Our framework and the resulting `SM` algorithm can be interpreted as a parallel, multi-layer extension to `SSO`, and we describe it in the next section.

3. Surrogate Minimization

We first describe the surrogate minimization framework and the resulting algorithm, and then analyze its theoretical convergence.

We use a single forward and backward pass over the network to decompose it and construct a set of local losses (per layer). More concretely, we use the smoothness of the global loss to form an upper bound for it i.e., a surrogate loss. We then use our decoupling Lemma 1 on the smooth surrogate loss to construct local losses for all layers recursively. In Proposition 1, we formally show this construction and prove that the resulting sum of local losses is a global upper-bound on $\ell(z^{(m)})$.

Proposition 1 *Consider an m-layer neural network model where $z = [z^{(1)}, z^{(2)}, \dots z^{(m)}]$ represent the post-activations with $z^0 = X$, $\theta = [\theta^{(0)}, \theta^{(1)}, \dots, \theta^{(m-1)}]$ correspond to the parameters, f is the activation function such that $z^{(i)} = f(z^{(i-1)}, \theta^{(i-1)})$. For an arbitrary, but fixed z_t, θ_t, the loss $\ell(z^m)$ can be upper-bounded in terms of the local losses $\mathcal{L}^{(i)}_{\theta,t}({z_t}^{(i-1)}, \theta^{(i-1)})$ at layer i as follows:*

$$h(\theta) = \ell(z^{(m)}) \leq \ell({z_t}^{(m)}) + \sum_{i=1}^{m} \mathcal{L}^{(i)}_{\theta,t}({z_t}^{(i-1)}, \theta^{(i-1)}) \text{ where,}$$

$$\mathcal{L}^{(i)}_{\theta,t}({z_t}^{(i-1)}, \theta^{(i-1)}) := \left\langle \frac{\partial h(\theta_t)}{\partial \theta^{(i-1)}}, \theta^{(i-1)} - \theta_t^{(i-1)} \right\rangle + \frac{1}{2} \left\| \theta^{(i-1)} - \theta_t^{(i-1)} \right\|^2_{\mathbb{H}^{(i)}_{\theta\theta,t}} + \frac{2\rho_t^{(i)}}{3} \left\| \theta^{(i-1)} - \theta_t^{(i-1)} \right\|^3$$

Here, $\mathbb{H}^{(i)}_{\theta\theta,t}$ is derived from $\frac{\partial^2 h(\theta_t)}{\partial(\theta^{(i-1)})^2}$ and depends on z_t (see Proposition 1 for details).

The norms in the local losses can be interpreted as regularization terms to control the deviation of $\theta^{(i-1)}$ from $\theta_t^{(i-1)}$. Proposition 1 is derived in Appendix B and uses a third-order Taylor series approximation along with Lemma 1 that enables us to decouple the different layers in the neural network. Lemma 1 can be interpreted as a "matrix" variant of the Fenchel-Young inequality, and might be of independent interest. We note that Proposition 1 holds for any θ and is thus a global upper-bound on $\ell(z^{(m)})$. Another interpretation of Proposition 1 is that it provides a method to create a block-diagonal approximation of $\nabla^2 h(\theta_t)$, the Hessian of the global loss, by only considering the activations and the weights at each layer separately, thus never working with the full matrix. To see this, note that $\nabla^2 \sum_{i=1}^{m} \mathcal{L}^{(i)}_{\theta,t}({z_t}^{(i-1)}, \theta_t^{(i-1)})$ is a block diagonal matrix with block i equal to $\mathbb{H}^{(i)}_{\theta\theta,t}$. Unlike other works that form block diagonal approximations of either the Hessian [5, 19, 26] or the Fisher information matrix [11, 20], the proposed block diagonal approximation is guaranteed to be an upper-bound on the full Hessian.

Using the concept of majorization-minimization immediately implies an algorithm – minimize the local losses individually and in parallel for N steps. Since the local losses majorize the function, the resulting algorithm will guarantee descent on $\ell(z^{(m)})$. Hence, `SM` involves solving multiple non-convex cubic-regularized problems [22] (one for each layer). These problems can be solved using standard techniques including gradient descent [6] or Krylov subspace methods such as conjugate gradient [7]. In Algorithm 1, we outline the pseudo-code for `SM` that uses GD to minimize the local losses. During each iterate of `SM`, a single forward and backward pass is completed to construct layer-wise local losses. The local losses are then used in parallel to perform N updates in the

Algorithm 1 Surrogate Minimization (SM)

```
Input: $\theta_0$ (initialization), $T$ (number of iterations), $N$ (number of inner-loops), $\alpha$ (step-size)
for $t = 0 \to T - 1$ do
    Access the gradient oracle to construct $\mathcal{L}^{(i)}_{\theta,t}(z_t{}^{(i-1)}, \theta^{(i-1)})$ as in Equation (1)
    do in parallel $\forall i \in [1, m]$
        Initialize inner-loop: $\omega_0^{(i-1)} = \theta_t^{(i-1)}$
        for $k \leftarrow 0$ to $N - 1$ do
            $\omega_{k+1}^{(i-1)} = \omega_k^{(i-1)} - \alpha \nabla_\omega \mathcal{L}^{(i)}_{\theta,t}(z_t{}^{(i-1)}, \omega_k^{(i-1)})$
        end
        $\theta_{t+1}^{(i-1)} = \omega_N^{(i-1)}$
    end
    $z_{t+1}^{(1)} = f(X, \theta_{t+1}^{(0)})$ ; $\forall i \in [2, m]$, $z_{t+1}^{(i)} = f(z_{t+1}^{(i-1)}, \theta_{t+1}^{(i-1)})$
end
Return $\theta_T$
```

parametric space, until convergence. Finally, we note that assuming $\rho_t^{(i)} = 0$ and upper-bounding the Hessian by its maximum eigenvalue (corresponding to a second-order Taylor series expansion) results in a quadratic local loss that can be minimized exactly. In this special case, SM corresponds to doing a gradient descent update w.r.t the parameters θ with a potentially different step-size for each layer. In contrast to SM, SSO [18] forms a single upper-bound (across all layers) on $\ell(z^m)$ and minimizes the resulting surrogate. Hence, SSO cannot be parallelized across layers, and every parametric update involves an expensive backward pass across the network. Hence, SM can be interpreted as a parallel multi-layer extension of SSO that can also be used for applications such as reinforcement learning. Next, we analyze the theoretical convergence of SM. Our main result shows that if the global loss ($\ell(z^{(m)})$) and local losses ($\mathcal{L}^{(i)}_{\theta,t}(z_t{}^{(i-1)}, \theta^{(i-1)})$) are smooth, SM is guaranteed to converge to a stationary point in $O\left(\frac{1}{T}\right)$ iterations.

Theorem 1 *Assuming that (i) $h(\theta) = \ell(z^m)$ is bounded from below by h^*, (ii) $\ell(z^{(m)})$ is $L_z^{(m)}$ smooth, (iii) local losses $\mathcal{L}^{(i)}_{\theta,t}$ are β smooth for all $i \in [1, m], t \in [T]$, T iterations of SM with $\eta_z^{(m)} \leq \frac{1}{L_z^{(m)}}$ and $N \geq 1$ inner-loops in each iteration converges as: $\min_{t \in [T]} \|\nabla h(\theta_t)\|_2^2 \leq \frac{2\beta\,[h(\theta_0) - h^*]}{T}$.*

Theorem 1 (proved in Appendix B) holds for any number of inner-loops $N \geq 1$. Since ℓ corresponds to a standard loss such as squared loss or cross-entropy, we can easily compute $L_z^{(m)}$. In practice, SM uses GD in conjunction with an Armijo line-search [2] for minimizing each local loss and hence does not require the knowledge of β to set the step-size α in the inner-loop.

4. Experiments

Across all experiments, we assume that $\rho_t^{(i)} = 0$ for all local losses, alleviating the need to estimate these constants. This reduces each local loss to a quadratic (as a function of θ) which we minimize using N steps of the conjugate gradient (CG) algorithm. Under this setup, we compare SM with various baselines using 2, 4, and 8 layers of MLP on the MNIST [9] dataset in the deterministic setting. All optimizers run for 300 epochs and we use torch's GPipe Pipeline Parallelism [15] library (except for DDG [13]). For DDG [13] we use the paper's code which utilizes Python's multi-processing and dedicates a process for each split handling delayed gradients[2]. For the 2-layer experiment, we dedicate one GPU per layer. For the experiments involving 4 and 8 layers of MLP, we distribute the layers equally among 4 GPUs. All experiments presented here use the Nvidia Tesla V100 GPU (16 GB memory) and Intel Xeon Gold 6126 CPU (2.60GHz). Although our theory requires a smooth activation function, we are able to outperform the baselines using the ReLU activation.

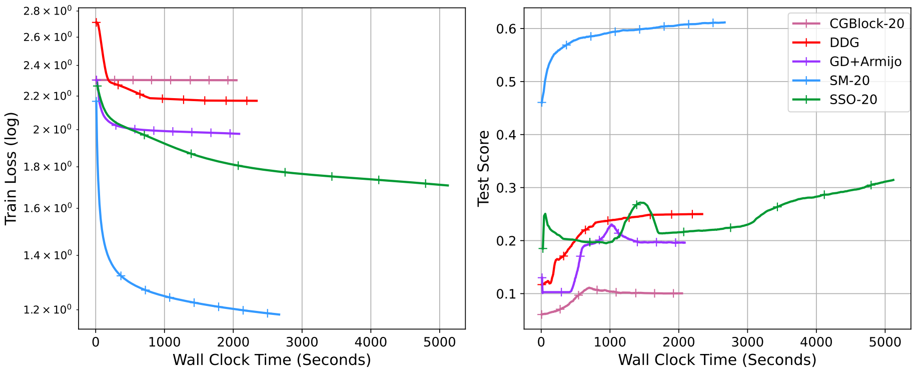

Table 1: MNIST Classification with 8 layers of MLP for N = 20

Table 1 shows the evolution of training loss and test accuracy for 8 layers of MLP. **GD+Armijo** refers to Gradient Descent where the step-size is chosen adaptively using the Armijo line search [2]. We also examined GD with a constant step-size and it consistently resulted in worse performance. **DDG** [13] splits a neural network across GPUs and uses delayed gradients for backward unlocking. We also compare to the parametric Newton method (**CGBlock**) that forms a block diagonal Hessian (using finite differences [21], similar to SM) and is minimized with N steps of CG. Note that both SM and CGBlock form a block diagonal Hessian. However, CGBlock constructs the Hessian using the global loss $h(\theta)$, whereas SM utilizes the per-layer local losses. Finally, we compare SM with SSO [18]. For SM, CGBlock, and SSO [18], we compare the optimizers for 1, 10, and 20 steps of parametric updates per iteration (see Tables 2, 3, and 4 in Appendix D). As we increase the number of parametric updates per iteration, our results consistently indicate that the performance gap between SM and the baselines increases. Furthermore, as the number of layers increases, the performance gap between SM and the baselines increases dramatically. We also observe that CGBlock results in poor performance due to the presence of negative curvature and adding regularization slows down the convergence. For details about other practical considerations, see Appendix A.

5. Discussion

We presented SM, an optimization algorithm that is suitable for training distributed neural networks using GPipe [12]. In the future, we aim to study other model parallelism frameworks that unlock a neural network during forward, backward and update phases. For example, we would like to explore the use of delayed gradients and activations (for backward and forward unlocking of the network) and its effect on the performance of SM.

[2] We were not successful in extending the paper's code to work with GPipe.

References

[1] Ehsan Amid, Rohan Anil, and Manfred Warmuth. Locoprop: Enhancing backprop via local loss optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 9626–9642. PMLR, 2022.

[2] Larry Armijo. Minimization of functions having lipschitz continuous first partial derivatives. *Pacific Journal of mathematics*, 16(1):1–3, 1966.

[3] Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Decoupled greedy learning of cnns. In *International Conference on Machine Learning*, pages 736–745. PMLR, 2020.

[4] Dankmar Böhning and Bruce G Lindsay. Monotonicity of quadratic-approximation algorithms. *Annals of the Institute of Statistical Mathematics*, 40(4):641–663, 1988.

[5] Aleksandar Botev, Hippolyt Ritter, and David Barber. Practical gauss-newton optimisation for deep learning. In *International Conference on Machine Learning*, pages 557–565. PMLR, 2017.

[6] Yair Carmon and John Duchi. Gradient descent finds the cubic-regularized nonconvex newton step. *SIAM Journal on Optimization*, 29(3):2146–2178, 2019.

[7] Coralia Cartis, Nicholas IM Gould, and Philippe L Toint. Adaptive cubic regularisation methods for unconstrained optimization. part i: motivation, convergence and numerical results. *Mathematical Programming*, 127(2):245–295, 2011.

[8] Jan De Leeuw. Block-relaxation algorithms in statistics. In *Information Systems and Data Analysis: Prospects—Foundations—Applications*, pages 308–324. Springer, 1994.

[9] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[10] Aidan N Gomez, Oscar Key, Kuba Perlin, Stephen Gou, Nick Frosst, Jeff Dean, and Yarin Gal. Interlocking backpropagation: Improving depthwise model-parallelism. *The Journal of Machine Learning Research*, 23(1):7714–7741, 2022.

[11] Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, pages 1842–1850. PMLR, 2018.

[12] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *Advances in neural information processing systems*, 32, 2019.

[13] Zhouyuan Huo, Bin Gu, Heng Huang, et al. Decoupled parallel backpropagation with convergence guarantee. In *International Conference on Machine Learning*, pages 2098–2106. PMLR, 2018.

[14] Max Jaderberg, Wojciech Marian Czarnecki, Simon Osindero, Oriol Vinyals, Alex Graves, David Silver, and Koray Kavukcuoglu. Decoupled neural interfaces using synthetic gradients. In *International conference on machine learning*, pages 1627–1635. PMLR, 2017.

[15] Chiheon Kim, Heungsub Lee, Myungryong Jeong, Woonhyuk Baek, Boogeon Yoon, Ildoo Kim, Sungbin Lim, and Sungwoong Kim. torchgpipe: On-the-fly pipeline parallelism for training giant models. 2020.

[16] Kenneth Lange. *MM optimization algorithms*. SIAM, 2016.

[17] Michael Laskin, Luke Metz, Seth Nabarro, Mark Saroufim, Badreddine Noune, Carlo Luschi, Jascha Sohl-Dickstein, and Pieter Abbeel. Parallel training of deep networks with local updates. *arXiv preprint arXiv:2012.03837*, 2020.

[18] Jonathan Wilder Lavington, Sharan Vaswani, Reza Babanezhad, Mark Schmidt, and Nicolas Le Roux. Target-based surrogates for stochastic optimization. *ICML*, 2023.

[19] Nicolas Le Roux, Pierre-Antoine Manzagol, and Yoshua Bengio. Topmoumoute online natural gradient algorithm. *Advances in neural information processing systems*, 20, 2007.

[20] James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pages 2408–2417. PMLR, 2015.

[21] James Martens et al. Deep learning via hessian-free optimization. In *ICML*, volume 27, pages 735–742, 2010.

[22] Yurii Nesterov and Boris T Polyak. Cubic regularization of newton method and its global performance. *Mathematical Programming*, 108(1):177–205, 2006.

[23] Gavin Taylor, Ryan Burmeister, Zheng Xu, Bharat Singh, Ankit Patel, and Tom Goldstein. Training neural networks without gradients: A scalable admm approach. In *International conference on machine learning*, pages 2722–2731. PMLR, 2016.

[24] Junxiang Wang, Fuxun Yu, Xiang Chen, and Liang Zhao. Admm for efficient deep learning with global convergence. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 111–119, 2019.

[25] Junxiang Wang, Zheng Chai, Yue Cheng, and Liang Zhao. Tunable subnetwork splitting for model-parallelism of neural network training. *arXiv preprint arXiv:2009.04053*, 2020.

[26] Huishuai Zhang, Caiming Xiong, James Bradbury, and Richard Socher. Block-diagonal hessian-free optimization for training neural networks. *arXiv preprint arXiv:1712.07296*, 2017.

[27] Huiping Zhuang, Yi Wang, Qinglai Liu, and Zhiping Lin. Fully decoupled neural network learning using delayed gradients. *IEEE transactions on neural networks and learning systems*, 33(10):6013–6020, 2021.

Supplementary Material

Organization of the Appendix

Appendix A. Practical Considerations

Notice that the construction of $\mathbb{H}$ involves a matrix inversion. To avoid this, we use another upper-bound as follows: For $i \in [2, m]$

$$\begin{bmatrix} H^{(i)}_{zz,t} & H^{(i)}_{z\theta,t} \\ H^{\top(i)}_{z\theta,t} & H^{(i)}_{\theta\theta,t} \end{bmatrix} \preceq \begin{bmatrix} L^{(i)}_{z,t} I & H^{(i)}_{z\theta,t} \\ H^{\top(i)}_{z\theta,t} & H^{(i)}_{\theta\theta,t} \end{bmatrix},$$

where $L^{(i)}_{z,t} = \lambda_{\max}[H^{(i)}_{zz,t}]$. To compute $L^{(i)}_{z,t}$ we use a few iterations of the power method (using finite differences [21], without storing the Hessian). We further upper-bound this matrix by using the decoupling Lemma 1 resulting in the conditions:

$$(1) : \mathbb{H}^{(i)}_{zz,t} \succeq L^{(i)}_{z,t} I \quad ; \quad (2) : \mathbb{H}^{(i)}_{\theta\theta,t} \succeq H^{(i)}_{\theta\theta,t} + H^{\top(i)}_{z\theta,t} [\mathbb{H}^{(i)}_{zz,t} - L^{(i)}_{z,t} I]^{-1} H^{(i)}_{z\theta,t}$$

We now choose $\mathbb{H}^{(i)}_{zz,t} = \tilde{L}^{(i)}_{z,t}$ s.t. $\tilde{L}^{(i)}_{z,t} - L^{(i)}_{z,t} = \delta > 0$, where δ is a tunable hyper-parameter. This ensures condition (1) is satisfied. In order to guarantee that condition (2) is satisfied, we set $\mathbb{H}^{(i)}_{\theta\theta,t}$ as follows:

$$\mathbb{H}^{(i)}_{\theta\theta,t} = H^{(i)}_{\theta\theta,t} + \frac{H^{\top(i)}_{z\theta,t} H^{(i)}_{z\theta,t}}{\delta},$$

which can be easily computed.

Appendix B. Proofs

Proposition 1 *Consider using an m-layer neural network model where $z = [z^{(1)}, z^{(2)}, \dots z^{(m)}]$ represents the post-activations across the m layers with $z^0 = X$, $\theta = [\theta^{(0)}, \theta^{(1)}, \dots, \theta^{(m-1)}]$ correspond to the parameters of the m layers, f represents the activation functions of the m layers such that $z^{(i)} = f(z^{(i-1)}, \theta^{(i-1)})$. Consider minimizing the loss $h(\theta) = \ell(z^{(m)})$ where ℓ corresponds to a standard loss function. For an arbitrary, but fixed z_t and θ_t, the loss $\ell(z^m)$ can be upper-bounded as follows:*

$$\ell(z^{(m)}) \leq \ell({z_t}^{(m)}) + \sum_{i=1}^{m} \mathcal{L}^{(i)}_{\theta,t}({z_t}^{(i-1)}, \theta^{(i-1)}),$$

$$\begin{aligned}\mathcal{L}^{(i)}_{\theta,t}({z_t}^{(i-1)}, \theta^{(i-1)}) := \langle g^{(i)}_{\theta,t}, \theta^{(i-1)} - \theta^{(i-1)}_t \rangle + \frac{1}{2} \left\| \theta^{(i-1)} - \theta^{(i-1)}_t \right\|^2_{\mathbb{H}^{(i)}_{\theta\theta,t}} \\ + \frac{2\rho^{(i)}_t}{3} \left\| \theta^{(i-1)} - \theta^{(i-1)}_t \right\|^3 \textit{ For } i \in [1, m]\end{aligned} \tag{1}$$

$$\begin{aligned}\Psi^{(m)}_t(z^{(m-1)}, \theta^{(m-1)}) := \langle \nabla \ell({z_t}^{(m)}), f(z^{(m-1)}, \theta^{(m-1)}) - f({z_t}^{(m-1)}, \theta^{(m-1)}_t) \rangle \\ + \frac{1}{2\,\eta^{(m)}_z} \left\| f(z^{(m-1)}, \theta^{(m-1)}) - f({z_t}^{(m-1)}, \theta^{(m-1)}_t) \right\|^2_2\end{aligned}$$

$$\begin{aligned}\Psi^{(i-1)}_t(z^{(i-2)}, \theta^{(i-2)}) := \langle g^{(i)}_{z,t}, f(z^{(i-2)}, \theta^{(i-2)}) - f({z_t}^{(i-2)}, \theta^{(i-2)}_t) \rangle \\ + \frac{1}{2} \left\| f(z^{(i-2)}, \theta^{(i-2)}) - f({z_t}^{(i-2)}, \theta^{(i-2)}_t) \right\|^2_{\mathbb{H}^{(i)}_{zz,t}} \\ + \frac{2\rho^{(i)}_t}{3} \left\| f(z^{(i-2)}, \theta^{(i-2)}) - f({z_t}^{(i-2)}, \theta^{(i-2)}_t) \right\|^3 \qquad \text{(For } i \in [2, m]\text{)}\end{aligned}$$

$$\begin{aligned}
g^{(i)}_{z,t} &:= \frac{\partial \Psi^{(i)}_t({z_t}^{(i-1)}, \theta^{(i-1)}_t)}{\partial z^{(i-1)}} = \frac{\partial \ell(z_t)}{\partial z^{(i-1)}} \quad ; \quad g^{(i)}_{\theta,t} := \frac{\partial \Psi^{(i)}_t({z_t}^{(i-1)}, \theta^{(i-1)}_t)}{\partial \theta^{(i-1)}} = \frac{\partial h(\theta_t)}{\partial \theta^{(i-1)}} \quad ; \\
H^{(i)}_t &= \nabla^2 \Psi^{(i)}_t({z_t}^{(i-1)}, \theta^{(i-1)}_t) = \begin{bmatrix} H^{(i-1)}_{zz,t} & H^{(i-1)}_{z\theta,t} \\ {H^{\intercal}}^{(i-1)}_{z\theta,t} & H^{(i-1)}_{\theta\theta,t} \end{bmatrix} \\
H^{(i)}_{zz,t} &:= \frac{\partial^2 \Psi^{(i)}_t({z_t}^{(i-1)}, \theta^{(i-1)}_t)}{\partial (z^{(i-1)})^2} = \frac{\partial^2 \ell(z_t)}{\partial (z^{(i-1)})^2} \\
H^{(i)}_{\theta\theta,t} &:= \frac{\partial^2 \Psi^{(i)}_t({z_t}^{(i-1)}, \theta^{(i-1)}_t)}{\partial (\theta^{(i-1)})^2} = \frac{\partial^2 h(\theta_t)}{\partial (\theta^{(i-1)})^2}\,; \\
H^{(i)}_{z\theta,t} &:= \frac{\partial^2 \Psi^{(i)}_t({z_t}^{(i-1)}, \theta^{(i-1)}_t)}{\partial z^{(i-1)}\,\partial \theta^{(i-1)}} = \frac{\partial^2 \ell(z_t)}{\partial (z^{(i-1)})\,\partial (\theta^{(i-1)})} \\
&\left\| \nabla^3 \Psi^{(i)}_t({z_t}^{(i-1)}, \theta^{(i-1)}_t) \right\| \leq \rho^{(i)}_t \quad ; \quad \left\| \nabla^2 \ell(z^{(m)}) \right\| \leq L^{(m)}_z
\end{aligned}$$

Matrices $\mathbb{H}^{(i)}_{zz,t}$ and $\mathbb{H}^{(i)}_{\theta\theta,t}$ are constructed such that,

$$\textit{(1)} : \mathbb{H}^{(i)}_{zz,t} \geq H^{(i)}_{zz,t} \quad ; \quad \textit{(2)} : \mathbb{H}^{(i)}_{\theta\theta,t} \geq H^{(i)}_{\theta\theta,t} + H^{\top}{}^{(i)}_{z\theta,t}[\mathbb{H}^{(i)}_{zz,t} - H^{(i)}_{zz,t}]^{-1} H^{(i)}_{z\theta,t}$$

Proof Using the $L_z^{(m)}$-smoothness of ℓ w.r.t $z^{(m)}$, for a fixed $z_t{}^{(m)}$,

$$\ell(z^{(m)}) \leq \ell(z_t{}^{(m)}) + \langle \nabla \ell(z_t{}^{(m)}), z^{(m)} - z_t{}^{(m)} \rangle + \frac{L_z^{(m)}}{2} \left\| z^{(m)} - z_t{}^{(m)} \right\|_2^2 \tag{2}$$

For $\eta_z^{(m)} \leq \frac{1}{L_z^{(m)}}$, since $z^{(m)} = f(z^{(m-1)}, \theta^{(m-1)})$ and $z_t{}^{(m)} = f(z_t{}^{(m-1)}, \theta_t^{(m-1)})$,

$$\begin{aligned} \Psi_t^{(m)}(z^{(m-1)}, \theta^{(m-1)}) := \langle \nabla \ell(z_t{}^{(m)}), f(z^{(m-1)}, \theta^{(m-1)}) - f(z_t{}^{(m-1)}, \theta_t^{(m-1)}) \rangle \\ + \frac{1}{2\,\eta_z^{(m)}} \left\| f(z^{(m-1)}, \theta^{(m-1)}) - f(z_t{}^{(m-1)}, \theta_t^{(m-1)}) \right\|_2^2 \end{aligned} \tag{3}$$

$$\implies \ell(z^{(m)}) \leq \ell(z_t{}^{(m)}) + \Psi_t^{(m)}(z^{(m-1)}, \theta^{(m-1)}) \tag{4}$$

Now, we will upper-bound $\Psi_t^{(m)}(z^{(m-1)}, \theta^{(m-1)})$ using a 3rd-order Taylor series approximation around $(z_t{}^{(m-1)}, \theta_t^{(m-1)})$.

$$\begin{aligned} \Psi_t^{(m)}(z^{(m-1)}, \theta^{(m-1)}) \leq \Psi_t^{(m)}(z_t{}^{(m-1)}, \theta_t^{(m-1)}) \\ + \left\langle \begin{bmatrix} g^{(m)}_{z,t} \\ g^{(m)}_{\theta,t} \end{bmatrix}, \begin{bmatrix} z^{(m-1)} - z_t{}^{(m-1)} \\ \theta^{(m-1)} - \theta_t^{(m-1)} \end{bmatrix} \right\rangle \\ + \frac{1}{2} \begin{bmatrix} z^{(m-1)} - z_t{}^{(m-1)} \\ \theta^{(m-1)} - \theta_t^{(m-1)} \end{bmatrix}^{\top} \begin{bmatrix} H^{(m)}_{zz,t} & H^{(m)}_{z\theta,t} \\ H^{\top}{}^{(m)}_{z\theta,t} & H^{(m)}_{\theta\theta,t} \end{bmatrix} \begin{bmatrix} z^{(m-1)} - z_t{}^{(m-1)} \\ \theta^{(m-1)} - \theta_t^{(m-1)} \end{bmatrix} \\ + \frac{\rho_t^{(m)}}{6} \left\| \begin{bmatrix} z^{(m-1)} - z_t{}^{(m-1)} \\ \theta^{(m-1)} - \theta_t^{(m-1)} \end{bmatrix} \right\|^3 \end{aligned}$$

Using Lemma 1, we can upper-bound the above term as:

$$\begin{aligned} \Psi_t^{(m)}(z^{(m-1)}, \theta^{(m-1)}) \leq \Psi_t^{(m)}(z_t{}^{(m-1)}, \theta_t^{(m-1)}) + \mathcal{L}^{(m)}_{z,t}(z^{(m-1)}, \theta_t^{(m-1)}) \\ + \mathcal{L}^{(m)}_{\theta,t}(z_t{}^{(m-1)}, \theta^{(m-1)}) \end{aligned} \tag{5}$$

$$\begin{aligned} \mathcal{L}^{(m)}_{z,t}(z^{(m-1)}, \theta_t^{(m-1)}) = \langle g^{(m)}_{z,t}, z^{(m-1)} - z_t{}^{(m-1)} \rangle + \frac{1}{2} \left\| z^{(m-1)} - z_t{}^{(m-1)} \right\|^2_{\mathbb{H}^{(m)}_{zz,t}} \\ + \frac{2\rho_t^{(m)}}{3} \left\| z^{(m-1)} - z_t{}^{(m-1)} \right\|^3 \end{aligned} \tag{6}$$

$$\begin{aligned}\mathcal{L}^{(m)}_{\theta,t}(z_t{}^{(m-1)}, \theta^{(m-1)}) = \langle g^{(m)}_{\theta,t}, \theta^{(m-1)} - \theta^{(m-1)}_t\rangle + \frac{1}{2}\left\|\theta^{(m-1)} - \theta^{(m-1)}_t\right\|^2_{\mathbb{H}^{(m)}_{\theta\theta,t}} \\ + \frac{2\rho^{(m)}_t}{3}\left\|\theta^{(m-1)} - \theta^{(m-1)}_t\right\|^3\end{aligned} \tag{7}$$

Since $z^{(m-1)} = f(z^{m-2}, \theta^{(m-2)})$, we can express $\mathcal{L}^{(m)}_{z,t}(z^{(m-1)}, \theta^{(m-1)}_t)$ in terms of $z^{(m-2)}$ and $\theta^{(m-2)}$, i.e.
$\mathcal{L}^{(m)}_{z,t}(z^{(m-1)}, \theta^{(m-1)}_t) = \Psi^{(m-1)}_t(z^{(m-2)}, \theta^{(m-2)})$ where $\Psi^{(m-1)}_t(z^{(m-2)}, \theta^{(m-2)})$ is defined as:

$$\begin{aligned}\Psi^{(m-1)}_t(z^{(m-2)}, \theta^{(m-2)}) &= \langle g^{(m)}_{z,t}, f(z^{(m-2)}, \theta^{(m-2)}) - f(z_t{}^{(m-2)}, \theta^{(m-2)}_t)\rangle \\ &+ \frac{1}{2}\left\|f(z^{(m-2)}, \theta^{(m-2)}) - f(z_t{}^{(m-2)}, \theta^{(m-2)}_t)\right\|^2_{\mathbb{H}^{(m)}_{zz,t}} \\ &+ \frac{2\rho^{(m)}_t}{3}\left\|f(z^{(m-2)}, \theta^{(m-2)}) - f(z_t{}^{(m-2)}, \theta^{(m-2)}_t)\right\|^3\end{aligned}$$

Putting together the above inequalities, we can express $\Psi^{(m)}_t$ in terms of $\Psi^{(m-1)}_t$ as follows:

$$\begin{aligned}\Psi^{(m)}_t(z^{(m-1)}, \theta^{(m-1)}) \le \Psi^{(m)}_t(z_t{}^{(m-1)}, \theta^{(m-1)}_t) + \mathcal{L}^{(m)}_{\theta,t}(z_t{}^{(m-1)}, \theta^{(m-1)}) \\ + \Psi^{(m-1)}_t(z^{(m-2)}, \theta^{(m-2)})\end{aligned} \tag{8}$$

Similarly, using the decoupling lemma, we can relate consecutive layers (i) and $(i{-}1)$ for $i \in [2, m]$,

$$\Psi^{(i)}_t(z^{(i-1)}, \theta^{(i-1)}) \le \Psi^{(i)}_t(z_t{}^{(i-1)}, \theta^{(i-1)}_t) + \mathcal{L}^{(i)}_{\theta,t}(z_t{}^{(i-1)}, \theta^{(i-1)}) + \Psi^{(i-1)}_t(z^{(i-2)}, \theta^{(i-2)}) \tag{9}$$

Recursing over $i \in [2, m]$,

$$\Psi^{(m)}_t(z^{(m-1)}, \theta^{(m-1)}) \le \sum_{i=2}^{m}\left[\Psi^{(i)}_t(z_t{}^{(i-1)}, \theta^{(i-1)}_t) + \mathcal{L}^{(i)}_{\theta,t}(z_t{}^{(i-1)}, \theta^{(i-1)})\right] + \Psi^{(1)}_t(z^{(0)}, \theta^{(0)}) \tag{10}$$

For bounding $\Psi^{(1)}_t(z^{(0)}, \theta^{(0)})$, recall that $z^{(0)} = X$. Using the third-order Taylor series w.r.t $\theta^{(0)}$,

$$\begin{aligned}\Psi^{(1)}_t(X, \theta^{(0)}) &\le \Psi^{(1)}_t(X, \theta^{(0)}_t) \\ &+ \underbrace{\langle g^{(1)}_{\theta,t}, \theta^{(0)} - \theta^{(0)}_t\rangle + \frac{1}{2}\left\|\theta^{(0)} - \theta^{(0)}_t\right\|^2_{H_{\theta\theta,1}} + \frac{\rho^{(1)}_t}{6}\left\|\theta^{(0)} - \theta^{(0)}_t\right\|^3}_{=\mathcal{L}^{(1)}_t(\theta^{(0)})} \\ &= \Psi^{(1)}_t(X, \theta^{(0)}_t) + \mathcal{L}^{(1)}_t(\theta^{(0)})\end{aligned} \tag{11}$$

Combining Equation (10) with Equation (11),

$$\Psi^{(m)}_t(z^{(m-1)}, \theta^{(m-1)}) \le \sum_{i=1}^{m}\left[\Psi^{(i)}_t(z_t{}^{(i-1)}, \theta^{(i-1)}_t) + \mathcal{L}^{(i)}_{\theta,t}(z_t{}^{(i-1)}, \theta^{(i-1)})\right] \tag{12}$$

Combining with Equation (4)

$$\ell(z^{(m)}) \leq \ell({z_t}^{(m)}) + \sum_{i=1}^{m}\left[\Psi_t^{(i)}({z_t}^{(i-1)}, \theta_t^{(i-1)}) + \mathcal{L}_{\theta,t}^{(i)}({z_t}^{(i-1)}, \theta^{(i-1)})\right] \tag{13}$$

Noting that $\Psi_t^{(i)}({z_t}^{(i-1)}, \theta_t^{(i-1)}) = 0$ for all $i \in [1,m]$ gives the desired upper-bound. Now we prove that $g_{z,t}^{(i)} = \frac{\partial \ell(z_t)}{\partial z^{(i-1)}}$ for $i \in [2,m]$. We will prove this by induction starting from $i = m$,
Base Case: For $i = m$,

$$\begin{aligned} g_{z,t}^{(m)} &= \frac{\partial \Psi_t^{(m)}({z_t}^{(m-1)}, \theta_t^{(m-1)})}{\partial z^{(m-1)}} = \frac{\partial \ell({z_t}^{(m)})}{\partial z^{(m)}} \frac{\partial f(z^{(m-1)}, \theta^{(m-1)})}{\partial z^{(m-1)}} \\ &= \frac{\partial \ell({z_t}^{(m)})}{\partial z^{(m)}} \frac{\partial {z_t}^{(m)}}{\partial z^{(m-1)}} = \frac{\partial \ell({z_t}^{(m)})}{\partial z^{(m-1)}} \end{aligned}$$

Inductive Hypothesis: Assuming the statement is true for $i+1$ i.e. $g_{z,t}^{(i+1)} = \frac{\partial \ell(z_t)}{\partial z^{(i)}}$, let us prove it for i.

$$\begin{aligned} g_{z,t}^{(i)} &= \frac{\partial \Psi_t^{(i)}({z_t}^{(i-1)}, \theta_t^{(i-1)})}{\partial z^{(i-1)}} = g_{z,t}^{(i+1)} \frac{\partial f(z^{(i-1)}, \theta^{(i-1)})}{\partial z^{(i-1)}} = g_{z,t}^{(i+1)} \frac{\partial {z_t}^{(i)}}{\partial z^{(i-1)}} \\ &= \frac{\partial \ell(z_t)}{\partial z^{(i)}} \frac{\partial {z_t}^{(i)}}{\partial z^{(i-1)}} = \frac{\partial \ell(z_t)}{\partial z^{(i-1)}} \end{aligned}$$

which completes the induction.
It remains to be proven that $g_{\theta,t}^{(i)} = \frac{\partial h(\theta_t)}{\partial \theta^{(i-1)}}$ for $i \in [2,m]$. Given that we proved $g_{z,t}^{(i)} = \frac{\partial \ell(z_t)}{\partial z^{(i-1)}}$ for each i,

$$\begin{aligned} g_{\theta,t}^{(i)} &= \frac{\partial \Psi_t^{(i)}({z_t}^{(i-1)}, \theta_t^{(i-1)})}{\partial \theta^{(i-1)}} = g_{z,t}^{(i+1)} \frac{\partial f(z^{(i-1)}, \theta^{(i-1)})}{\partial \theta^{(i-1)}} = g_{z,t}^{(i+1)} \frac{\partial {z_t}^{(i)}}{\partial \theta^{(i-1)}} = \frac{\partial \ell(z_t)}{\partial z^{(i)}} \frac{\partial {z_t}^{(i)}}{\partial \theta^{(i-1)}} \\ &= \frac{\partial \ell(z_t)}{\partial \theta^{(i-1)}} = \frac{\partial h(\theta_t))}{\partial \theta^{(i-1)}} \end{aligned}$$

■

Theorem 1 *Assuming that (i) $h(\theta) = \ell(z^m)$ is bounded from below, (ii) $\ell(z^{(m)})$ is $L_z^{(m)}$ smooth, (iii) local losses $\mathcal{L}_{\theta,t}^{(i)}$ are β smooth for all $i \in [1,m], t \in [T]$, T iterations of* SM *with $\eta_z^{(m)} \leq \frac{1}{L_z^{(m)}}$ and N inner-loops in each iteration results in the following convergence rate,*

$$\min_{t \in [T]} \|\nabla h(\theta_t)\|_2^2 \leq \frac{2\beta\left[h(\theta_0) - h(\theta^*)\right]}{T}$$

Proof Using the same notation as in Theorem 1, since $\mathcal{L}_{\theta,t}^{(i)}({z_t}^{(i-1)}, \theta^{(i-1)})$ is β smooth for all $i \in [1,m]$, using GD with step-size equal to $\frac{1}{\beta}$ results in the descent at iteration $k \in [N-1]$. Specifically, for layer i, for iterates ω_k and ω_{k+1} in the inner-loop, where $\omega_0 = \theta_t^{(i-1)}$ and $\theta_{t+1}^{(i-1)} = \omega_N$,

$$\mathcal{L}_{\theta,t}^{(i)}({z_t}^{(i-1)}, \omega_{k+1}) \leq \mathcal{L}_{\theta,t}^{(i)}({z_t}^{(i-1)}, \omega_k) - \frac{1}{2\beta}\left\|\nabla \mathcal{L}_{\theta,t}^{(i)}({z_t}^{(i-1)}, \omega_k)\right\|_2^2$$

After N steps, for all $i \in [1, m]$

$$\mathcal{L}^{(i)}_{\theta,t}({z_t}^{(i-1)}, \omega_N) \leq \mathcal{L}^{(i)}_{\theta,t}({z_t}^{(i-1)}, \omega_0) - \frac{1}{2\beta} \sum_{k=0}^{N-1} \left\| \nabla \mathcal{L}^{(i)}_{\theta,t}({z_t}^{(i-1)}, \omega_k) \right\|_2^2$$

$$\begin{aligned} \implies \mathcal{L}^{(i)}_{\theta,t}({z_t}^{(i-1)}, \theta^{(i-1)}_{t+1}) \leq \mathcal{L}^{(i)}_{\theta,t}({z_t}^{(i-1)}, \theta^{(i-1)}_{t}) &- \frac{1}{2\beta} \left\| \nabla \mathcal{L}^{(i)}_{\theta,t}({z_t}^{(i-1)}, \theta^{(i-1)}_{t}) \right\|_2^2 \\ &- \sum_{k=1}^{N-1} \left\| \nabla \mathcal{L}^{(i)}_{\theta,t}({z_t}^{(i-1)}, \omega_k) \right\|_2^2 \end{aligned}$$

$$\implies \mathcal{L}^{(i)}_{\theta,t}({z_t}^{(i-1)}, \theta^{(i-1)}_{t+1}) \leq \mathcal{L}^{(i)}_{\theta,t}({z_t}^{(i-1)}, \theta^{(i-1)}_{t}) - \frac{1}{2\beta} \left\| \nabla \mathcal{L}^{(i)}_{\theta,t}({z_t}^{(i-1)}, \theta^{(i-1)}_{t}) \right\|_2^2$$

Recall that,
$\mathcal{L}^{(i)}_{\theta,t}({z_t}^{(i-1)}, \theta^{(i-1)}) = \langle g^{(i)}_{\theta,t}, \theta^{(i-1)} - \theta^{(i-1)}_t \rangle + \frac{1}{2} \left\| \theta^{(i-1)} - \theta^{(i-1)}_t \right\|^2_{\mathbb{H}^{(i)}_{\theta\theta,t}} + \frac{2\rho^{(i)}_t}{3} \left\| \theta^{(i-1)} - \theta^{(i-1)}_t \right\|^3$.
Hence,

$$\begin{aligned} \nabla \mathcal{L}^{(i)}_{\theta,t}({z_t}^{(i-1)}, \theta^{(i-1)}_{t}) &= g^{(i)}_{\theta,t} = \frac{\partial h(\theta_t)}{\partial \theta^{(i-1)}} \\ \implies \mathcal{L}^{(i)}_{\theta,t}({z_t}^{(i-1)}, \theta^{(i-1)}_{t+1}) \leq \mathcal{L}^{(i)}_{\theta,t}({z_t}^{(i-1)}, \theta^{(i-1)}_{t}) &- \frac{1}{2\beta} \left\| \frac{\partial h(\theta_t)}{\partial \theta^{(i-1)}} \right\|_2^2 = -\frac{1}{2\beta} \left\| \frac{\partial h(\theta_t)}{\partial \theta^{(i-1)}} \right\|_2^2 \\ &\text{(Since } \mathcal{L}^{(i)}_{\theta,t}({z_t}^{(i-1)}, \theta^{(i-1)}_{t}) = 0) \end{aligned}$$

Setting $z = z_{t+1}$, $\theta = \theta_{t+1}$ in the upper-bound from Theorem 1.

$$\begin{aligned} \ell(z^{(m)}_{t+1}) &\leq \ell({z_t}^{(m)}) + \sum_{i=1}^{m} \mathcal{L}^{(i)}_{\theta,t}({z_t}^{(i-1)}, \theta^{(i-1)}_{t+1}) \\ &= \ell({z_t}^{(m)}) - \frac{1}{2\beta} \sum_{i=1}^{m} \left\| \frac{\partial h(\theta_t)}{\partial \theta^{(i-1)}} \right\|_2^2 = \ell({z_t}^{(m)}) - \frac{1}{2\beta} \left\| \nabla h(\theta_t) \right\|_2^2 \\ \implies \left\| \nabla h(\theta_t) \right\|_2^2 &\leq 2\beta \left[h(\theta_t) - h(\theta_{t+1}) \right] \end{aligned}$$

Summing from $t = 0$ to $T - 1$ and telescoping

$$\sum_{t=0}^{T-1} \left\| \nabla h(\theta_t) \right\|_2^2 \leq 2\beta \left[h(\theta_0) - h(\theta_{T+1}) \right] \leq 2\beta \left[h(\theta_0) - h(\theta^*) \right]$$

Dividing by T, and using that $\min_{t \in [T]} \left\| \nabla h(\theta_t) \right\|_2^2 \leq \frac{\sum_{t=0}^{T-1} \left\| \nabla h(\theta_t) \right\|_2^2}{T}$ concludes the proof. ■

Appendix C. Helper Lemmas

Lemma 1 *For any vector $b = [b_u b_v]$ and matrix $A = \begin{bmatrix} A_{uu} & A_{uv} \\ A_{uv}^\mathsf{T} & A_{vv} \end{bmatrix}$ and coefficients c_1, c_2, c_3, function*

$$\Psi(u,v) := c_1 \left\langle \begin{bmatrix} b_u \\ b_v \end{bmatrix}, \begin{bmatrix} u \\ v \end{bmatrix} \right\rangle + c_2 \begin{bmatrix} u \\ v \end{bmatrix}^\mathsf{T} \begin{bmatrix} A_{uu} & A_{uv} \\ A_{uv}^\mathsf{T} & A_{vv} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} + c_3 \left\| \begin{bmatrix} u \\ v \end{bmatrix} \right\|^3 ,$$

that is dependent on both u, v can be upper-bounded in terms of functions $\mathcal{L}(u)$ and $\mathcal{L}(u)$.

$$\Psi(u,v) \leq \mathcal{L}(u) + \mathcal{L}(v) \quad \textit{where,} \quad \begin{aligned} \mathcal{L}(u) &:= c_1 \langle b_u, u \rangle + c_2 \|u\|_{B_{uu}}^2 + 4c_3 \|u\|^3 \\ \mathcal{L}(v) &:= c_1 \langle b_v, v \rangle + c_2 \|v\|_{B_{vv}}^2 + 4c_3 \|v\|^3 , \end{aligned}$$

where $B = \begin{bmatrix} B_{uu} & 0 \\ 0 & B_{vv} \end{bmatrix}$ such that (i) $B_{uu} \geq A_{uu}$; $B_{vv} \geq A_{vv} + A_{uv}^\mathsf{T} [B_{uu} - A_{uu}]^{-1} A_{uv}$.

Proof Note that we can easily split the first-order and third-order terms, i.e.

$$\left\langle \begin{bmatrix} b_u \\ b_v \end{bmatrix}, \begin{bmatrix} u \\ v \end{bmatrix} \right\rangle = \langle b_u, u \rangle + \langle b_v, v \rangle \quad ; \quad c \left\| \begin{bmatrix} u \\ v \end{bmatrix} \right\|^3 = \left(\sqrt{\|u\|_2^2 + \|v\|_2^2} \right)^3$$

$$\leq (\|u\| + \|v\|)^3 \leq 4\|u\|^3 + 4\|v\|^3$$

$$\text{(For } a, b \geq 0,\ (a+b)^p \leq 2^{p-1}(a^p + b^p))$$

In order to split the second-order term, we will find a block-diagonal matrix $B = \begin{bmatrix} B_{uu} & 0 \\ 0 & B_{vv} \end{bmatrix}$ such that $A \leq B$. Hence, we want that $B - A \geq 0$, implying that we want that,

$$\begin{bmatrix} B_{uu} - A_{uu} & -A_{uv} \\ -A_{uv}^\mathsf{T} & B_{vv} - A_{vv} \end{bmatrix} \geq 0$$

Note that a block matrix $X = \begin{bmatrix} X_{11} & X_{21} \\ X_{21}^\mathsf{T} & X_{22} \end{bmatrix}$ is positive semi-definite, iff (i) $X_{11} \geq 0$ and (ii) its Schur complement equal to $X_{22} - X_{21}^\mathsf{T} X_{11}^{-1} X_{21} \geq 0$. Hence, we want that,

$$B_{uu} \geq A_{uu} \quad ; \quad B_{vv} \geq A_{vv} + A_{uv}^\mathsf{T} [B_{uu} - A_{uu}]^{-1} A_{uv}$$

Given such a B, we can now split and upper-bound the second-order term as follows:

$$\begin{bmatrix} u \\ v \end{bmatrix}^\mathsf{T} \begin{bmatrix} A_{uu} & A_{uv} \\ A_{uv}^\mathsf{T} & A_{vv} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \leq \begin{bmatrix} u \\ v \end{bmatrix}^\mathsf{T} \begin{bmatrix} B_{uu} & 0 \\ 0 & B_{vv} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \|u\|_{B_{uu}}^2 + \|v\|_{B_{vv}}^2$$

Putting everything together,

$$\begin{aligned} \Psi(u,v) &:= \left\langle \begin{bmatrix} b_u \\ b_v \end{bmatrix}, \begin{bmatrix} u \\ v \end{bmatrix} \right\rangle + \begin{bmatrix} u \\ v \end{bmatrix}^\mathsf{T} \begin{bmatrix} A_{uu} & A_{uv} \\ A_{uv}^\mathsf{T} & A_{vv} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} + \left\| \begin{bmatrix} u \\ v \end{bmatrix} \right\|^3 \\ &\leq \underbrace{\langle b_u, u \rangle + \|u\|_{B_{uu}}^2 + 4\|u\|^3}_{:=\mathcal{L}(u)} + \underbrace{\langle b_v, v \rangle + \|v\|_{B_{vv}}^2 + 4\|v\|^3}_{:=\mathcal{L}(v)} \end{aligned}$$

■

Appendix D. Additional Quantitative Results

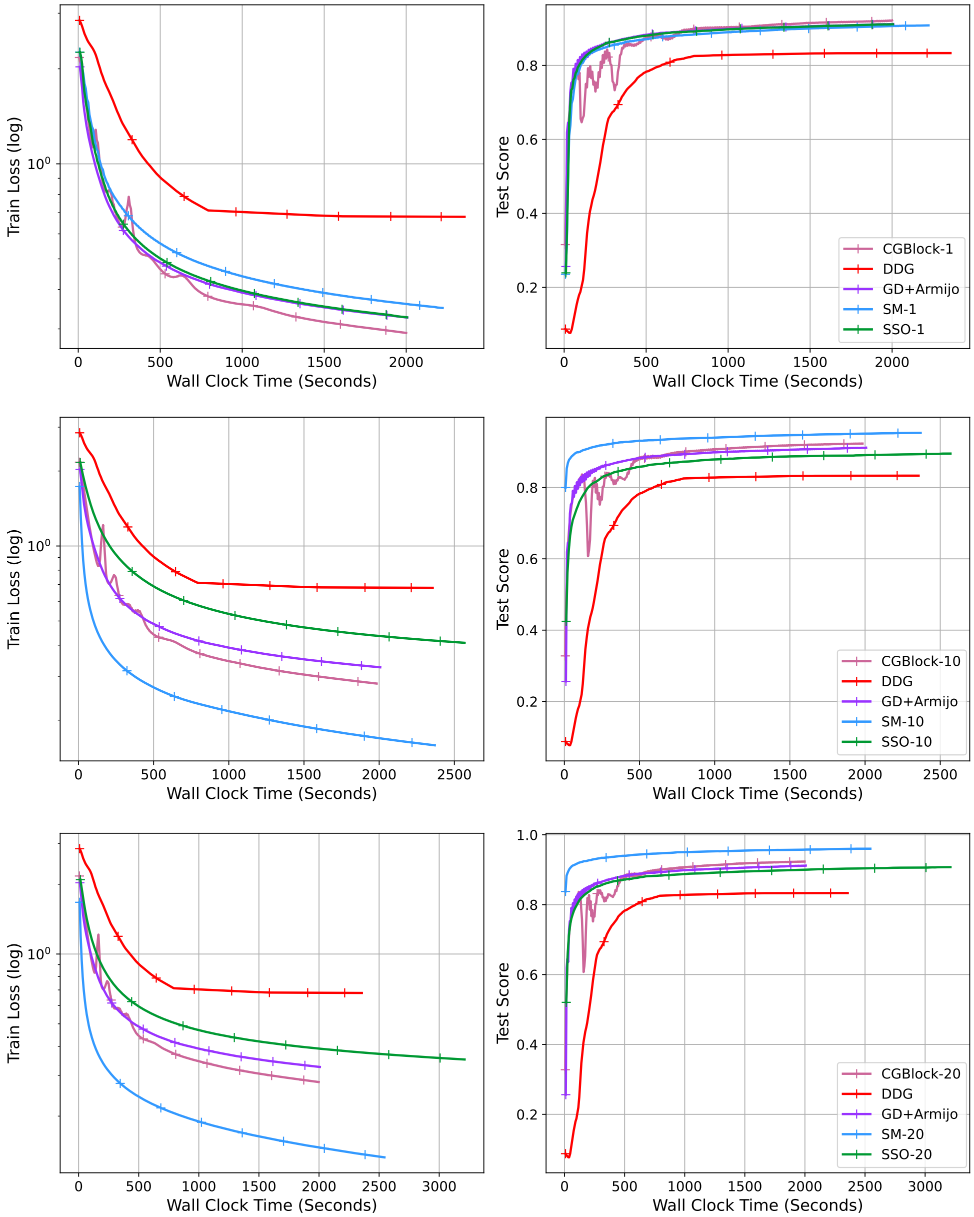

Table 2: Comparing classification results on MNIST for SM vs baselines using 2 layers of MLP. The rows correspond to 1, 10, and 20 iterations.

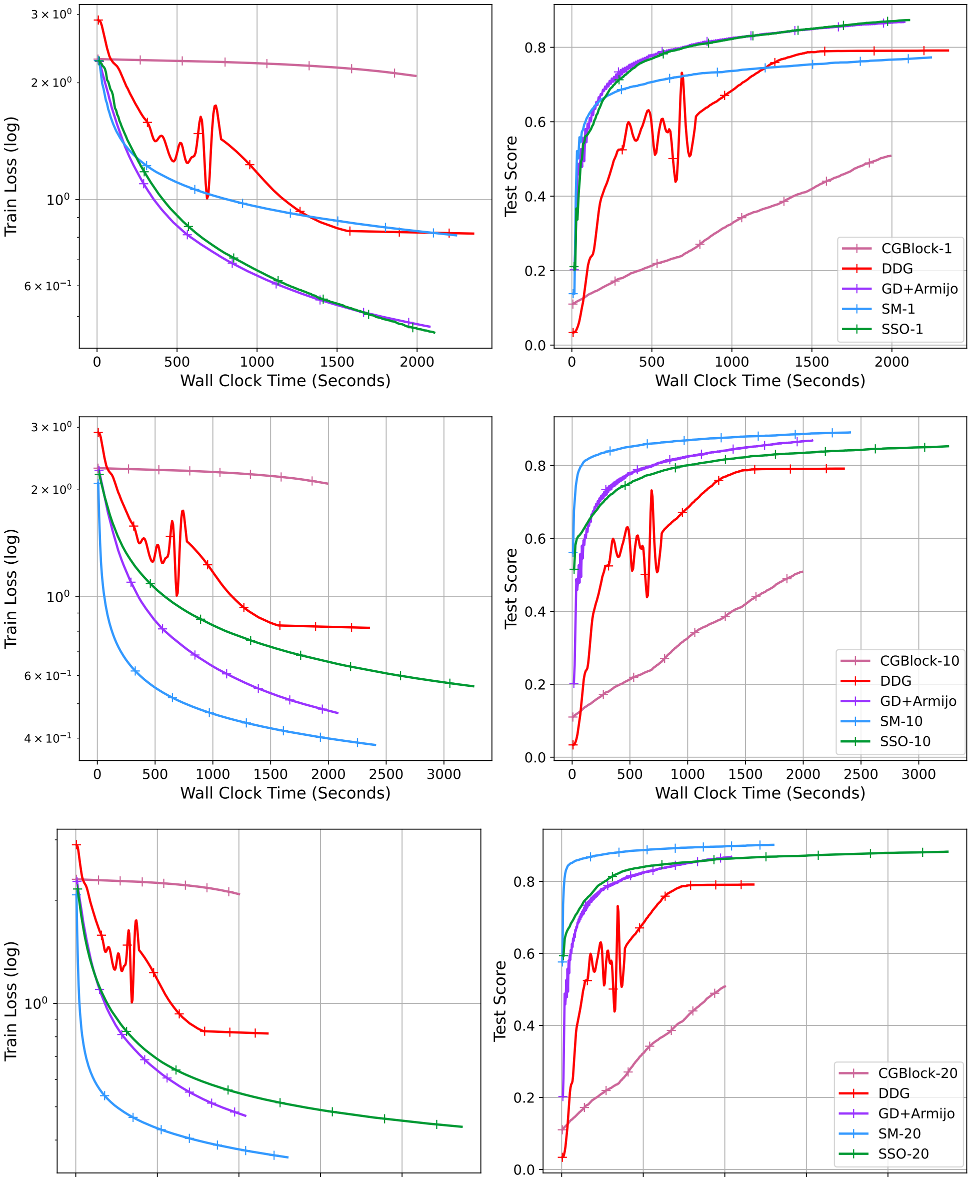

Table 3: Comparing classification results on MNIST for SM vs baselines using 4 layers of MLP. The rows correspond to 1, 10, and 20 iterations.

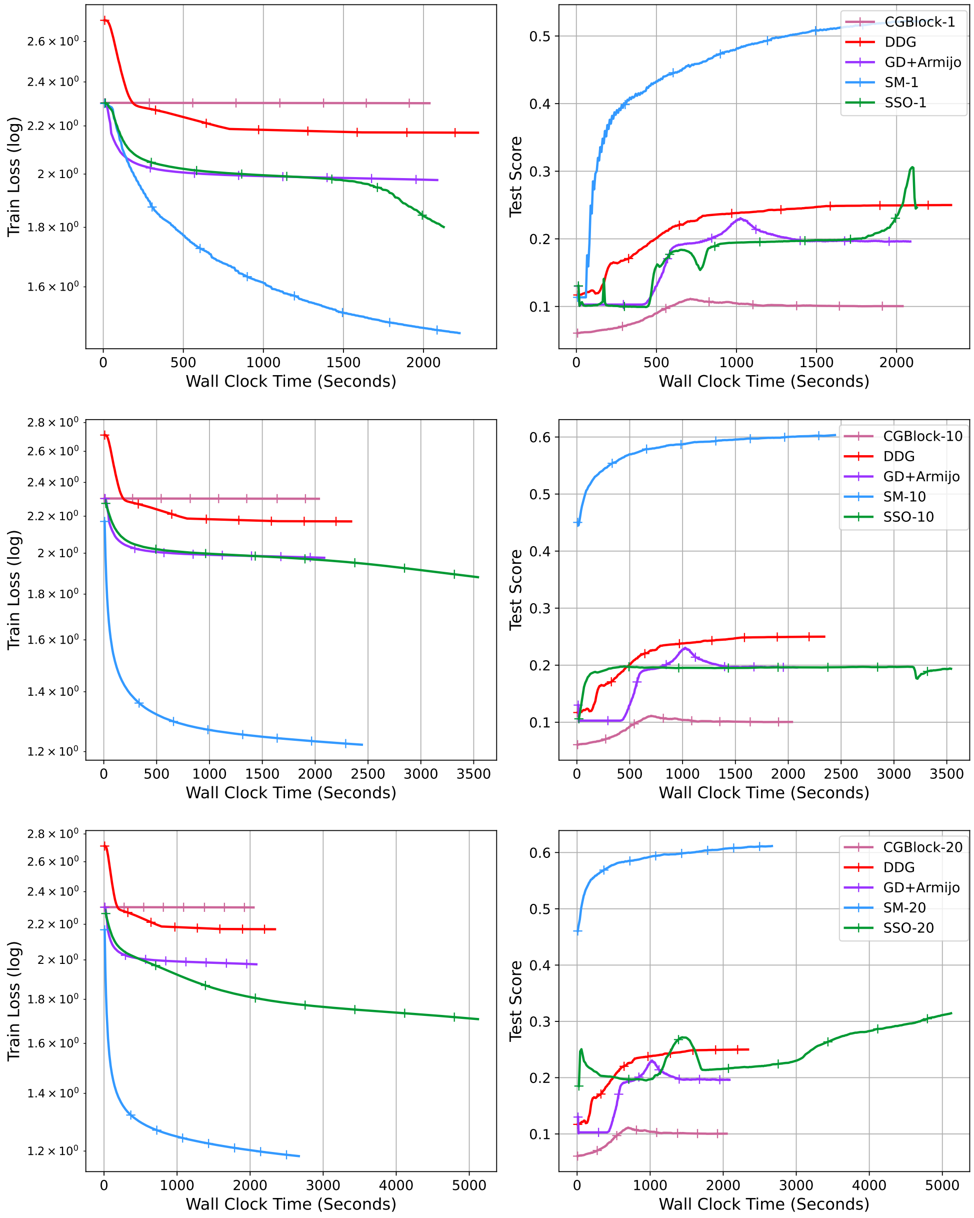

Table 4: Comparing classification results on MNIST for SM vs baselines using 8 layers of MLP. The rows correspond to 1, 10, and 20 iterations.