Masked Training of Neural Networks with Partial Gradients

Amirkeivan Mohtashami
EPFL

Martin Jaggi
EPFL

Sebastian U. Stich
CISPA[†]

Abstract

State-of-the-art training algorithms for deep learning models are based on stochastic gradient descent (SGD). Recently, many variations have been explored: perturbing parameters for better accuracy (such as in Extragradient), limiting SGD updates to a subset of parameters for increased efficiency (such as meProp) or a combination of both (such as Dropout). However, the convergence of these methods is often not studied in theory.
We propose a unified theoretical framework to study such SGD variants—encompassing the aforementioned algorithms and additionally a broad variety of methods used for communication efficient training or model compression. Our insights can be used as a guide to improve the efficiency of such methods and facilitate generalization to new applications. As an example, we tackle the task of jointly training networks, a version of which (limited to sub-networks) is used to create Slimmable Networks. By training a low-rank Transformer jointly with a standard one we obtain superior performance than when it is trained separately.

1 Introduction

Deep learning models are being used successfully in an increasing variety of scenarios. Consequently, new variants of SGD—the traditional training method for neural networks—have been designed, encouraged by various goals such as improving training efficiency (Sun et al., 2017), improving final model's accuracy (Korpelevich, 1976), or imposing additional properties on the trained model (e.g. being low rank Wang et al., 2021). In distributed training, compressed versions of gradients are used in order to reduce communication costs (Alistarh et al., 2017). Dropout (Srivastava et al., 2014), a method that computes the gradient after deactivating a random subset of neurons in the network, is widely used and is known to yield models with better performance. The Slimmable Nets approach (Yu and Huang, 2019b; Yu et al., 2019a) allows adjusting the *width* of a neural network dynamically during training and inference. While extensive literature exists on theoretical convergence bounds of SGD (Bottou et al., 2018; Stich et al., 2021; Stich, 2019), variants of SGD, such as those we discussed earlier, often lack a theoretical analysis even though they perform well in practice. However, theoretical understanding of a method can significantly ease and guide its adaptation to different tasks as well as its combination with other methods.

In this paper, we propose a unified theoretical framework to study variants of SGD that rely on a combination of parameter perturbation and gradient masking. In particular, we theoretically analyze convergence of these algorithms under a generic template called partial SGD and identify a set of criteria that when satisfied yield a theoretical convergence guarantee.

Our result applies to a wide range of existing methods such as the Extragradient method (Korpelevich, 1976), Dropout (Srivastava et al., 2014), extensions of Slimmable Nets (Yu et al., 2019a), partial backpropagation methods such as meProp (Sun et al., 2017) and parameter freezing methods such as layerwise training (Bengio et al., 2007), wide-and-deep networks (Cheng et al., 2016), weight pruning (Lin et al., 2019), as well as distributed independent subnet training (Yuan et al., 2019), among others. We evaluate some of these methods based on our criteria and obtain rigorous convergence rates as well as insights on how these algorithms might be further improved.

More importantly, our results can also be used as a guideline to create new training methods (e.g. to impose special properties on the final model) since design

[†]CISPA Helmholz Center for Information Security. Most research carried out while at EPFL.

Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151. Copyright 2022 by the author(s).

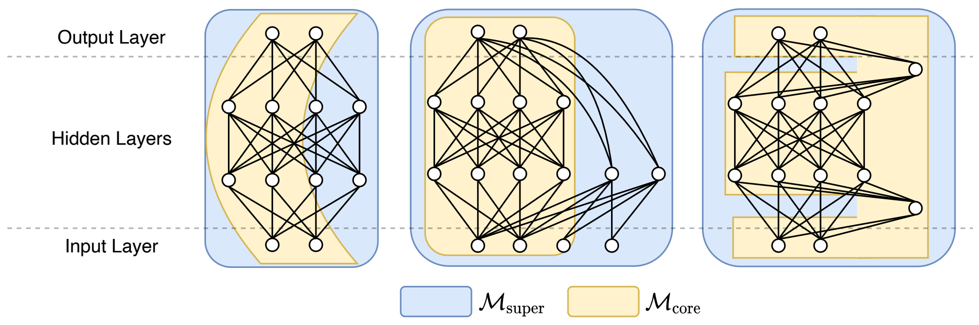


Figure 1: Illustrations of relevant compositions of a core network $\mathcal{M}_{\text{core}}$ (yellow) with a larger network $\mathcal{M}_{\text{super}}$ (blue). The left figure corresponds to the *slimmable* case of $\mathcal{M}_{\text{core}}$ being a small-width sub-network of $\mathcal{M}_{\text{super}}$. The middle example additively combines the output of two networks of different shape, matching *wide-and-deep networks* (Cheng et al., 2016). The right architecture shows the a more complex scheme where several fully-connected layers are replaced by their low-rank variant in $\mathcal{M}_{\text{core}}$, which are combined in $\mathcal{M}_{\text{super}}$ by adding the outputs layer-wise, while sharing the weights for the remaining layers.

choices can guarantee satisfaction of said criteria. As an example, we propose a training scheme to address the fundamental task of simultaneous training of two networks—a core network together with a larger architecture containing it. This includes the case of joint training of arbitrary combinations formed by two different neural networks, in which case the larger network is the union of the two. Figure 1 illustrates several example cases. Interestingly our designed method is an unrolled version of slimmable training. However, whereas slimmable training was limited to training sub-networks with reduced width we apply our method to more generalized scenarios. Joint training can have various motivations including but not limited to allowing weight sharing across different networks, improving training speed, or improving final accuracy.

To demonstrate the practical effectiveness of our method, we train a low-rank core network for a Transformer architecture (Vaswani et al., 2017) together with the larger original architecture. As a result of joint training, the trained low-rank model has superior stand-alone performance than when it is trained alone. The gain is particularly pronounced for significant model size reductions (32x rank reduction) where computational cost savings are highest (17.95 vs. increase to 24.03 BLEU score). Furthermore, the larger architecture can be converted to a full-rank standard Transformer matching the performance of the highly-optimized original Transformer model.

Contributions Our main contributions can be summarized as follows:

- We provide a rigorous theoretical analysis of partial SGD—a generic template encompassing training schemes that involve a combination of parameter perturbation and gradient masking.
- We present and discuss a host of existing methods that fit our theoretical framework. Our detailed discussion illustrates how our insights can guide future design decisions or lead to improvements for existing methods (showcased by designing a training method that jointly trains a full model and a reduced capacity model of much smaller size).
- We generalize slimmable training to go beyond just reducing width but allowing arbitrary sub-networks, and show the effectiveness of joint training in various practical scenarios, including training Transformer models together with low-rank variants on NLP tasks.

2 Related Work

The theoretical framework developed in this work covers a broad set of training methods used in a wide variety of settings. In distributed training, a worker can be responsible for optimizing a subnetwork while the local updates are expected to contribute to optimization of the full network (Yuan et al., 2019). Even in the data-parallel setting where each worker optimizes the full model but on its own local dataset, a compressed (masked) version of gradient is often used for communication efficiency (Aji and Heafield, 2017; Dryden et al., 2016). A similar technique is applied in (Sun et al., 2017) to improve speed of single-node training. Moreover, certain parameter perturbations seem to be effective in improving a model's ability to generalize, such as in extragradient methods (Korpelevich, 1976). A combination of masking and parameter perturbation can be seen in Dropout (Srivastava et al., 2014), a widely used technique which also fits our framework.

In this work, we show convergence bounds for algorithms that conform to our partial SGD template in smooth non-convex settings which is widely used in the SGD convergence literature (Arjevani et al., 2020; Bottou et al., 2018). The convergence of such algorithms is also investigated under NTK assumptions (Jacot et al.,

2021) for shallow neural networks (Liao and Kyrillidis, 2021) but only for random masking applied for both gradient computation and masking. In contrast, we allow arbitrary parameter perturbations for gradient computation, as well as arbitrary gradient masking which might not be random nor necessarily corresponding to said perturbation. Other works investigate convergence of instances of our generic template, such as Dropout (Jacot et al., 2021; Senen-Cerda and Sanders, 2020a,b), or gradient compression (Alistarh et al., 2018; Stich and Karimireddy, 2020; Stich et al., 2018).

Our framework can also be used to guide the design of new methods. We illustrate this by designing a method for joint training. The resulting method resembles an unrolled version of Slimmable training (Yu and Huang, 2019b; Yu et al., 2019a). Slimmable networks are trained such that the width can be chosen dynamically based on available resources during inference. Dynamic sparsity neural networks (Wu et al., 2021) extend this paradigm to subnetworks with different levels of sparsity, allowing removal of individual weights instead of neurons. In separate work simultaneous to ours, Peste et al. (2021) apply a similar method, also used in (Jin et al., 2016), to train sparse models. They theoretically prove convergence in the case when sparse minimizers exist, such as in overparameterized networks. While we only perturb the parameters for computing the gradient vector and apply the updates on the original values, their method applies the perturbation permanently; zeroing the masked parameters whereas we keep them unchanged. Moreover, our setting is more general since we allow perturbations other than masking.

An example of new applications considered in our work is training a network jointly with its low-rank variant. Previous work in this area includes adding regularization to the objective loss to encourage low-rank learning (Wen et al., 2017). Another approach is used in (Wang et al., 2021) to train the low-rank variant by initially using the full network to jump-start training. In contrast, our method continuously trains both the full network and the low-rank variant without any changes to the loss.

It is also possible to obtain a good small network from a larger network using methods such as knowledge distillation (Hinton et al., 2015; Lin et al., 2020) or model pruning (Han et al., 2015; Lin et al., 2019; Zhu and Gupta, 2017). Distillation improves the training of a student network (here the small network) by using the outputs of a teacher network (here the larger network). However, this requires training the small and large networks separately, which is more costly than our method. On the other hand, model pruning methods allow extracting a small network with good performance from a large network. While these methods can be fast and sometimes remove the need for training the small network, it is usually not possible to impose a structure on the small network. In contrast, our method allows a specifying a precise pre-defined architecture as the small network. A different line of work, looks for lottery ticket sub-networks in a normally trained network that can be used separately or trained from scratch to obtain the same performance (Frankle and Carbin, 2018; Frankle et al., 2020). However, these sub-networks often lack a hardware-compatible structure that can hinder obtaining a performance boost. In contrast, joint training can intuitively be seen as shaping the lottery ticket.

Algorithm 1 PARTIAL SGD

```
1: for $t = 0, \dots, T$ do
2:     Choose a mask $\mathbf{p}_t$, and
3:     a perturbation $\delta\mathbf{x}_t$ for the current step $t$
4:     $\tilde{\mathbf{x}}_t \leftarrow \mathbf{x}_t + \delta\mathbf{x}_t$                ▷ perturb parameters
5:     Take a stochastic gradient $\mathbf{g}_t$ of $\nabla f(\tilde{\mathbf{x}}_t)$
6:     $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t - \gamma_t \mathbf{p}_t \odot \mathbf{g}_t$
7: end for
```

Previous work successfully use joint training for neural architecture search where multiple networks are trained at once and the right architecture is determined by comparing their performance (Cai et al., 2019; Yu and Huang, 2019a; Yu et al., 2020, 2019b). As an example of fusing two network together, Wide-and-deep networks (Cheng et al., 2016) consist of a wide and another deep but narrow network with the output being the sum of the two networks. The wide part is a linear model that e.g. uses carefully selected features as well as their cross-product as input, at a very low inference latency cost. On the other hand, the deep part works on the raw features which allows the wide part to memorize more complex patterns of interactions between features while the deep part can help generalization (Cheng et al., 2016). Guo et al. (2017) extend these networks by removing the need for manual feature engineering.

Finally, in another line of research, the implicit bias caused by applying these methods is investigated (Hooker et al., 2020). We leave further research of this aspect of our method to future work.

3 Convergence Analysis

We consider the optimization of the full network parameters $\mathbf{x}$. In formal terms, we consider finding the minima of the empirical loss $f\colon \mathbb{R}^d \to \mathbb{R}$:

$$f^\star := \min\nolimits_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x})\,. \tag{1}$$

3.1 Partial SGD

We start by introducing partial SGD (Algorithm 1), a generic template designed to fit various training algorithms that apply a combination of gradient masking, and parameter perturbation for gradient computation. In particular, we make the following changes compared to vanilla SGD:

1. We allow a time-dependent binary mask $\mathbf{p}_t$ in each step to be chosen arbitrarily from a set of binary masks $\mathcal{P} \subseteq \{0,1\}^d$. Selecting the all-one vector in every step recovers vanilla SGD. Another example is a method that picks a random mask each time.
2. We allow arbitrary perturbation $\delta\mathbf{x}_t$ of the current parameters for calculating the loss and its gradient. Naturally, setting $\delta\mathbf{x}_t = 0$ recovers vanilla SGD while setting $\delta\mathbf{x}_t = -(\mathbf{1} - \mathbf{p}_t) \odot \mathbf{x}_t$ would mean applying the gradient of the sub-network induced by the mask $\mathbf{p}_t$.

Partial SGD can represent various training schemes. As an example consider the case of applying dropout with probability μ over the network. In this case $\mathbf{p}_t$ is a random mask where each element (or each group of elements if the dropout is applied to neurons instead of individual weights) is one with probability μ. The perturbation $\delta\mathbf{x}_t$ should be set equal to $-(\mathbf{1}-\mathbf{p}_t)\odot\mathbf{x}_t$ (similar to alternating training scheme) to compensate for the masking of weights during forward propagation while $\mathbf{p}_t$ limits the parameter update during backward propagation.

3.2 Assumptions & Definitions

We focus on the non-convex setting and assume the function f to be L-smooth.

Assumption 1 (L-smoothness)**.** *The function $f\colon \mathbb{R}^d \to \mathbb{R}$ is differentiable and there exists a constant $L > 0$ such that:*

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\| \,, \qquad \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d \,. \tag{2}$$

We assume that for every point in $\mathbf{x} \in \mathbb{R}^d$ we can query a stochastic gradient $\mathbf{g}(\mathbf{x})$ of $f(\mathbf{x})$, that is

$$\mathbf{g}(\mathbf{x}) := \nabla f(\mathbf{x}) + \boldsymbol{\xi}(\mathbf{x}) \,, \tag{3}$$

for a zero-mean stochastic noise vector $\boldsymbol{\xi}\colon \mathbb{R}^d \to R^d$.

We assume that the noise is bounded. Since we are interested in masking the gradient for partial training, we use a modified assumption over the set of allowed masks $\mathcal{P} \subseteq \{0,1\}^d$.

Assumption 2 ((M,σ^2)-bounded noise over $\mathcal{P}$)**.** *There exist parameters $M \geq 0, \sigma^2 \geq 0$ such that for any mask $\mathbf{p} \in \mathcal{P}$:*

$$\mathbb{E}\|\mathbf{p} \odot \boldsymbol{\xi}(\mathbf{x})\|^2 \leq M\|\mathbf{p} \odot \nabla f(\mathbf{x})\|^2 + \sigma^2 \,, \qquad \forall \mathbf{x} \in \mathbb{R}^d \,.$$

Remark 1. *When $\mathcal{P} = \{\mathbf{1}\}$, the assumption becomes the same as the standard assumption used for analyzing SGD convergence in previous work (e.g. Bottou et al., 2018). When the size of $\mathcal{P}$ is larger, the assumption becomes stronger. For example, if $\mathcal{P} = \{\mathbf{1}, \mathbf{1}_{\mathcal{M}_{core}}\}$, where $\mathcal{M}_{core}$ is a sub-network, the norm of noise over the sub-network is separately bounded by the norm of the gradient of the sub-network parameters. In its strongest form, i.e. when $\mathcal{P}$ contains all the possible masks, the assumption implies that the noise over each parameter is separately bounded by the norm of that parameter's gradient which is still a reasonable assumption.*

3.3 Main Result

We state the following main convergence theorem for non-convex objectives:

Theorem 1. *Let Assumptions 1–2, hold, and let the stepsize $\gamma_t = \alpha_t\gamma_{base}$ in Algorithm 1 with $\gamma_{base} = \min\left\{\frac{1}{L(M+1)}, \frac{\varepsilon}{L\sigma^2}\right\}$ for any $\varepsilon > 0$ and with $\alpha_t = \min\{1, \frac{\langle \mathbf{p}_t\odot\nabla f(\mathbf{x}_t), \mathbf{p}_t\odot\nabla f(\tilde{\mathbf{x}}_t)\rangle}{\|\mathbf{p}_t\odot\nabla f(\tilde{\mathbf{x}}_t)\|^2}\}$, where $\mathbf{p}_t$ denotes the binary mask and $\tilde{\mathbf{x}}_t$ the perturbed parameters. Then,*

- *$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}_{\boldsymbol{\xi}}\alpha_t^2\|\mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t)\|^2 < \varepsilon$, after at most the following number of iterations T:*

$$\mathcal{O}\left(\frac{\sigma^2}{\varepsilon^2} + \frac{(M+1)}{\varepsilon}\right) \cdot LF_0 \,.$$

- *Let $q := \max_{t\in[T],\boldsymbol{\xi}} \Big(q_t := \frac{\|\nabla f(\mathbf{x}_t)\|}{\|\mathbf{p}_t\odot\nabla f(\mathbf{x}_t)\|} \cdot \max\left\{\frac{\|\mathbf{p}_t\odot\nabla f(\mathbf{x}_t)\|}{\|\mathbf{p}_t\odot\nabla f(\tilde{\mathbf{x}}_t)\|}, \frac{\|\mathbf{p}_t\odot\nabla f(\mathbf{x}_t)\|\|\mathbf{p}_t\odot\nabla f(\tilde{\mathbf{x}}_t)\|}{\langle \mathbf{p}_t\odot\nabla f(\mathbf{x}_t), \mathbf{p}_t\odot\nabla f(\tilde{\mathbf{x}}_t)\rangle}\right\}\Big)$. Then $\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}_{\boldsymbol{\xi}}\|\nabla f(\mathbf{x}_t)\|^2 < \varepsilon$ after at most the following number of iterations T:*

$$\mathcal{O}\left(\frac{q^4\sigma^2}{\varepsilon^2} + \frac{q^2(M+1)}{\varepsilon}\right) \cdot LF_0 \,,$$

where $F_0 := f(\mathbf{x}_0) - f^\star$ and we use $\boldsymbol{\xi}$ to denote the full sequence $\boldsymbol{\xi}(\mathbf{x}_0), \boldsymbol{\xi}(\mathbf{x}_1), \ldots, \boldsymbol{\xi}(\mathbf{x}_{T-1})$.

We highlight that the bounds proven by this theorem depend on the values of the stepsize scaling α_t and gradient alignment q_t. In particular, if the values of α_t are sufficiently large, the first part of the theorem shows convergence of the partially masked core networks. Moreover, sufficiently small q ensures convergence of the full network. Note that these two values are inversely proportional ($q_t = \frac{\|\nabla f(\mathbf{x}_t)\|}{\alpha_t\|\mathbf{p}_t\odot\nabla f(\tilde{\mathbf{x}}_t)\|}$) so that usually a large α_t results in a small q_t and vice-versa.

In reality, the values q_t and α_t also depend on the selection of masks which vary depending on training scheme and application. The above theorem holds under the most general set of assumptions for widest applicability. More informative upper bounds on the value q can be obtained in relevant applications. Note that

Theorem 1 naturally recovers the standard SGD convergence theorem (Bottou et al., 2018) as a special case, by setting $\mathbf{p}_t = \mathbf{1}_d$. In this case $\alpha_t = q_t = 1$ and we recover the best known SGD bounds.

We also remark that while the theoretical result assumes that the learning rate is multiplied by α_t, in practice it is not necessary to compute an exact value for α_t in each step. Instead, a lower bound α can be tuned as a hyper-parameter (which might also be different for different masks). Even simpler, in our experiments, we observed that no tuning is necessary, and using the same learning rate as for the standard training of the full network performs very well.

We leave the proof of the theorem (leveraging techniques from the proof of BiasedSGD (Ajalloeian and Stich, 2020)) to Appendix A. In the following we propose a set of criteria that can be used to ensure the value q for a training algorithm is small. As such they can be used both as a guide for developing new methods as well as proving convergence of existing methods.

Criterion 1 (Gradient norm preservation against masking)**.** *To ensure a small q value, the binary masks applied on the gradient vector should not drastically reduce its norm, i.e. the ratio $\frac{\|\nabla f(\mathbf{x}_t)\|}{\|\mathbf{p}_t \odot \nabla f(\mathbf{x}_t)\|} \leq c_{\text{norm}}$ should be bounded by a small constant c_{norm}. Note that this criterion is automatically satisfied with $c_{\text{norm}} = 1$ when no masking is done.*

Criterion 2 (Gradient norm preservation against perturbation)**.** *When applying perturbations before computing the gradient, the training algorithm should ensure that the perturbation does not lead to a drastic decrease in gradient norm, i.e., there should exist a constant c_{sim} such that $\frac{\|\mathbf{p}_t \odot \nabla f(\mathbf{x}_t)\|}{\|\mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t)\|} \leq c_{\text{sim}}$. This holds with $c_{\text{sim}} = 1$ when no perturbation is done.*

Criterion 3 (Gradient alignment)**.** *The gradient vectors computed with and without perturbation should be aligned. More formally, there should exist an upper bound c_{align} such that $\frac{\|\mathbf{p}_t \odot \nabla f(\mathbf{x}_t)\|\|\mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t)\|}{\langle \mathbf{p}_t \odot \nabla f(\mathbf{x}_t), \mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t)\rangle} \leq c_{\text{align}}$. This criterion also automatically holds with $c_{\text{align}} = 1$ when no perturbation is applied, i.e. when $\mathbf{x}_t = \tilde{\mathbf{x}}_t$.*

We will now example various use-cases of these criteria.

meProp meProp is a training scheme which only uses the top-k elements of the gradient vector for optimization (Sun et al., 2017). Since no perturbation is applied, Criteria 2 and 3 are satisfied. Choosing elements with the highest value in the gradient vector ensures Criterion 1 as the upper bound $c_{\text{norm}} \leq \frac{d}{k}$ holds. Note that if instead of proving the existing method we were looking for a method which only uses a portion of the gradient vector, this criterion would guide us to choose the elements with the highest value, guiding us toward the right method. For completeness, we point out that, in practice, meProp applies the top-k approximation layer-wise during the backward pass. This speeds up the back-propagation process but leads to using an incorrect gradient in the lower layers, thus only being an approximation of our variant using the actual gradient.

Extragradient Method In this method, the gradient is calculated after making a small step (with step size η_s) in the direction of the gradient, i.e. $\delta \mathbf{x}_t = \eta_s \nabla f(\mathbf{x}_t)$. Setting $\mathbf{p}_t = \mathbf{1}$, recovers the Extragradient method from Algorithm 1 which is a more generalized version that allows gradient masking as well as parameter perturbations that are not in the direction of the gradient. Since no masking is done, criterion 1 is automatically satisfied. In order to show the other criteria are also satisfied, we introduce the following assumption and lemma.

Assumption 3 (bounded perturbation)**.** *For a L-smooth function, i.e. satisfying Assumption 1, the training algorithm makes perturbations such that*

$$\max_{t \in [T]} \frac{\|\delta \mathbf{x}_t\|}{\max\left\{\|\mathbf{p}_t \odot \nabla f(\mathbf{x}_t)\|, \|\mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t)\|\right\}} < \frac{1}{2L}\,. \tag{4}$$

Note that by choosing a small perturbation coefficient, in particular $\eta_s \leq \frac{1}{2L}$, the assumption holds for the extragradient method since $\|\delta \mathbf{x}_t\| = \eta_s \|\nabla f(\mathbf{x}_t)\|$. The following lemma guarantees the other criteria.

Lemma 2. *If Assumptions 1 and 3 hold, $c_{align} \leq \sqrt{10}$ and $c_{sim} \leq \frac{\sqrt{10}}{2}$.*

We postpone the proof of the above lemma to the Appendix B. Note that the above lemma guarantees $q \leq 10$ for the extragradient method yielding convergence bounds with the same complexity as previously known results (Xu et al., 2019). Assumption 3 can also be useful in other settings, especially because it can be verified during training. Therefore, it is possible to design training algorithms which determine the perturbation and the mask based on whether this assumption holds. We now provide one possible example in the settings of model parallel training.

Model-Parallel Training One of the recently introduced methods for model-parallel distributed training of a network across several workers is independent subnet training (Yuan et al., 2019). In this method, neurons in each layer of the network are randomly partitioned into k disjoint sets (equal to the number of workers), partitioning the whole network into k disjoint parts. Each worker trains one part of the network by independently performing SGD steps on its personal core part for a limited number of steps. Next, the new weights are communicated, and the network is again randomly re-partitioned, each worker receiving a new part of the network. From the perspective of a single worker, the training scheme matches the use of a fixed dropout mask for several steps, which exactly fits our Algorithm 1. Moreover, because of the disjoint partitioning, the result is identical if the worker steps were interleaved instead of being run in parallel. More specifically, if $\mathbf{p}_{i,k}$ corresponds with the mask for worker i at step k, the sequence $\mathbf{p}_t$ can be set to $\mathbf{p}_{1,1}, \mathbf{p}_{2,1}, \ldots, \mathbf{p}_{k,1}, \mathbf{p}_{1,2}, \ldots$. Hence, Theorem 1 can be used to analyze the convergence of this scheme. It might be possible to evaluate the satisfaction of the above criteria using theoretical arguments or practical measurements. However, here we want to point toward another application of these criteria. In particular, we note that currently the algorithm requires tuning the number of local steps on each worker. Another approach is to verify Assumption 3 and change the partitioning whenever it is violated. Note that this is possible with minimal communication (norm of the weights) between workers. Even the communication might not be necessary since the norm of the weights does not significantly change in a single partitioning. We leave the practical investigation into this as a future work.

Dropout In the case of dropout, each element of the mask $\mathbf{p}_t$ (or each group of elements corresponding to the weights connected to a specific neuron) is i.i.d. chosen from a Bernoulli distribution with mean μ. In this case, the expected value of $\frac{\|\nabla f(\mathbf{x}_t)\|^2}{\|\mathbf{p}_t \odot \nabla f(\mathbf{x}_t)\|^2}$ is $\frac{1}{\mu}$ which means Criterion 1 is on average satisfied. We use a practical approach to ensure the other criteria holds. Note that it might also be possible to show the other criteria are also satisfied in theory by making the right assumptions which we leave as a future work. We measure the values upper bounded in each criterion throughout training of a ResNet-18 on CIFAR10 with and without Dropout. The results are plotted in Figure 2. It can be seen that when applying dropout, both Criterion 2 and 3 hold in practice. We now provide possible explanations as to why this happens.

Assuming a positive correlation between gradient's norm and convergence, it is intuitive that replacing an optimized parameter with a random value (such as zero) would increase gradient norm. As a simple example, consider a 1-dimensional quadratic function $(x-a)^2$ where a is a large positive number. After some optimization steps, x will be close to a and replacing it with zero would increase the gradient's norm. While this is not a formal proof, it gives an intuition as to why Criterion 2 would be satisfied.

However, it is not directly clear why running SGD with dropout results in the last criterion to be satisfied. In

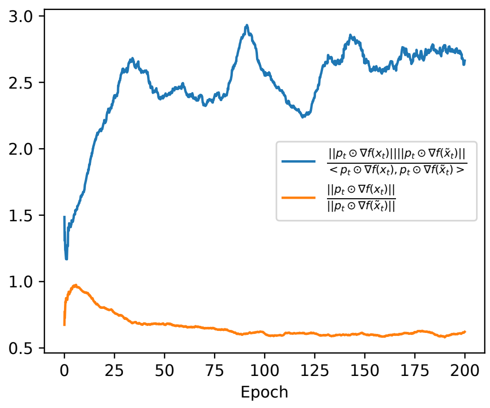


Figure 2: The values in Criterion 2 and 3 measured while training a ResNet-18 on CIFAR10 with dropout. It can be seen that these criteria are satisfied in practice.

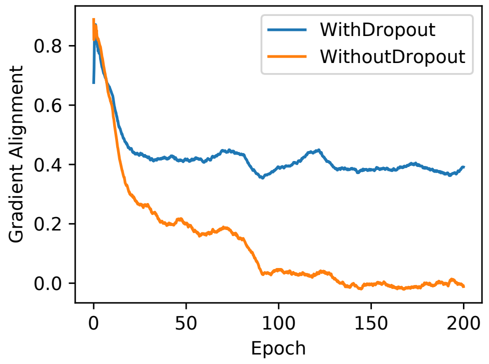


Figure 3: The value $\frac{\langle \mathbf{p}_t \odot \nabla f(\mathbf{x}_t), \mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t) \rangle}{\|\mathbf{p}_t \odot \nabla f(\mathbf{x}_t)\| \|\mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t)\|}$, where $\tilde{\mathbf{x}_t}$ is obtained by applying a random Dropout mask (masking neurons instead of weights), measured while training a ResNet-18 on CIFAR10 with and without dropout. Observably, applying dropout improves gradient alignment.

particular, Figure 3 shows that this criterion is not satisfied when running SGD without dropout. We hypothesize that this phenomena is related to the findings of previous work (Dandi et al., 2021; Nichol et al., 2018) which shows that SGD imposes an implicit regularization to increase the dot product of consecutive mini-batches. In particular, since the proof in (Nichol et al., 2018) allows for a different loss functions per mini-batch, the result is applicable to our case as well and means that SGD with dropout promotes alignment between gradients of random sub-networks. Intuitively, if the gradients of any pair of sub-networks are aligned with each other, it can be expected that they are also aligned with the full network, for example due to existence of sub-networks obtained as a result of pruning that are able to produce similar output as the full network.

3.4 Alternating Training Scheme

We now showcase how it is possible to use the criteria we introduced as guidelines for designing new methods. In particular, we design an algorithm for training

Algorithm 2 Alternating Training

```
1: for $t = 0, \dots, T$ do
2:     if $t$ is even then
3:         $\mathbf{p}_t \leftarrow \mathbf{1}_{\mathcal{M}_{\text{super}}}$        ▷ all-one vector
4:     else
5:         $\mathbf{p}_t \leftarrow \mathbf{1}_{\mathcal{M}_{\text{core}}}$        ▷ mask of core network
6:     end if
7:     Take stochastic gradient $\mathbf{g}_t$ of $\nabla f(\mathbf{p}_t \odot \mathbf{x}_t)$
8:     $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t - \gamma_t \mathbf{p}_t \odot \mathbf{g}_t$
9: end for
```

a given network $\mathcal{M}_{\text{super}}$, which we refer to as super-network, jointly with one of its sub-networks $\mathcal{M}_{\text{core}}$, which we refer to as the core network. First, note that any algorithm that trains the full network on some steps and the sub-network on other steps fits the partial SGD template since the parameters of $\mathcal{M}_{\text{core}}$ can be simply extracted by applying a binary mask $\mathbf{1}_{\mathcal{M}_{\text{core}}}$ over $\mathbf{x}$, i.e. $\mathbf{1}_{\mathcal{M}_{\text{core}}} \odot \mathbf{x}$. In order to choose between different switching patterns we compare them in terms of our criteria. In particular, relying on the implicit regularization of SGD that promotes alignment between consecutive mini-batches that we discussed in analyzing dropout, the maximum alignment should be obtained when consecutively switching between training $\mathcal{M}_{\text{core}}$ and $\mathcal{M}_{\text{super}}$. This yields Algorithm 2 which we refer to as Alternating Training Scheme (ATS). Note that here we focus on the case where $\mathcal{M}_{\text{core}}$ is predetermined. However, if this was not the case, Criterion 1 would point toward selecting the sub-network containing largest elements of the gradient vector.

ATS is clearly a special case of partial SGD using the same mask sequence $\mathbf{1}_{\mathcal{M}_{\text{super}}}, \mathbf{1}_{\mathcal{M}_{\text{core}}}, \mathbf{1}_{\mathcal{M}_{\text{super}}}, \dots$ as $\mathbf{p}_t$ and $\delta\mathbf{x}_t = -\mathbf{x}_t$ (thus zeroing weights not part of the core network). In order to evaluate ATS in terms of our criteria, we follow a similar approach as for Dropout and make the following observations based on training a Transformer jointly with its subnetwork with half the full width:

Large gradient overlap Let r be the ratio of parameters in the subnetwork to the number of parameters in the full network. For example, in our experiment $r = 0.5$. Given the random initialization the expected value for $\frac{\|\nabla f(\mathbf{x}_t)\|^2}{\|\mathbf{p}_t \odot \nabla f(\mathbf{x}_t\|^2}$ is $\frac{1}{r}$ when $t = 0$. Figure 4a shows that when training using ATS, this quantity remains low and does not increase during training. In fact, it can be seen that when using ATS, the overlap value is significantly improved compared to standard training.

Norm similarity Figure 4b shows that even when running SGD, the norm of the gradient does not decrease due to applying perturbation. In fact, applying ATS results in less increase in gradient norm as a result of perturbation. However, whether running SGD or ATS the ratio is upper bounded for example by 1.

Gradient alignment Finally, we measure $\frac{\|\mathbf{p}_t \odot \nabla f(\mathbf{x}_t)\| \|\mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t)\|}{\langle \mathbf{p}_t \odot \nabla f(\mathbf{x}_t), \mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t) \rangle}$ in practice and demonstrate that, as expected, using ATS lowers this value (thus improves alignment) fulfilling Criterion 3. The result is plotted Figure 4c.

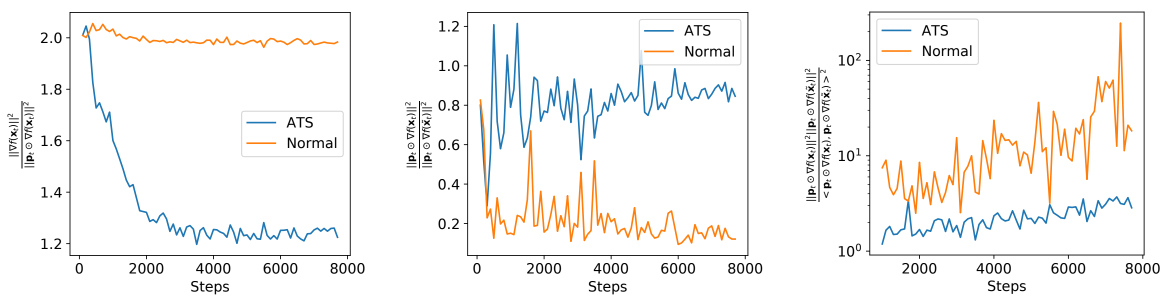
The convergence bound obtained for ATS from our observations is within a small constant of vanilla SGD. On the other hand, in Section 4, we observe that the same number of iterations as in standard training is typically sufficient in practice.

Applying ATS to small-width subnetworks recovers the setting of slimmable nets (Yu and Huang, 2019b; Yu et al., 2019a). While the original papers apply gradient updates of narrow and wide widths at the same timestep, our scheme unrolls this by updating the small and large parts sequentially. Therefore, the convergence result for ATS also informs the sequential version of slimmable nets, and proves that the q_t parameter characterizes the successful joint training. As we discussed earlier, the sequential version benefits from implicit SGD regularization to keep the gradients of the sub-network and full network aligned.

We also point out that while at first glance the algorithm seems tailored for training a network jointly with one of its sub-networks, the same method can be used to train a pair of arbitrary networks, by using one as the core network, while building the super network $\mathcal{M}_{\text{super}}$ to be the combination of the two. There are various ways to build $\mathcal{M}_{\text{super}}$. In its simplest form, this can be done by creating a super-network that has a copy of each network and outputs the sum of their outputs. Wide-and-deep networks are an example of using this simple combination. This can be seen by letting $\mathcal{M}_{\text{core}}$ correspond to the deep network while the wide network acts as the extension network. Using ATS in this setting can allow the wide part to become an extension to the deep part which can focus on solving cases where feature interactions are necessary, which was the intuition behind the design of these networks (Cheng et al., 2016). Furthermore, the deep part can be used as a standalone network when necessary.

A more sophisticated combination is summing the output of both models in all hidden layers, instead of just in the output layer. This is the combination scheme we use in our experiments for training a low-rank model. These cases are illustrated in Figure 1.

While we have so far focused on training two networks, the scheme can be extended to train several more networks as a hierarchy, to allow more flexibility and fine-grained control over the amount of resource usage.



(a) $\frac{\|\nabla f(\mathbf{x}_t)\|^2}{\|\mathbf{p}_t \odot \nabla f(\mathbf{x}_t)\|^2}$ (b) $\frac{\|\mathbf{p}_t \odot \nabla f(\mathbf{x}_t)\|^2}{\|\mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t)\|^2}$ (c) $\frac{\|\mathbf{p}_t \odot \nabla f(\mathbf{x}_t)\|^2 \|\mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t)\|^2}{\langle \mathbf{p}_t \odot \nabla f(\mathbf{x}_t), \mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t)\rangle^2}$

Figure 4: Measurement of quantities affecting convergence rate of Theorem 1 during standard training of a Transformer vs. joint training together with its subnetwork with half width using ATS. In all plots, a mini-batch gradient is used to estimate the mentioned quantities, instead of the full gradient.

4 Experiments

We consider the application of alternating training scheme in two use cases. In Section 4.1, we show the effectiveness of ATS by training a standard Transformer together with a Transformer with a low-rank architecture. We show that the low-rank model trained with ATS is able to match or in some cases even outperform training the low-rank variant separately. At the same time the full-rank obtains a comparable performance while the total training time is less than training both models separately. In Section 4.2, we show that ATS can also be used to train a Transformer together with its small-width sub-network with reduced width and observe similar results to the low-rank training. We trained our models using Adam and trained for 22 epochs on WMT17 (Bojar et al., 2017) and for 30 epochs on IWSLT14 (Cettolo et al., 2014). We report a detailed list of hyper-parameters in Appendix D.

4.1 Low-Rank Training

In this section we demonstrate how our alternating training scheme can be used to train a low-rank model. In particular, we build a super-model composed of a low-rank as well as a full-rank Transformer by replacing the linear transformations (e.g. fully connected layers) in the feed-forward modules and key/query projection in the attention layers of a standard Transformer, with low-rank fully connected (LFC) layer:

$$LFC(X) := (VU + W)X$$

where $U \in \mathbb{R}^{k \times f_{in}}$, $V \in \mathbb{R}^{f_{out} \times k}$, and $W \in \mathbb{R}^{f_{out} \times f_{in}}$ are the parameters. We set $k = r \min(f_{in}, f_{out})$ for all layers where r is the rank reduction ratio. The low-rank variant is the sub-network which only contains U and V parameters of each LFC layer as well as all other parameters in the super-network that are not in a LFC layer. Note that in this case, the super-network is equivalent to a full-rank variant and can be converted to a standard Transformer after the training.

Table 1: BLEU scores of Transformer trained jointly with its **low-rank variant** using ATS on the IWSLT14 (top) and WMT17 (bottom) machine translation datasets respectively. The sub-network corresponds to the low-rank variant with rank reduced by factor r. The baseline (full network with standard training) shows the performance of a standard Transformer. In other rows, the full network corresponds to the super-network which can be converted to a standard Transformer.

(a) IWSLT14 dataset.

Model	Sub-Network	Full Network
Standard Training	-	32.48 (0.32)
Standard Training ($r = 0.25$)	28.96 (0.04)	-
Alternating Training ($r = 0.25$)	29.15 (0.63)	31.34 (0.36)
Standard Training ($r = 0.125$)	27.87 (0.11)	-
Alternating Training ($r = 0.125$)	28.20 (0.48)	31.62 (0.13)
Standard Training ($r = 0.03125$)	17.95 (3.65)	-
Alternating Training ($r = 0.03125$)	**24.03 (0.77)**	31.20 (0.58)

(b) WMT17 dataset.

Model	Sub-Network	Full Network
Standard Training	-	26.41 (0.14)
Standard Training ($r = 0.03125$)	23.20 (0.09)	-
Alternating Training ($r = 0.03125$)	23.18 (0.12)	25.50 (0.22)

We compared the performance of the trained low-rank and full-rank models with the performance of the same model trained separately using the standard optimizer on IWSLT14 dataset in Table 1a. Observably, the low-rank model trained in our scheme has a superior performance for lower values of r while it matches the performance of standard training for larger r. At the same time, the full-rank model has a much higher performance near that of the standard training of the same model, showing the success of ATS in joint training. We repeated the same experiment on WMT17 and obtained similar results which we reported in Table 1b.

4.2 Small-Width Training

In order to show that our model can also be used for training a network together with its sub-network, we train a Transformer with its small-width variant. As in the low-rank settings, ATS obtains core networks with similar or even superior performance to solo training as well as super networks with comparable performance. We report our result in Appendix C and also compare per-step computation cost of our method with slimmable training and standard training in FLOPs. It can be seen that ATS is significantly more efficient than slimmable training and even standard training of the super model.

5 Future Work

Our theoretical framework is widely applicable to a variety of existing and novel training scenarios. Hence, it can be used to analyze other partial training schemes and gain insights useful to improve such schemes. For example, a future direction would be using the analysis done in this work on independent subnet training to allow dynamic switching between local training and global coordination automatically. We have also used theoretical insights of our results to design a joint training method for two arbitrary neural networks. While we applied our method to train Transformers on translation tasks, applications in other areas or for other architectures is grounds for future work. Finally, other training schemes can be designed relying on the proposed set of criteria identified in our work.

6 Conclusion

In this work, we developed a theoretical framework to analyze convergence of SGD variants that apply parameter perturbation and gradient masking, and showed that our settings cover a large number of scenarios involving partial training. Moreover, we presented a new simultaneous training scheme for arbitrary network pairs, and showed the efficacy of our scheme for training a Transformer together with its low-rank variant, and alternatively its subnetwork of reduced width.

Acknowledgement

This work was supported by Innosuisse Project Privately And by SNSF grant 200020_200342.

We thank Lie He and Anastasia Koloskova for their help in proofreading of the manuscript.

References

Ahmad Ajalloeian and Sebastian U. Stich. On the convergence of SGD with biased gradients. *arXiv preprint arXiv:2008.00051*, 2020.

Alham Fikri Aji and Kenneth Heafield. Sparse communication for distributed gradient descent. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017.

Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. QSGD: Communication-efficient SGD via gradient quantization and encoding. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Dan Alistarh, Torsten Hoefler, Mikael Johansson, Nikola Konstantinov, Sarit Khirirat, and Cedric Renggli. The convergence of sparsified gradient methods. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Yossi Arjevani, Ohad Shamir, and Nathan Srebro. A tight convergence analysis for stochastic gradient descent with delayed updates. In *Algorithmic Learning Theory*. PMLR, 2020.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, et al. Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2007.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, et al. Findings of the 2017 conference on machine translation (wmt17). In *Proceedings of the Second Conference on Machine Translation*, 2017.

L. Bottou, F. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, (2), 2018.

Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. In *International Conference on Learning Representations (ICLR)*, 2019.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. Report on the 11th iwslt evaluation campaign. In *Proceedings of the 11th International Workshop on Spoken Language Translation: Evaluation Campaign*, 2014.

Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, 2016.

Yatin Dandi, Luis Barba, and Martin Jaggi. Implicit gradient alignment in distributed and federated learning. *arXiv preprint arXiv:2106.13897*, 2021.

Nikoli Dryden, Tim Moon, Sam Ade Jacobs, and Brian Van Essen. Communication quantization for data-parallel training of deep neural networks. In *2016 2nd Workshop on Machine Learning in HPC Environments (MLHPC)*. IEEE, 2016.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations (ICLR)*, 2018.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning*. PMLR, 2020.

Huifeng Guo, Ruiming TANG, Yunming Ye, Zhenguo Li, and Xiuqiang He. DeepFM: A factorization-machine based neural network for CTR prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, 2017.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Sara Hooker, Nyalleng Moorosi, Gregory Clark, Samy Bengio, and Emily Denton. Characterising bias in compressed models. *arXiv preprint arXiv:2010.03058*, 2020.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: convergence and generalization in neural networks. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, 2021.

Xiaojie Jin, Xiaotong Yuan, Jiashi Feng, and Shuicheng Yan. Training skinny deep neural networks with iterative hard thresholding methods. *arXiv preprint arXiv:1607.05423*, 2016.

Galina M Korpelevich. The extragradient method for finding saddle points and other problems. *Matecon*, 1976.

Fangshuo Liao and Anastasios Kyrillidis. On the convergence of shallow neural network training with randomly masked neurons. *arXiv preprint arXiv:2112.02668*, 2021.

Tao Lin, Sebastian U Stich, Luis Barba, Daniil Dmitriev, and Martin Jaggi. Dynamic model pruning with feedback. In *International Conference on Learning Representations (ICLR)*, 2019.

Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Yurii Nesterov. *Introductory Lectures on Convex Optimization*, volume 87 of *Springer Science & Business Media*. Springer US, Boston, MA, 2004.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, 2019.

Alexandra Peste, Eugenia Iofinova, Adrian Vladu, and Dan Alistarh. Ac/dc: Alternating compressed/decompressed training of deep neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Albert Senen-Cerda and Jaron Sanders. Almost sure convergence of dropout algorithms for neural networks. *arXiv preprint arXiv:2002.02247*, 2020a.

Albert Senen-Cerda and Jaron Sanders. Asymptotic convergence rate of dropout on shallow linear neural networks. *arXiv preprint arXiv:2012.01978*, 2020b.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research (JMLR)*, (1), 2014.

Sebastian Stich, Amirkeivan Mohtashami, and Martin Jaggi. Critical parameters for scalable distributed learning with large batches and asynchronous updates. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, 2021.

Sebastian U Stich. Unified optimal analysis of the (stochastic) gradient method. *arXiv preprint arXiv:1907.04232*, 2019.

Sebastian U Stich and Sai Praneeth Karimireddy. The error-feedback framework: Better rates for sgd with delayed gradients and compressed updates. *Journal of Machine Learning Research*, 2020.

Sebastian U Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified sgd with memory. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Xu Sun, Xuancheng Ren, Shuming Ma, and Houfeng Wang. meprop: Sparsified back propagation for accelerated deep learning with reduced overfitting. In *International Conference on Machine Learning*. PMLR, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Hongyi Wang, Saurabh Agarwal, and Dimitris Papailiopoulos. Pufferfish: Communication-efficient models at no extra cost. *Proceedings of Machine Learning and Systems*, 2021.

Wei Wen, Cong Xu, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Coordinating filters for faster deep neural networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. ISBN 9781538610329.

Zhaofeng Wu, Ding Zhao, Qiao Liang, Jiahui Yu, Anmol Gulati, and Ruoming Pang. Dynamic sparsity neural networks for automatic speech recognition. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021.

Yi Xu, Zhuoning Yuan, Sen Yang, Rong Jin, and Tianbao Yang. On the convergence of (stochastic) gradient descent with extrapolation for non-convex minimization. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, 7 2019.

Jiahui Yu and Thomas Huang. Autoslim: Towards one-shot architecture search for channel numbers. *arXiv preprint arXiv:1903.11728*, 2019a.

Jiahui Yu and Thomas S Huang. Universally slimmable networks and improved training techniques. In *Proceedings of the IEEE/CVF international conference on computer vision*, 2019b.

Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. In *International Conference on Learning Representations (ICLR)*, 2019a.

Jiahui Yu, Pengchong Jin, Hanxiao Liu, Gabriel Bender, Pieter-Jan Kindermans, Mingxing Tan, Thomas Huang, Xiaodan Song, Ruoming Pang, and Quoc Le. BigNAS: Scaling up neural architecture search with big single-stage models. In *European Conference on Computer Vision*. Springer, 2020.

Kaicheng Yu, Christian Sciuto, Martin Jaggi, Claudiu Musat, and Mathieu Salzmann. Evaluating the search phase of neural architecture search. In *International Conference on Learning Representations (ICLR)*, 2019b.

Binhang Yuan, Cameron R. Wolfe, Chen Dun, Yuxin Tang, Anastasios Kyrillidis, and Christopher M. Jermaine. Distributed learning of deep neural networks using independent subnet training. *arXiv preprint arXiv:1910.02120*, 2019.

Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.

Supplementary Material: Masked Training of Neural Networks with Partial Gradients

A Proof of Theorem 1

The following proof is inspired by the proof used for BiasedSGD (Ajalloeian and Stich, 2020) and adapted for Algorithm 1.

First, let us recall the following remark about the properties of L-smooth functions.

Remark 2. *L-smooth functions satisfy the following (Nesterov, 2004, Lemma 1.2.3):*

$$f(\mathbf{y}) \leq f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \frac{L}{2}\|\mathbf{x} - \mathbf{y}\|^2 \,, \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d \,. \tag{5}$$

We now prove the following lemma:

Lemma 3. *If the assumptions of Theorem 1 hold:*

$$\mathbb{E}_{\boldsymbol{\xi}(\mathbf{x}_t)} f(\mathbf{x}_{t+1}) \leq f(\mathbf{x}_t) - \frac{\gamma_{base}}{2}\alpha_t^2 \|\mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t)\|^2 + \frac{\gamma_{base}^2 L}{2}\sigma^2 \,, \tag{6}$$

Proof.

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{\xi}(\mathbf{x}_t)} f(\mathbf{x}_{t+1}) &\leq f(\mathbf{x}_t) - \gamma_t \langle \nabla f(\mathbf{x}_t), \mathbb{E}[\mathbf{p}_t \odot \mathbf{g}_t] \rangle + \frac{\gamma_t^2 L}{2}\mathbb{E}\|\mathbf{p}_t \odot \mathbf{g}_t\|^2 \\
&= f(\mathbf{x}_t) - \gamma_t \langle \nabla f(\mathbf{x}_t), \mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t) \rangle + \frac{\gamma_t^2 L}{2}(\mathbb{E}\|\mathbf{p}_t \odot (\mathbf{g}_t - \mathbb{E}\mathbf{g}_t)\|^2 + \|\mathbb{E}\mathbf{g}_t\|^2) \\
&= f(\mathbf{x}_t) - \gamma_t \langle \nabla f(\mathbf{x}_t), \mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t) \rangle + \frac{\gamma_t^2 L}{2}(\mathbb{E}\|\mathbf{p}_t \odot \boldsymbol{\xi}(\tilde{\mathbf{x}}_t)\|^2 + \|\mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t)\|^2) \\
&\leq f(\mathbf{x}_t) - \gamma_t \langle \nabla f(\mathbf{x}_t), \mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t) \rangle + \frac{\gamma_t^2 L}{2}(M+1)\|\mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t)\|^2 + \frac{\gamma_t^2 L}{2}\sigma^2 \\
&\leq f(\mathbf{x}_t) - \gamma_t \alpha_t (1 - \frac{\gamma_t}{2\alpha_t} L(M+1))\|\mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t)\|^2 + \frac{\gamma_t^2 L}{2}\sigma^2 \\
&\leq f(\mathbf{x}_t) - \frac{\gamma_t}{2}\alpha_t \|\mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t)\|^2 + \frac{\gamma_t^2 L}{2}\sigma^2 \\
&\leq f(\mathbf{x}_t) - \frac{\gamma_{base}}{2}\alpha_t^2 \|\mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t)\|^2 + \frac{\gamma_{base}^2 L}{2}\sigma^2 \,.
\end{aligned}$$

Where in the last equation we used the facts that $\alpha_t \leq 1$ and $\gamma_t = \alpha_t \gamma_{base}$. □

We can now prove Theorem 1:

Proof of Theorem 1. Recall that we use $\mathbb{E}_{\boldsymbol{\xi}}$ to refer to the expected over the sequence $\boldsymbol{\xi}(\mathbf{x}_0), \ldots, \boldsymbol{\xi}(\mathbf{x}_{T-1})$. Define $F_t := \mathbb{E}_{\boldsymbol{\xi}} f(\mathbf{x}_t) - f(\mathbf{x}^\star)$. By rearraging the result of Lemma 3 and taking the expectation over all elements in $\boldsymbol{\xi}$ (in addition to $\boldsymbol{\xi}(\mathbf{x}_t)$) we get:

$$\frac{1}{2}\mathbb{E}_{\boldsymbol{\xi}} \alpha_t^2 \|\mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t)\|^2 \leq \frac{F_t - F_{t+1}}{\gamma_{base}} + \frac{\gamma_{base} L}{2}\sigma^2 \,.$$

Averaging over all T, we get a telescoping summation leading to:

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}_{\boldsymbol{\xi}} \alpha_t^2 \|\mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t)\|^2 \leq \frac{2F_0}{T\gamma_{base}} + \gamma_{base} L \sigma^2 \,.$$

We now use $\gamma_{base} = \min\left\{\frac{1}{L(M+1)}, \frac{\varepsilon}{2L\sigma^2}\right\}$ to prove the first part of the theorem. Note that this means $\gamma_{base} L\sigma^2 \leq \frac{\varepsilon}{2}$. Moreover, $T \geq 4\left(\frac{\sigma^2}{\varepsilon^2} + \frac{(M+1)}{\varepsilon}\right) \cdot LF_0$ yields $T\gamma_{base} \geq \frac{4F_0}{\varepsilon}$ which yields $\frac{2F_0}{T\gamma_{base}} \leq \frac{\varepsilon}{2}$. Using these two bounds, proves the first part of the theorem.

For proving the second part note that:

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{\xi}} \|\nabla f(\mathbf{x}_t)\|^2 &= \frac{\|\nabla f(\mathbf{x}_t)\|^2}{\alpha_t^2 \|\mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t)\|^2} \alpha_t^2 \|\mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t)\|^2 \\
&= q_t^2 \alpha_t^2 \|\mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t)\|^2 \\
&\leq q^2 \mathbb{E}_{\boldsymbol{\xi}} \alpha_t^2 \|\mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t)\|^2 .
\end{aligned}$$

where we used $q \geq q_t$ for all $t \in [T]$ and any sequence of noises $\boldsymbol{\xi}$, which directly follows from the definition of q, to get the last inequality. We can now see that

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}_{\boldsymbol{\xi}} \alpha_t^2 \|\mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t)\|^2 \leq \frac{\varepsilon}{q^2} \implies \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}_{\boldsymbol{\xi}} \|\nabla f(\mathbf{x}_t)\|^2 \leq q^2 \cdot \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}_{\boldsymbol{\xi}} \alpha_t^2 \|\mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t)\|^2 \leq \varepsilon .$$

Using $\frac{\varepsilon}{q^2}$ as the target threshold in the bound we obtained for the first part of the theorem gives the desired result. □

B Proof of Lemma 2

Proof. We will first prove the bound for c_{sim}. We apply the inequality $\|a+b\|_2^2 \leq 2\|a\|_2^2 + 2\|b\|_2^2$ and get

$$\begin{aligned}
\|\mathbf{p}_t \odot \nabla f(\mathbf{x}_t)\|^2 &\leq 2\|\mathbf{p}_t \odot (\nabla f(\mathbf{x}_t) - \nabla f(\tilde{\mathbf{x}}_t))\|_2^2 + 2\|\mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t)\|_2^2 \\
&\leq 2\|\nabla f(\mathbf{x}_t) - \nabla f(\tilde{\mathbf{x}}_t)\|_2^2 + 2\|\mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t)\|_2^2 \\
&\overset{(2)}{\leq} 2L^2\|\delta \mathbf{x}_t\|^2 + 2\|\mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t)\|^2 \\
&\overset{(4)}{\leq} (\frac{1}{2} + 2)\|\mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t)\|^2 .
\end{aligned}$$

Hence $c_{\text{sim}} \leq \frac{\sqrt{10}}{2}$. To prove the bound for alignment note that:

$$\begin{aligned}
\langle \mathbf{p}_t \odot \nabla f(\mathbf{x}_t), \mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t)\rangle &= \|\mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t)\|^2 + \langle \mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t), \mathbf{p}_t \odot (\nabla f(\mathbf{x}_t) - \nabla f(\tilde{\mathbf{x}}_t))\rangle \\
&= \|\mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t)\|^2 + \langle \mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t), \nabla f(\mathbf{x}_t) - \nabla f(\tilde{\mathbf{x}}_t)\rangle \\
&\geq \|\mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t)\|^2 - \|\mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t)\|\|\nabla f(\mathbf{x}_t) - \nabla f(\tilde{\mathbf{x}}_t)\| \\
&\overset{(2)}{\geq} \|\mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t)\|^2 - L\|\mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t)\|\|\mathbf{x}_t - \tilde{\mathbf{x}}_t\| \\
&= (1 - L\frac{\|\delta \mathbf{x}_t\|}{\|\mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t)\|})\|\mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t)\|^2 \\
&\overset{(4)}{\geq} \frac{1}{2}\|\mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t)\|^2 \\
&\geq \frac{1}{2} \cdot \frac{2}{\sqrt{10}}\|\mathbf{p}_t \odot \nabla f(\tilde{\mathbf{x}}_t)\|\|\mathbf{p}_t \odot \nabla f(\mathbf{x}_t)\| .
\end{aligned}$$

Where in the last inequality we used the bound for c_{sim} which in turn shows $c_{\text{align}} \leq \sqrt{10}$. □

C Small-Width Training Results

We obtain the small-width variant by reducing the width of the feed-forward modules and the key/query projections in attention layers by a factor r. In particular, we reduce the number of neurons in the hidden layer of the feed-forward modules as well as the key/query embedding dimension of the attention layers by the factor r. In contrast to the low-rank setting where we use an additive combination of the full-rank and low-rank model,

Table 2: BLEU scores of Transformer trained jointly with its **small width sub-network** on IWSLT14 (left) and WMT17 (right) datasets respectively, with and without ATS. The sub-network width reduction factor is denoted by r.

(a) IWSLT14 dataset.

Model	Sub-Network	Full Network
Standard Training	-	32.48 (0.32)
Standard Training ($r = 0.25$)	31.29 (0.27)	-
Alternating Training ($r = 0.25$)	31.59 (0.18)	31.95 (0.35)
Standard Training ($r = 0.03125$)	23.58 (0.58)	-
Alternating Training ($r = 0.03125$)	23.84 (0.25)	30.41(0.40)

(b) WMT17 dataset.

Model	Sub-Network	Full Network
Standard Training	-	26.41(0.14)
Standard Training ($r = 0.03125$)	20.47 (0.34)	-
Alternating Training ($r = 0.03125$)	20.08 (0.40)	25.46(0.17)

Table 3: Final model accuracy and per-epoch FLOPs count of ATS and slimmable training for ResNet-18 over CIFAR10 dataset.

Method	Model	Accuracy	10^{12} FLOPs
ATS	Core	93.94	31.3
	Full	95.05	
Slimmable	Core	93.99	62.7
	Full	94.72	

we use the standard Transformer as the super-network and train it jointly with its sub-network corresponding to the small-width variant.

Table 2a includes the performance of both the full-width and the small-width models trained with this scheme in terms of the BLEU score on IWSLT14 dataset. Furthermore, the performances of the same models with standard training are also reported. It can be clearly seen that training with this scheme yields better small-width models. Additionally, while the full-width models have a lower performance when trained with ATS than when trained standardly, their performance are still comparable. We also perform the same experiment on WMT17 dataset and report the results in Table 2b and similarly observe that using ATS, results in a small network matching the performance of standard training and a full model with comparable performance while the cost of training is less than training these two models separately.

We would like to point out that in comparison to low-rank models, small-width models have a lower performance on the WMT17 dataset for small values of r while the number of parameters remain comparable. In particular, standard training of a low-rank Transformer with $r = 0.015625$ (which has the same number of parameter as low-width Transformer with $r = 0.03125$) yields 22.10 BLEU score (vs 20.47 BLEU score). This encourages using low-rank models instead of low-width models.

For completeness, given the similarity of our method and slimmable training we compare the two by performing the small-width experiments with $r = 0.5$ on a ResNet-18 over CIFAR10 datasets. The results are reported in Table 3. We additionally reported an estimate of the number of FLOPs per epoch performed by each method. It can be seen that while the performance of both methods are similar, ATS is much more efficient.

D Hyperparameters for Experiments

We train each experiment with IWSLT14 dataset on a single GTX Titan X GPU and use a Tesla V100 GPU for training WMT17 experiments. For the implementation of our experiments we extend the implementation of transformer in FairSeq (Ott et al., 2019) and use its CLI (`fairseq-train`) for training. The hyperparameters based on the dataset are reported in Table 4. For training the sub-network alone we additionally set the parameter ”max_level” to 1. For training with alternating scheme we set the parameter “alternate” to True. Training the normal Transformer (with or without ATS) for one epochs take approximately 7 minutes on IWSLT14 dataset and 100 minutes on WMT17 dataset.

For CIFAR10, we train using SGD with momentum 0.9 and weight decay 5×10^{-4}. Separate batch normalization layers were used for the core and the super network. For measuring alignment with and without dropout (Figure 2), batch normalization was disabled.

Table 4: Hyper-parameters used for training on different datasets

Parameter	WMT17	IWSLT14
adam_betas	(0.9, 0.98)	
adam_eps	1e-09	
alternate_lr_coef	1.0	
criterion	label_smoothed_cross_entropy	
decoder_attention_heads	8	4
decoder_embed_dim	512	
decoder_ffn_embed_dim	2048	1024
decoder_layers	6	
dropout	0.1	0.3
encoder_attention_heads	8	4
encoder_embed_dim	512	
encoder_ffn_embed_dim	2048	1024
encoder_layers	6	
label_smoothing	0.1	
lr	0.0007	0.0005
lr_scheduler	inverse_sqrt	
max_epoch	22	30
max_tokens	18000	12000
optimizer	adam_for_partitioned_model	
partitioned_linear_module_name	PartitionedLinear or PartitionedRankLinear	
share_decoder_input_output_embed	False	True
small_ratio	0.03125	
task	translation_with_partitioned_model	
warmup_init_lr	1e-07	-1
warmup_updates	4000	
weight_decay	0.0	0.0001