ACCORDION: ADAPTIVE GRADIENT COMPRESSION VIA CRITICAL LEARNING REGIME IDENTIFICATION

Saurabh Agarwal [1] **Hongyi Wang** [1] **Kangwook Lee** [2] **Shivaram Venkataraman** [1] **Dimitris Papailiopoulos** [2]

ABSTRACT

Distributed model training suffers from communication bottlenecks due to frequent model updates transmitted across compute nodes. To alleviate these bottlenecks, practitioners use gradient compression techniques like sparsification, quantization, low-rank updates etc. The techniques usually require choosing a static compression ratio, often requiring users to balance the trade-off between model accuracy and per-iteration speedup. In this work, we show that such performance degradation due to choosing a high compression ratio is not fundamental and that an adaptive compression strategy can reduce communication while maintaining final test accuracy. Inspired by recent findings on critical learning regimes, in which small gradient errors can have irrecoverable impact on model performance, we propose ACCORDION a simple yet effective adaptive compression algorithm. While ACCORDION maintains a high enough compression rate on average, it avoids detrimental impact by not compressing gradients too much whenever in critical learning regimes, detected by a simple gradient-norm based criterion. Our extensive experimental study over a number of machine learning tasks in distributed environments indicates that ACCORDION, maintains similar model accuracy to uncompressed training, yet achieves up to 5.5× better compression and up to 4.1× end-to-end speedup over static approaches. We show that ACCORDION also works for adjusting the batch size, another popular strategy for alleviating communication bottlenecks. Our code is available at https://github.com/uw-mad-dash/Accordion.

1 INTRODUCTION

Billion-parameter-scale neural networks and the rapid increase in compute requirements for training them has made distributed gradient-based methods a necessity. Synchronous, data-parallel training is the most widely adopted approach in this context, and requires combining the per-node gradient updates at every iteration (Dean et al., 2012; Iandola et al., 2016; Goyal et al., 2017; Shallue et al., 2018). Communicating gradients frequently at such a large parameter scale leads to sub-optimal scalability in distributed implementations (Dean et al., 2012; Seide et al., 2014; Alistarh et al., 2017; Mattson et al., 2020; Luo et al., 2020).

To alleviate gradient communication bottlenecks, there are two main approaches proposed by prior work: (1) increasing the batch size (Smith et al., 2017; Goyal et al., 2017; Yao et al., 2018a), such that gradients are computed on a large batch by each worker thus reducing the frequency of per-epoch communication and (2) by performing lossy gradient compression (Alistarh et al., 2017; Vogels et al., 2019), to reduce the size of the data communicated. Both of these methods involve navigating a trade-off between performance and accuracy.

It is a widely observed phenomenon that using large batch size can lead to degradation in final accuracy (Yao et al., 2018b; Golmant et al., 2018; Shallue et al., 2018). In response, a number of recent works have proposed techniques to mitigate this accuracy loss, by using learning rate warmup (Goyal et al., 2017) or second order information (Yao et al., 2018a;b) or layer-wise LR tuning (You et al., 2017). Some deployment challenges with these methods include the need for significant amount of hyper-parameter tuning or running more epochs to converge to a good accuracy (Golmant et al., 2018; Shallue et al., 2018).

On the other hand, when using gradient compression techniques including low-precision training (Seide et al., 2014; Alistarh et al., 2017; Wen et al., 2017; Bernstein et al., 2018; Acharya et al., 2019), TOPK methods that exchange only the largest gradient coordinates (Aji & Heafield, 2017; Lin et al., 2017; Shi et al., 2019b;a) or methods that use low-rank based updates (Wang et al., 2018; Vogels et al., 2019), users need to specify an additional hyper-parameter that determines the degree of compression or sparsification before training begins. Choosing compression ratios presents a seemingly

[1]Department of Computer Science, University of Wisconsin-Madison [2]Department of Electrical and Computer Engineering, University of Wisconsin-Madison. Correspondence to: Saurabh Agarwal <agarwal@cs.wisc.edu>.

Proceedings of the 4^{th} MLSys Conference, San Jose, CA, USA, 2021.

inherent trade-off between final model accuracy and the per-iteration communication overhead. For instance, training ResNet-18 on CIFAR-10 using TOP$K$with $K = 10\%$ sparsification (i.e., where only 10% of the top entries per gradient are communicated) takes around 3.6× less wall-clock time than training using $K = 99\%$, but causes around 1.5% degradation in final accuracy.

This raises a fundamental question related to gradient compression: *Is this observed trade-off between communication and accuracy fundamental?* In this work, we first show that such a trade-off is *not* fundamental but a mere artifact of using a *fixed communication scheme* throughout training. In other words, if a gradient communication schedule is chosen adaptively, then both the final model performance and communication efficiency can be improved, when compared against any fixed gradient communication schedule throughout training. Figure 1 shows an experiment where we train ResNet-18 on CIFAR-100, with different compression schemes. We see that there exists an adaptive compression scheme that significantly reduces communication while maintaining final test accuracy.

We attribute the power of adaptive schemes to the existence of *critical regimes* in training. We build upon recent work by Achille et al. (2019); Jastrzębski et al. (2019), who show that even adding small noise to the data during critical regimes can result in poor generalization. We extend this notion to gradient communication and show that more communication is only required during these critical regimes. Thus, we can design an adaptive scheme that tries to identify whether the current training regime is critical or not and then accordingly adjust the gradient compression rate.

Based on these findings, we propose ACCORDION, a simple but powerful gradient communication scheduling algorithm that is generic across models while imposing low computational overheads. ACCORDION inspects the change in the gradient norms to detect critical regimes and adjusts the communication schedule dynamically leading to performance improvements without sacrificing accuracy. Finally, we also show that ACCORDION works for both adjusting the gradient compression rate or the batch size without additional parameter tuning, hinting at a possible equivalence between the two.

Our experiments show that ACCORDION can achieve improved communication efficiency while maintaining high generalization performance on computer vision and language modelling tasks. In particular, using the SOTA sparsification (TOPK) and low-rank approximation methods (POWERSGD), we show reduction in communication up to 3.7× and speedup up to 3.6× compared to using low compression throughout, while achieving similar accuracy with the full rank/uncompressed SGD. When used for batch size tuning, we show that ACCORDION, *without any hyper-parameter tuning* is able to utilize extremely large batch sizes leading to reduction in communication of up to $5.5\times$ and speedup of $4.4\times$.

Contributions:

- We show that gradient compression schemes need to be aware of critical regimes in training to avoid the trade-off between accuracy and performance.
- We design ACCORDION an adaptive scheme that can switch the amount of communication used by inspecting the gradient norms thereby yielding performance improvements without sacrificing accuracy.
- We provide extensive empirical evaluation of ACCORDION on several different neural networks using POWERSGD and TOPK two state of the art gradient compressors with different data sets (CIFAR-10, CIFAR-100, WikiText-2) showing ACCORDION reduces communication by $3.7\times$ without loss of accuracy or change in hyper-parameters.
- We also show that ACCORDION can enable large batch training by switching between large batch size and small batch size. ACCORDION *without any hyper-parameter tuning*, is able to reducing the time to accuracy by performing up to 5x fewer updates compared to using a small batch size.

2 RELATED WORK

Lossy gradient compression Inspired by the fact that SGD can make good progress even with approximate gradients,various Gradient Compression methods have been proposed in recent literature. They can be broadly grouped into quantization, sparsification and low rank aproximations. For quantization, (Seide et al., 2014; Bernstein et al., 2018) replace each weight with just the sign values. While (Aji & Heafield, 2017; Lin et al., 2017; Shi et al., 2019a;b) use the largest few co-ordinates to create a sparse gradient. Wangni et al. (2018) randomly drop coordinates in the gradient update to create sparse gradient updates. For quantization (Alistarh et al., 2017; Wen et al., 2017) quantize each gradient coordinate. In (Wang et al., 2018; Vogels et al., 2019) authors show that extremely low rank updates can achieve good compression without loss in accuracy. Yu et al. (2018) utilize correlation between gradients for linear compression. In ACCORDION our goal is to operate over an existing gradient compression method and provide reduction in communication without hurting generalization performance.

Local SGD Unlike the Lossy Gradient Compression methods which reduce the size of gradient updates, Local SGD methods reduce the frequency of updates, by averaging

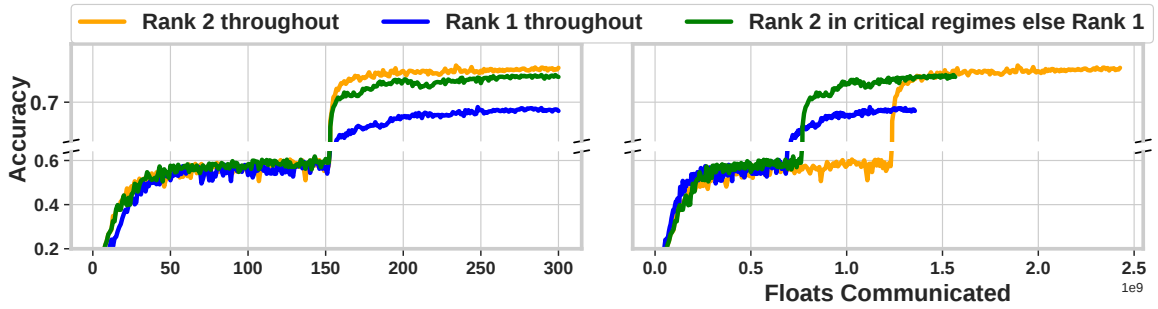


Figure 1. **Effect of gradient compression in Critical Regimes when Training ResNet-18 on CIFAR-100:** We conduct the experiments on a 4 Node cluster using POWERSGD RANK 2 (low compression) and RANK 1 (high compression) (left) Accuracy vs Epochs, there exist a compression pattern which provides similar training accuracy as using low compression(RANK 2) throughout. (right) Accuracy vs Floats Communicated, this compression pattern communicates significantly less than using low compression throughout (RANK 2).

weights every τ steps. (Stich, 2019; Lin et al., 2020; Wang et al., 2020; Dutta et al., 2020) show that local SGD offers competitive performance on a variety of tasks. In this work we explicitly focus on reducing communication further using gradient compression or by varying batch size and plan to investigate if our insights can also apply for Local SGD in the future.

Adaptive communication Wang et al. (Wang & Joshi, 2018) proposed an adaptive scheme to choose number of local steps τ adaptively, this method is applicable only on local SGD. Chen et al. (2018) proposed an auto-tuned compression method, but unlike ACCORDION it is a gradient compression method in itself and can't be applied with other methods. Recently Guo et al. (2020) proposed an adaptive scheme to choose quantization factor for each coordinate in gradient vector, however in Figure 7 we observe that their method leads to some accuracy loss when used for POWERSGD.

Critical regimes In (Achille et al., 2019; Frankle et al., 2020) authors highlighted the presence of critical regimes in neural network training. Various other works have highlighted the importance of early phases of training including, Gur-Ari et al. (2018) who show that gradient descent moves into a small sub-space after a few iterations, (Keskar et al., 2016; Jastrzębski et al., 2019; Jastrzebski et al., 2020) show that SGD is initially driven to difficult to optimize regimes. We leverage these insights to reduce communication when using gradient compression algorithms.

Batchsize scheduling (Smith et al., 2017; Goyal et al., 2017; Hoffer et al., 2017; You et al., 2017; Yin et al., 2017), show that large-batch SGD will hurt the eventual generalization performance. More surprisingly, You et al. (2017) show that the use of large-batch SGD does not hurt the performance if used in later phases of the training. There are several works (Goyal et al., 2017; You et al., 2017; Smith et al., 2017; Devarakonda et al., 2017; Yao et al., 2018b) which use adaptive batch size scaling. Either these methods require significant hyper-parameter tuning (Smith et al., 2017) or require second order information (Yao et al., 2018a;b). (Yao et al., 2018a) does provide an adaptive method for batch size scheduling, but their method requires calculation of second order statistics which can often require more time than the gradient computation itself. In Sec. 5 we show that without any hyper-parameter tuning ACCORDION can enable large batch training and converge to same test accuracy with the same epoch budget as small batch training. In Sec. 4.3, we show some non-trivial connection between ACCORDION and well-known suggestions on batch-size scheduling.

3 DISTRIBUTED SGD

In this section, we formally describe the distributed SGD setting Consider the standard synchronous distributed SGD setting with N distributed workers (Sergeev & Del Balso, 2018). For simplicity, we assume that each worker stores n data points, giving us a total of $N \times n$ data points, say $\{(x_i, y_i)\}_{i=1}^{Nn}$.

The goal is to find a model parameter w that minimizes $f(w) = \frac{1}{Nn}\sum_{i=1}^{Nn} \ell(w; x_i, y_i)$ where (x_i, y_i) is the i-th example. In particular, we minimize $f(w)$ using distributed SGD that operates as follows: $w_{k+1} = w_k - \gamma_k \frac{1}{N}\sum_{i=1}^{N} \widehat{g}_i(w_k)$ for $k \in \{0, 1, 2, \ldots\}$, where w_0 is the initial model, γ_k is the step size, and $\widehat{g}_i(w)$ is a gradient computed at worker i for a minibatch (of size B, with $B < n$).

Distributed SGD with adaptive gradient compression Vanilla distributed SGD incurs a huge communication cost per iteration that is proportional to the number of workers N and the size of the gradient. To reduce this communication overhead, we consider a gradient compression strategy, say $C(\cdot, \ell)$, where ℓ is the parameter that determines the compression level used. With such a gradient compression strategy, the update equation be-

comes $w_{k+1} = w_k - \gamma_k \frac{1}{N} \sum_{i=1}^{N} C(\widehat{g}_i(w_k), \ell_k)$ for $k \in \{0, 1, 2, \ldots\}$, where communicating $C(\widehat{g}_i(w_k), \ell_k)$ requires much fewer bits than communicating the original gradients.

Distributed SGD with adaptive batch size The number of communication rounds in a given epoch also depend on the batch size. For example a batch size $B_{high} > B_{low}$ will communicate $\left\lfloor \frac{B_{high}}{B_{low}} \right\rfloor$ times less than using batch size B_{low} in a given epoch. Although the update equation remains the same $w_{k+1} = w_k - \gamma_k \frac{1}{N} \sum_{i=1}^{N} \widehat{g}_i(w_k)$ for $k \in \{0, 1, 2, \ldots\}$, the number of steps k, taken by a model decreases by $\left\lfloor \frac{B_{high}}{B_{low}} \right\rfloor$ times for a fixed number of epochs.

Goals Our goal is to design an algorithm that automatically adapts the compression rate $\{\ell_k\}$ or batch size B_k while training. Although the interplay between batch size and compression ratio is interesting, we don't explore these together, i.e. we don't vary batch size when training with gradient compression. Here, we consider a centralized algorithm, *i.e.*, one of the participating node decides ℓ_{k+1} or B_{k+1} based on all the information available up till step k. This communication rate is then shared with all the N workers so that they can adapt either their compression ratio or batch size.

4 ACCORDION

In this section, we first explain why adaptive gradient communication can help maintain high generalization performance while minimizing the communication cost. We study this first with gradient compression techniques and then based on these insights we propose ACCORDION a gradient communication scheduling algorithm. Finally, we show that there is a connection between batch size and gradient compression, and thus ACCORDION can also be used to enable large batch training without accuracy loss.

4.1 Adaptive communication using critical regimes

Recent work by Achille et al. (2019) has identified *critical regimes* or phases of training that are important for training a high quality model. In particular, Achille et al. (2019) show that the early phase of training is critical. They setup an experiment where the first few epochs have corrupted training data and then continue training the DNN with clean training data for the rest of the epochs. Surprisingly, the DNN trained this way showed a significantly impaired generalization performance no matter how long it was trained with the clean data after the critical regime.

We extend these ideas to aid in the design of an adaptive communication schedule and first study this using POWERSGD as the gradient compression scheme. We begin by observing how the gradient norm for each layer behaves while training. When training ResNet-18 on CIFAR-100, in Figure 2a we see two regions where gradient norm decreases rapidly; during the first 20 epochs and the 10 epochs right after the 150-th epoch, i.e., the point at which learning rate decay occurs. We experimentally verify that these two regions are critical by considering the following compression schedule $\ell =$ LOW for the first 20 epochs and for 10 epochs after the 150 epoch, and $\ell =$ HIGH elsewhere. Under this scheme the gradients will not be over-compressed in the critical regimes, but at the same time the overall communication will be close to high compression. Figure 2b shows the experimental results with ResNet-18 on CIFAR-100 for the above scheme. It can be observed that just using low compression (rank 2) in these critical regimes and high compression (rank 1) elsewhere is sufficient to get the same accuracy as using low compression throughout while reducing communication significantly.

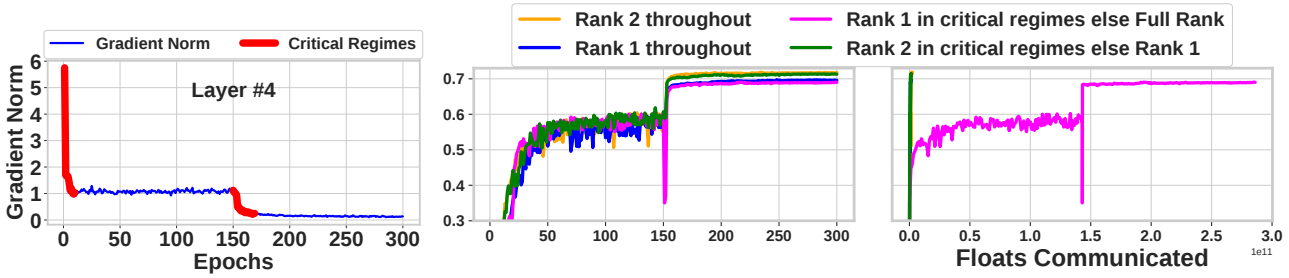
Interestingly we also observe in Figure 2b that any loss in accuracy by using high compression in critical regimes is not recoverable by using low compression elsewhere. For instance, consider the following compression schedule: $\ell =$ HIGH COMPRESSION RATE for first 20 epochs and for 10 epochs after the 150 epoch, and $\ell =$ NO COMPRESSION elsewhere. Under this schedule, gradients will be over-compressed in the critical regimes, but will be *uncompressed* elsewhere. We see that for ResNet-18 on CIFAR-100 even with significantly higher communication one can not overcome the damage done to training by over compressing in critical regimes. We hypothesize that in critical regimes, SGD is navigating to the steeper parts of the loss surface and if we use over-compressed gradients in these regimes, then the training algorithm might take a different trajectory than what SGD would have taken originally. This might cause training to reach a sub-optimal minima leading to degradation in final test accuracy.

Detecting Critical Regimes: Prior work for detecting critical regimes (Jastrzębski et al., 2019) used the change in eigenvalues of the Hessian as an indicator. We next compare the critical regimes identified by the gradient norm approach described above with the approach used in Jastrzębski et al. (2019). In Figure 3, we show that these two approaches yield similar results for ResNet-18 on CIFAR-10, with the latter having an advantage of being orders of magnitude faster to compute.

Thus, we can see that finding an effective communication schedule is akin to finding critical regimes in neural network training and these *critical regimes* can be identified by measuring the change in gradient norm.

4.2 ACCORDION's Design

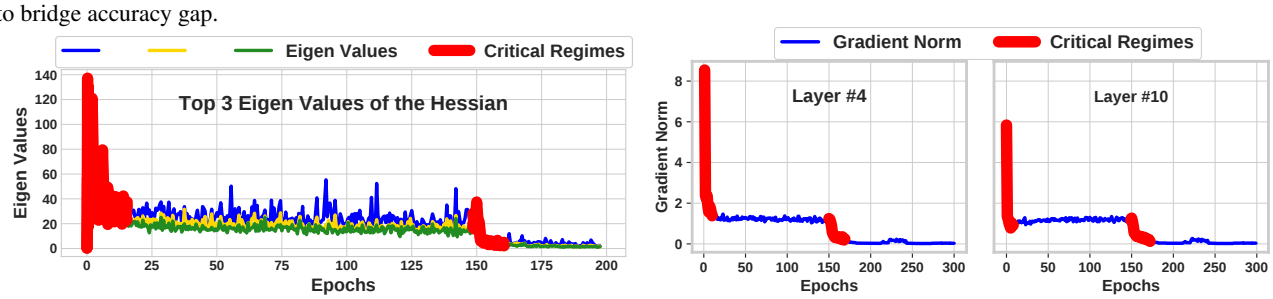
We now provide a description of ACCORDION, our proposed algorithm that automatically switches between lower and



(a) Critical Regimes

(b) Accuracy vs Epochs and Floats Communicated

Figure 2. **Effect of gradient compression in Critical Regimes when Training ResNet-18 on CIFAR-100:** (a) Critical regimes in CIFAR-100, ResNet-18 (b, Left) Accuracy vs Epochs. Show the significance of critical regimes in training, using low compression(Rank 2) in critical regimes is enough to get similar accuracy as using low compression throughout . (b, Right) Accuracy vs Floats Communicated, Even when we use uncompressed (Full Rank) gradients everywhere but use high compression (Rank 1) in critical regimes it is not possible to bridge accuracy gap.



(a) Critical Regimes based on Hessian

(b) Critical Regimes based of Gradient Norm

Figure 3. **Comparison of Critical Regimes found using Analysis of eigenvalues of Hessian vs Using the Norm of the Gradient:** The experiment is performed on ResNet-18, for CIFAR-10. We show that Critical Regimes detected by rapid decay in top eigenvalues of Hessian can also be detected using decay in gradient norm.

higher communication levels by detecting critical regimes. ACCORDION's first goal is to identify critical regimes efficiently. Our experiments, as discussed previously (Figure 3), reveal that critical regimes can be identified by detecting the rate of change in gradient norms without using the computationally expensive technique of (Keskar et al., 2016; Jastrzębski et al., 2019; Jastrzebski et al., 2020), where eigenvalues of the Hessian are used to detect critical regimes. This leads us to propose the following simple way to detect critical regimes:

$$\frac{\big|\, \|\Delta_{\text{old}}\| - \|\Delta_{\text{curr}}\| \,\big|}{\|\Delta_{\text{old}}\|} \geq \eta,$$

where Δ_{curr} and Δ_{prev}, denotes the accumulated gradient in the current epoch and some previous epoch respectively, and η is the threshold used to declare critical regimes. We set $\eta = 0.5$ in all of our experiments.

We depict ACCORDION for gradient compression in Algorithm 1. For simplicity and usability, ACCORDION only switches between two levels of compression levels: ℓ_{low} and ℓ_{high}. Once ACCORDION detects critical regimes, it sets the compression level as ℓ_{low} to avoid an undesirable drop in accuracy. Based on our observation, critical regimes also almost always occur after learning rate decay, therefore we let ACCORDION declare critical regime after every learning rate decay. If ACCORDION detects that the critical phase ends, it changes the compression level to ℓ_{high} to save communication cost. For batch size we use the same algorithm, except instead of switching between ℓ_{low} and ℓ_{high} we switch between B_{low} and B_{high}.

We remark that ACCORDION operates at the granularity of the gradient compressor being used. For instance, POWERSGD approximates the gradients of each layer independently, so ACCORDION will also operate at each layer independently and provide a suitable compression ratio for each layer in an adaptive manner during training. While batch size scheduling operates at the whole model so ACCORDION looks at the gradient of whole model and chooses a suitable batch size.

Computational and memory overhead: ACCORDION accumulates gradients of each layer during the backward pass. After each epoch, norms are calculated, creating $\|\nabla_{curr}\|$. Once the compression ratio is chosen $\|\nabla_{curr}\|$ becomes $\|\nabla_{old}\|$. Thus requiring only size of the model(47 MB in ResNet-18) and a few float values worth of storage. Also ACCORDION only uses the ratio between previous and cur-

Algorithm 1 ACCORDION for Gradient Compression

HyperParameters:compression levels $\{\ell_{\text{low}}, \ell_{\text{high}}\}$ and detection threshold η
Input: accumulated gradients in the current epoch (Δ_{curr}) and in the previous epoch (Δ_{prev})
Input: learning rate of the current epoch (γ_{curr}) and of the next epoch (γ_{next})
Output: compression ratio to use ℓ
if $|\|\Delta_{\text{prev}}\| - \|\Delta_{\text{curr}}\|| / \|\Delta_{\text{prev}}\| \geq \eta$ or $\gamma_{next} < \gamma_{curr}$ **then**
  return ℓ_{low}
else
  return ℓ_{high}
end if

rent gradient norms to detect critical regimes. This allows ACCORDION to be easily integrated in a training job where gradients are already calculated, thus making the computational overhead negligible.

4.3 Relationship between gradient compression and adaptive batch-size

We first evaluate the effect of batch size on neural network training through the lens of *critical regimes*, which suggests using small batch sizes in critical regimes and large batch size outside critical regimes should not hurt test accuracy. We empirically show in Figure 4b that this is indeed true.

Next, the connection between compression and batch size tuning can be made more formal under the following assumption: "each stochastic gradient is the sum of a sparse mean and a dense noise", i.e.,

$$\nabla_w \ell(w; x_i, y_i) = \underbrace{\mathbb{E}_j \nabla_w \ell(w; x_j, y_j)}_{\text{sparse, large magnitudes}} + \underbrace{(\nabla_w \ell(w; x_i, y_i) - \mathbb{E}_j \nabla_w \ell(w; x_j, y_j))}_{\text{dense, small magnitudes}} \tag{1}$$

Under this assumption, we can see that "large batch gradient $\approx$ highly compressed gradient", as a large batch gradient will be close to $\mathbb{E}_j \nabla_w \ell(w; x_j, y_j)$ by the law of large numbers, a highly compressed gradient will also pick up the same sparse components. Similarly, a small batch gradient is equivalent to weakly compressed gradient. We will like to point out that this assumption is not general and is not applicable on all data or models. It will only hold for models trained with sparsity inducing norms.

We also conduct a simple experiment to support our intuition. We collect all stochastic gradients in an epoch and compute the overlap in coordinates of *Top10%* entries to find how much their supports overlap. Figure 4a shows that $> 90\%$ of the top-K entries are common between a pair of stochastic gradients, thereby justifying the above gradient modeling.

Thus, our findings along with prior work in literature can be summarized as high gradient compression, noisy training data, or large batch size in the critical regimes of training hurts generalization. We study this connection further in Appendix B. This connection also suggests that ACCORDION can also be used to schedule batch size and in Section 5 we evaluate this.

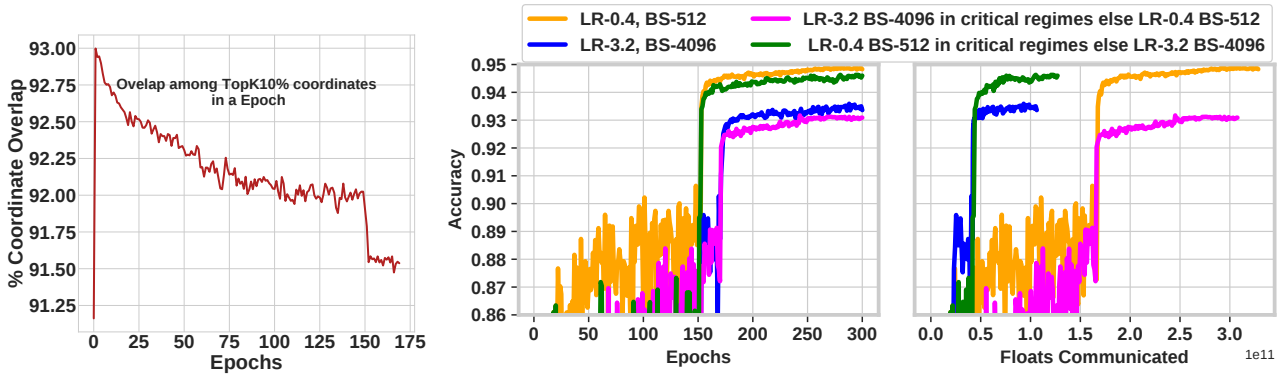
5 EXPERIMENTAL EVALUATION

We experimentally verify the performance of ACCORDION when paired with two SOTA gradient compressors, *i.e.*, (i) POWERSGD (Vogels et al., 2019), which performs low-rank gradient factorization via a computationally efficient approach, and (ii) TOPK sparsification (Aji & Heafield, 2017), which sparsifies the gradients by choosing the K entries with largest absolute values. Further we also use ACCORDION to schedule batch size switching between batch size 512 and 4096 for CIFAR-100and CIFAR-10.

5.1 Experimental setup

Implementation We implement ACCORDION in PyTorch (Paszke et al., 2019). All experiments were conducted on a cluster that consists of 4 `p3.2xlarge` instances on Amazon EC2. Our implementation used NCCL an optimized communication library for use with NVIDIA GPUs. For POWERSGD and Batch Size experiments we used the all-reduce collective in NCCL and for TOPK we used the all-gather collective.

Hyperparameters We fix η to be 0.5 and run ACCORDION every 10 epochs i.e. ACCORDION detects critical regimes by calculating rate of change between gradients accumulated in current epoch and the gradients accumulated 10 epochs back. We empirically observe that these choices of hyper-parameters lead to good results and have not tuned them. One of our primary goal was to design ACCORDION such that it should not require signifcant amount of hyper-parameter tuning. Therefore for all of our experiments we didn't perform any hyper-parameter tuning and used the same hyper-parameters as suggested by authors of previous compression methods, e.g. For POWERSGD we used the same setting as suggested by Vogels et al. (2019). For large batch size experiments we use the same hyper-parameters as used for regular training. For all our experiments on batch size we performed LR Warmup of 5 epochs as suggested by Goyal et al. (2017), i.e. for batch size 512 we linearly increase the learning rate from 0.1 to 0.4 in five epochs where 0.1 is learning rate for batch size 128. Due to relationship shown between batch Size and learning rate by Smith et al. (2017); Devarakonda et al. (2017) when ACCORDION shifts to large batch it also correspondingly increases the learning



(a) Overlap in coordinates

(b) Effect of Different Batch sizes in critical regimes

Figure 4. **Effect of batch size (ResNet-18 on CIFAR-10):** (a) We show that there is significant overlap among the TOP10% coordinates. (b, left) Shows that using small batches only in critical regimes is enough to get performance similar to using small batches everywhere. We scale learning rate linearly with batch size as in (Goyal et al., 2017), at steps 150 and 250 we decay the learning rate by 10 and 100 respectively. (b, right) accuracy vs communication.

in the same ratio, i.e. when switching between Batch Size 512 to Batch size 4096, ACCORDION also scales the learning rate by $8\times$. Detailed experimental setup can be found in the Appendix.

Dataset and Models For image classification tasks we evaluated ACCORDION on CIFAR-10 and CIFAR-100 (Krizhevsky et al., 2009). CIFAR-10 consists of 50,000 train images and 10,000 test images for 10 classes. CIFAR-100 has similar number of samples but for 100 classes. For language modeling we used WIKITEXT-2 which has around 2 million train tokens and around 245k testing tokens. We used standard prepossessing steps, details of which can be found in the Appendix. To show the wide applicability of ACCORDION we consider a number of model architectures. For CNNs, we study networks both with and without skip connections. VGG-19 (Simonyan & Zisserman, 2014) and GoogleNet (Szegedy et al., 2015) are two networks without skip connections. While ResNet-18 (He et al., 2016), Densenet (Huang et al., 2017), and Squeeze-and -Excitation (Hu et al., 2018) are networks with skip connections. For language tasks we used a two layer LSTM. More details about the models can be found in Appendix.

Metrics We evaluate ACCORDION against high communication training on three different metrics: (i) accuracy; (ii) communication savings; (iii) total wall clock time saved. We train each method for the same number of epochs with the hyper-parameters suggested in prior literature (Goyal et al., 2017; Aji & Heafield, 2017; Smith et al., 2017; Vogels et al., 2019). We report the mean test accuracy reached after three independent trials. Our error bars report 95% confidence interval.

5.2 Results

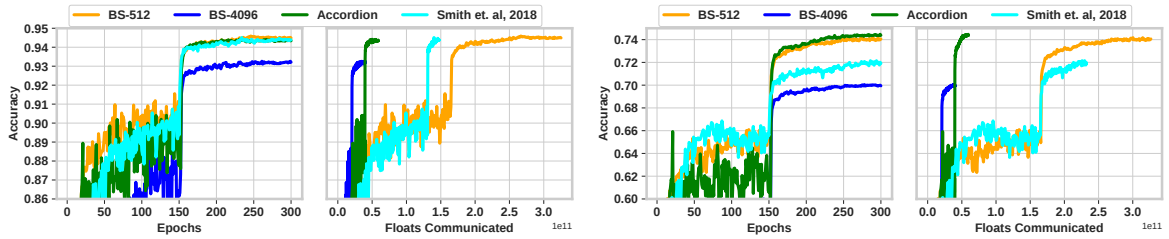
ACCORDION's performance is summarized in Tables 1 to 6. For each model we state the accuracy achieved when using low communication, high communication, and compare it to using ACCORDION to automatically switch between low and high communication. Detailed convergence curves with error bars can be found in Appendix.

5.3 ACCORDION with POWERSGD

POWERSGD (Vogels et al., 2019) shows that using extremely low rank updates (Rank-2 or Rank-4) with error-feedback (Stich & Karimireddy, 2019) can lead to the the same accuracy as syncSGD. In Table 1 and 2 we show that ACCORDION by performing adaptive switching between Rank-1 and Rank-2,4 reaches similar accuracy but with significantly less communication. For e.g. in Table 2 with ResNet-18 on CIFAR-100 using $\ell_{\text{low}} =$ Rank 2 leads to accuracy of 72.4% while $\ell_{\text{high}} =$ Rank 1 achieves 71.3%. ACCORDION switching between RANK 2 and RANK 1 achieves an accuracy of 72.3%. Figure 6 shows the result for VGG-19bn trained with CIFAR-10, in this case ACCORDION almost bridges accuracy gap of 25% while saving almost $2.3\times$ in communication.

5.4 ACCORDION with TOPK

In Table 3 and 4 we show ACCORDION reaches same accuracy as using TOPK99% but with significantly less communication. Our implementation of TopK follows from Aji & Heafield (2017). We were unable to find details on parameters which work reasonably well for all networks. Thus, from a users perspective who wants performance extremely close to syncSGD we choose TOPK99% as low compression. For high compression we choose a value



(a) ResNet-18, trained on CIFAR-10

(b) ResNet-18 trained on CIFAR-100

Figure 5. **ACCORDION compared with scheme proposed by Smith et al. (2017):** We observe ACCORDION communicates around $3\times$ less than the scheme proposed by Smith et al. (2017). Moreover ACCORDION for both CIFAR-100 and CIFAR-10 maintains same accuracy as using small batch size (high communication).

Table 1. ACCORDION with POWERSGD on CIFAR-10

Network	Rank	Accuracy	Data Sent (Million Floats)		Time (Seconds)	
Resnet-18	Rank 2	**94.5%**	2418.4	(1×)	3509	(1×)
	Rank 1	94.1%	1350.4	(1.7×)	3386	(1.03×)
	ACCORDION	**94.5%**	1571.8	(**1.5**×)	3398	(**1.03**×)
VGG-19bn	Rank 4	**93.4%**	6752.0	(1×)	3613	(1×)
	Rank 1	68.6%	2074.9	(3.25×)	3158	(1.14×)
	ACCORDION	**92.9%**	2945.1	(**2.3**×)	3220	(**1.12**×)
Senet	Rank 4	**94.5%**	4361.3	(1×)	4689	(1×)
	Rank 1	94.2%	1392.6	(3.1×)	4134	(1.13×)
	ACCORDION	**94.5%**	2264.4	(**1.9**×)	4298	(**1.09**×)

Table 2. ACCORDION with POWERSGD on CIFAR-100

Network	Rank	Accuracy	Data Sent (Million Floats)		Time (Seconds)	
Resnet-18	Rank 2	**71.7%**	2426.3	(1×)	3521	(1×)
	Rank 1	70.0%	1355.7	(1.8×)	3388	(1.04×)
	ACCORDION	**71.8%**	1566.3	(**1.6**×)	3419	(**1.03**×)
DenseNet	Rank 2	**72.0%**	3387.4	(1×)	13613	(1×)
	Rank 1	71.6%	2155.6	(1.6×)	12977	(1.04×)
	ACCORDION	**72.5%**	2284.9	(**1.5**×)	13173	(**1.03**×)
Senet	Rank 2	**72.5%**	2878.1	(1×)	5217	(1×)
	Rank 1	71.5%	1683.1	(1.7×)	4994	(1.04×)
	ACCORDION	**72.4%**	2175.6	(**1.3**×)	5074	(**1.03**×)

which provides significantly more compression. For ResNet-18 trained on CIFAR-10 we observe that high compression, TOPK25% reaches accuracy of 71.3% while low compression TOPK99% reaches accuracy of 72.4%, ACCORDION on the other hand reaches accuracy of 72.3% while reducing the communication by $2.8\times$.

5.5 ACCORDION with Large Batch size

In Table 5 and 6 we show that ACCORDION is able to reach the same accuracy as small batch training without any hyper-parameter tuning. We modified no other parameter except scaling learning rate when switching Batch Size as described in Section 5.1. ACCORDION by switching between batch size of 512 and 4096 is able to save around $5.5\times$ in communications and up to $4.1\times$ reduction in wall clock training time.

5.6 Comparison with Prior Work

We compare ACCORDION with prior work in adaptive gradient compression and adaptive batch size tuning. For adaptive gradient compression we consider recent work by Guo et al. (2020) that uses the mean to standard deviation ratio (MSDR) of the gradients. If they observe that MSDR has reduced by a certain amount(a hyper-parameter), they correspondingly reduce the compression ratio by half (i.e., switch to a more accurate gradient). We use this approach with POWERSGD and our experiments in Figure 7 suggest that their switching scheme ends up requiring more communication and also leads to some loss in accuracy.

For batch size we compare to (Smith et al., 2017) in Figure 5. We used the exact same setup as suggested by (Smith et al., 2017) and we use the *Increased Initial Learning Rate* setting as shown in Figure 5 of their paper. We observe that ACCORDION reduces communication by $5.4\times$. On the other hand Smith et al. (2017) only reduce communication by $2.2\times$. For CIFAR-100 as shown in Figure 5b we observe that the approach presented by Smith et al. (2017) doesn't yield the same accuracy as small batch training.

Prior work (Alistarh et al., 2017) has shown that in theory highly compressed gradients can reach the same accuracy as low compressed gradients when trained long enough. However, it only makes sense to run high compression if it can reach the same accuracy as low compression while communicating fewer bytes. To test this we ran ResNet-18 (He et al., 2016) with POWERSGD Rank-1 and Rank-2. We ran Rank-2 for 300 epochs and allowed Rank-1 to communicate the same amount as Rank-2. As observe in 8, POWERSGD Rank-1 cannot reach the same accuracy as POWERSGD Rank-2. Moreover ACCORDION still achieves performance at par with low compression, while using a smaller communication budget.

Table 3. ACCORDION using TOPK on CIFAR-10

Network	K(%)	Accuracy	Data Sent (Billion Floats)		Time (Seconds)	
Resnet-18	99	**94.2%**	2626.1	(1×)	33672	(1×)
	10	93.1%	262.8	(9.9×)	7957	(4.2×)
	ACCORDION	**93.9%**	976.7	(**2.8**×)	9356	(**3.6**×)
GoogleNet	99	**94.6%**	1430.9	(1×)	28476	(1×)
	10	94.1%	145.2	(9.8×)	13111	(2.1×)
	ACCORDION	**94.7%**	383.8	(**3.7**×)	16022	(**1.7**×)
Senet	99	**94.6%**	2648.7	(1×)	29977	(1×)
	10	93.8%	267.9	(9.8×)	8055	(3.7×)
	ACCORDION	**94.5%**	869.8	(**3.0**×)	13071	(**2.29**×)

Table 4. ACCORDION using TOPK on CIFAR-100

Network	K(%)	Accuracy	Data Sent (Billion Floats)		Time (Seconds)	
Resnet-18	99	**72.4%**	2636.9	(1×)	53460	(1×)
	25	71.3%	659.4	(3.9×)	6176	(8.6×)
	ACCORDION	**72.3%**	923.6	(**2.8**×)	14223	(**3.8**×)
GoogleNet	99	**76.2%**	1452.4	(1×)	28579	(1×)
	25	75.3%	367.3	(3.9×)	12810	(2.23×)
	ACCORDION	**76.2%**	539.9	(**2.7**×)	15639	(**1.82**×)
Senet	99	**72.8%**	2659.5	(1×)	30312	(1×)
	25	71.9%	671.9	(3.9×)	7376	(4.1×)
	ACCORDION	**72.7%**	966.13	(**2.8**×)	10689	(**2.8**×)

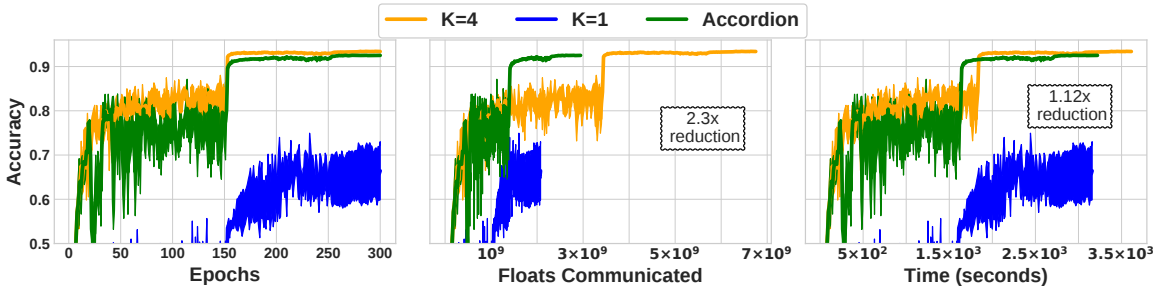


Figure 6. **ACCORDION using POWERSGD with $\ell_{\text{low}} =$ rank 4 and $\ell_{\text{high}} =$ rank 1 on VGG-19bn:** We show ACCORDION being able to bridge more that 25% of accuracy difference with 2.3× less communication .

Table 5. ACCORDION switching Batch Size on CIFAR-10

Network	K(%)	Accuracy	Data Sent (Billion Floats)		Time (Seconds)	
Resnet-18	512	**94.5%**	326.5	(1×)	5009	(1×)
	4096	93.2%	40.22	(8×)	1721	(2.9×)
	ACCORDION	**94.4%**	59.22	(**5.6**×)	1959	(**2.5**×)
GoogLeNet	512	**94.7%**	181.28	(1×)	12449	(1×)
	4096	93.1%	22.19	(8.1×)	3386	(3.67×)
	ACCORDION	**94.7%**	32.68	(**5.5**×)	6220	(**2.0**×)
DenseNet	512	**93.9%**	29.4	(1×)	14489	(1×)
	4096	93.1%	3.6	(8.1×)	2759	(5.2×)
	ACCORDION	**94.0%**	5.3	(**5.5**×)	3547	(**4**×)

Table 6. ACCORDION switching Batch Size on CIFAR-100

Network	K(%)	Accuracy	Data Sent (Billion Floats)		Time (Seconds)	
Resnet-18	512	**73.1%**	326.5	(1×)	5096	(1×)
	4096	70.0%	40.39	(8×)	1635	(3.1×)
	ACCORDION	**73.3%**	54.96	(**5.5**×)	1852	(**2.7**×)
GoogleNet	512	**77.0%**	182.1	(1×)	12443	(1×)
	4096	73.7%	22.5	(8.1×)	5755	(2.1×)
	ACCORDION	**77.0%**	33.1	(**5.4**×)	6228	(**2.0**×)
DenseNet	512	**73.7%**	30.126	(1×)	14928	(1×)
	4096	70.0%	3.72	(8×)	2775	(5.3×)
	ACCORDION	**73.9%**	5.48	(**5.4**×)	3585	(**4.1**×)

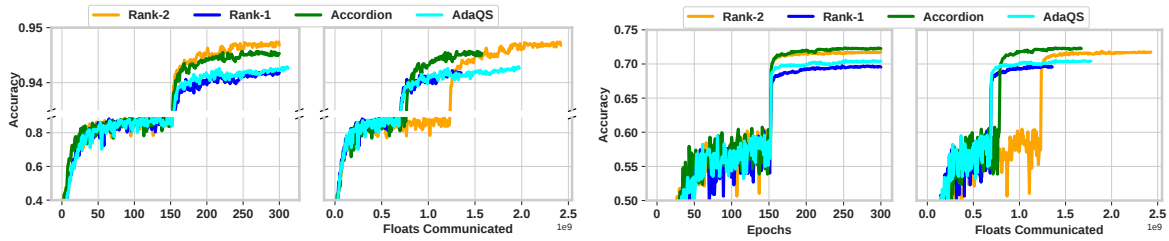
6 FUTURE WORK AND LIMITATIONS

We have shown that ACCORDION provides significant benefits over static compression schemes without compromising accuracy or requiring additional hyper-parameter tuning. Next, we would like to point out some of the future directions and current limitations of our approach.

- **Theoretical Understanding of Critical Regimes:** Although our work is motivated by several previous works (Jastrzębski et al., 2019; Jastrzebski et al., 2020; Achille et al., 2019) which have discovered and analyzed critical regimes, building a better theoretical understanding of how change in gradient norm relates to critical regimes is an avenue for future work.

- **Equivalence between batch size and gradient compression:** While we have shown that there might be a connection between batch size and gradient compression, rigorously verifying this connection can lead to better theoretical understanding of our technique.

- **Choosing ℓ_{low}, ℓ_{high}, B_{low} and B_{high}**: Choosing the low and high compression ratios used by ACCORDION is currently left to the user. In case of POWERSGD we chose ℓ_{low} based on the results of Vogels et al. (2019) where authors showed Rank 2 and 4 achieved



(a) ResNet-18 trained on CIFAR-10

(b) ResNet-18 trained of CIFAR-100

Figure 7. **Comparison with AdaQS:** We compare ACCORDION against AdaQS (Guo et al., 2020) on CIFAR-10 and CIFAR-100. We use POWERSGDas the Gradient Compressor. Even though AdaQS communicates more that ACCORDION it still looses accuracy compared to low compression. ACCORDION on other hand with less communication is able reach the accuracy of low compression.

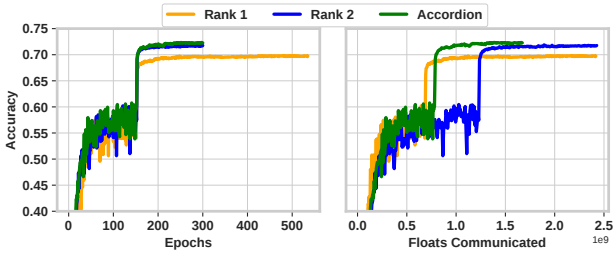


Figure 8. **Evaluating high compression(Rank-1) training when allowed to same total communication budget as low compression(Rank-2):**ResNet-18 trained on CIFAR-100, using POWERSGD. we observe that even when we allow highly compressed training to communicate same amount as low compressed training it is still not possible to get the same accuracy. Meanwhile ACCORDION is able to achieve the same accuracy as low compressed training but with much lesser communication.

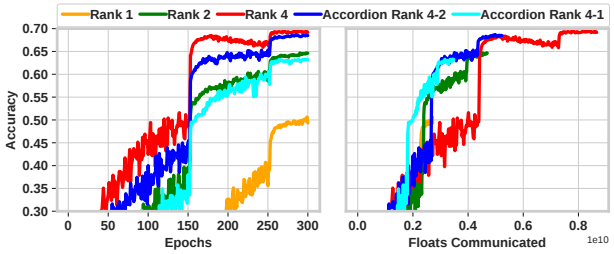


Figure 9. **Limitation of ACCORDION:** For the specific case of VGG-19 trained on CIFAR-100we observe that when ACCORDION switches between Rank-1(50% accuracy) and Rank-4(68% accuracy) it only reaches accuracy of Rank-2(63% accuracy) but when allowed to switch between Rank-2 and Rank-4 it reaches the accuracy of Rank-4 with around 2.3× lesser communication. This shows that oftentimes the ℓ_{low} needs to used chosen carefully.

same accuracy as syncSGD, making Rank 1 the natural choice for ℓ_{high}. Similarly for TopK we chose ℓ_{low} to be close to SGD and ℓ_{high} to provide significant communication saving. However these settings do not work for all models. For example, in case of VGG19 on Cifar100 with POWERSGD we observed that using $\ell_{high} = Rank1$ leads to a model with very low accuracy(50%). In that case ACCORDION cannot match the accuracy of $\ell_{low} = Rank4$ as shown in Figure 9. Automating these choices has the potential of making gradient compression techniques much more user friendly and is an avenue for future work.

- **Verifying ACCORDION on more Models:** We have evaluated ACCORDION on popular Vision and Language models. In future we plan to extend ACCORDION to newer models like DLRM (Naumov et al., 2019), Transformers (Vaswani et al., 2017) and Graph Neural Networks (Zhang et al., 2020).

- **Jointly adapting batch size and gradient compression:** In this work, we study gradient compression and batch size scaling independently. Understanding how to vary both of them in tandem might lead to even larger gains in the future.

7 CONCLUSION

In this paper we propose ACCORDION, an adaptive gradient compression method that can automatically switch between low and high compression. ACCORDION works by choosing low compression in critical regimes of training and high compression elsewhere. We show that such regimes can be efficiently identified using the rate of change of the gradient norm and that our method matches critical regimes identified by prior work. We also discuss connections between the compression ratio and batch size used for training and show that the insights used in ACCORDION are supported by prior work in adaptive batch size tuning. Finally, we show that ACCORDION is effective in practice and can save upto 3.7× communication compared to using low compression without affecting generalization performance. Overall, our work provides a new principled approach for building adaptive-hyperparameter tuning algorithms, and we believe that further understanding of critical regimes in neural network training can help us design better hyperparameter tuning algorithms in the future.

ACKNOWLEDGEMENT

We will like to thank the anonymous reviewers for their insightful comments. Kangwook Lee is supported by NSF/Intel Partnership on Machine Learning for Wireless Networking Program under Grant No. CNS-2003129. Shivaram Venkataraman is supported by the National Science Foundation grant CNS-1838733, a Facebook faculty research award and by the Office of the Vice Chancellor for Research and Graduate Education at UW-Madison with funding from the Wisconsin Alumni Research Foundation. Dimitris Papailiopoulos is supported by an NSF CAREER Award #1844951, two Sony Faculty Innovation Awards, an AFOSR & AFRL Center of Excellence Award FA9550-18-1-0166, and an NSF TRIPODS Award #1740707.

REFERENCES

Pytorch-cifar10, a. URL https://github.com/kuangliu/pytorch-cifar.

Pytorch-cifar100, b. URL https://github.com/weiaicunzai/pytorch-cifar100.

Acharya, J., De Sa, C., Foster, D. J., and Sridharan, K. Distributed learning with sublinear communication. *arXiv preprint arXiv:1902.11259*, 2019.

Achille, A., Rovere, M., and Soatto, S. Critical learning periods in deep networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=BkeStsCcKQ.

Aji, A. F. and Heafield, K. Sparse communication for distributed gradient descent. *arXiv preprint arXiv:1704.05021*, 2017.

Alistarh, D., Grubic, D., Li, J., Tomioka, R., and Vojnovic, M. Qsgd: Communication-efficient sgd via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*, pp. 1709–1720, 2017.

Bernstein, J., Wang, Y.-X., Azizzadenesheli, K., and Anandkumar, A. signsgd: Compressed optimisation for non-convex problems. *arXiv preprint arXiv:1802.04434*, 2018.

Chen, C.-Y., Choi, J., Brand, D., Agrawal, A., Zhang, W., and Gopalakrishnan, K. Adacomp: Adaptive residual gradient compression for data-parallel distributed training. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., aurelio Ranzato, M., Senior, A., Tucker, P., Yang, K., Le, Q. V., and Ng, A. Y. Large scale distributed deep networks. In *Advances in Neural Information Processing Systems 25*, pp. 1223–1231. 2012.

Devarakonda, A., Naumov, M., and Garland, M. Adabatch: Adaptive batch sizes for training deep neural networks. *arXiv preprint arXiv:1712.02029*, 2017.

Dutta, S., Wang, J., and Joshi, G. Slow and stale gradients can win the race. *arXiv preprint arXiv:2003.10579*, 2020.

Frankle, J., Schwab, D. J., and Morcos, A. S. The early phase of neural network training. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=Hkl1iRNFwS.

Golmant, N., Vemuri, N., Yao, Z., Feinberg, V., Gholami, A., Rothauge, K., Mahoney, M. W., and Gonzalez, J. On the computational inefficiency of large batch sizes for stochastic gradient descent. *arXiv preprint arXiv:1811.12941*, 2018.

Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., and He, K. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Guo, J., Liu, W., Wang, W., Han, J., Li, R., Lu, Y., and Hu, S. Accelerating distributed deep learning by adaptive gradient quantization. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1603–1607, 2020.

Gur-Ari, G., Roberts, D. A., and Dyer, E. Gradient descent happens in a tiny subspace. *arXiv preprint arXiv:1812.04754*, 2018.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hoffer, E., Hubara, I., and Soudry, D. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In *Advances in Neural Information Processing Systems*, pp. 1731–1741, 2017.

Honnibal, M. and Montani, I. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, 2017.

Hu, J., Shen, L., and Sun, G. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, 2018.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Iandola, F. N., Moskewicz, M. W., Ashraf, K., and Keutzer, K. Firecaffe: near-linear acceleration of deep neural network training on compute clusters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2592–2600, 2016.

Jastrzebski, S., Szymczak, M., Fort, S., Arpit, D., Tabor, J., Cho*, K., and Geras*, K. The break-even point on optimization trajectories of deep neural networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1g87C4KwB.

Jastrzębski, S., Kenton, Z., Ballas, N., Fischer, A., Bengio, Y., and Storkey, A. On the relation between the sharpest directions of DNN loss and the SGD step length. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SkgEaj05t7.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

Krizhevsky, A. et al. Learning multiple layers of features from tiny images. 2009.

Lin, T., Stich, S. U., Patel, K. K., and Jaggi, M. Don't use large mini-batches, use local sgd. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=B1eyO1BFPr.

Lin, Y., Han, S., Mao, H., Wang, Y., and Dally, W. J. Deep gradient compression: Reducing the communication bandwidth for distributed training. *arXiv preprint arXiv:1712.01887*, 2017.

Luo, L., West, P., Nelson, J., Krishnamurthy, A., and Ceze, L. Plink: Discovering and exploiting locality for accelerated distributed training on the public cloud. In *Proceedings of Machine Learning and Systems 2020*, pp. 82–97. 2020.

Mattson, P., Cheng, C., Diamos, G., Coleman, C., Micikevicius, P., Patterson, D., Tang, H., Wei, G.-Y., Bailis, P., Bittorf, V., Brooks, D., Chen, D., Dutta, D., Gupta, U., Hazelwood, K., Hock, A., Huang, X., Kang, D., Kanter, D., Kumar, N., Liao, J., Narayanan, D., Oguntebi, T., Pekhimenko, G., Pentecost, L., Janapa Reddi, V., Robie, T., St John, T., Wu, C.-J., Xu, L., Young, C., and Zaharia, M. Mlperf training benchmark. In *Proceedings of Machine Learning and Systems 2020*, pp. 336–349. 2020.

Naumov, M., Mudigere, D., Shi, H.-J. M., Huang, J., Sundaraman, N., Park, J., Wang, X., Gupta, U., Wu, C.-J., Azzolini, A. G., et al. Deep learning recommendation model for personalization and recommendation systems. *arXiv preprint arXiv:1906.00091*, 2019.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pp. 8024–8035, 2019.

Seide, F., Fu, H., Droppo, J., Li, G., and Yu, D. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

Sergeev, A. and Del Balso, M. Horovod: fast and easy distributed deep learning in tensorflow. *arXiv preprint arXiv:1802.05799*, 2018.

Shallue, C. J., Lee, J., Antognini, J., Sohl-Dickstein, J., Frostig, R., and Dahl, G. E. Measuring the effects of data parallelism on neural network training. *arXiv preprint arXiv:1811.03600*, 2018.

Shi, S., Chu, X., Cheung, K. C., and See, S. Understanding top-k sparsification in distributed deep learning. *arXiv preprint arXiv:1911.08772*, 2019a.

Shi, S., Wang, Q., Zhao, K., Tang, Z., Wang, Y., Huang, X., and Chu, X. A distributed synchronous sgd algorithm with global top-k sparsification for low bandwidth networks. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pp. 2238–2247. IEEE, 2019b.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Smith, S. L., Kindermans, P.-J., Ying, C., and Le, Q. V. Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*, 2017.

Stich, S. U. Local sgd converges fast and communicates little. In *ICLR 2019 ICLR 2019 International Conference on Learning Representations*, number CONF, 2019.

Stich, S. U. and Karimireddy, S. P. The error-feedback framework: Better rates for sgd with delayed gradients and compressed communication. *arXiv preprint arXiv:1909.05350*, 2019.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

Tibshirani, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

Vogels, T., Karimireddy, S. P., and Jaggi, M. Powersgd: Practical low-rank gradient compression for distributed optimization. In *Advances in Neural Information Processing Systems*, pp. 14236–14245, 2019.

Wang, H., Sievert, S., Liu, S., Charles, Z., Papailiopoulos, D., and Wright, S. Atomo: Communication-efficient learning via atomic sparsification. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 9850–9861. Curran Associates, Inc., 2018.

Wang, J. and Joshi, G. Adaptive communication strategies to achieve the best error-runtime trade-off in local-update sgd. *arXiv preprint arXiv:1810.08313*, 2018.

Wang, J., Liang, H., and Joshi, G. Overlap local-sgd: An algorithmic approach to hide communication delays in distributed sgd. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8871–8875. IEEE, 2020.

Wangni, J., Wang, J., Liu, J., and Zhang, T. Gradient sparsification for communication-efficient distributed optimization. In *Advances in Neural Information Processing Systems*, pp. 1299–1309, 2018.

Wen, W., Xu, C., Yan, F., Wu, C., Wang, Y., Chen, Y., and Li, H. Terngrad: Ternary gradients to reduce communication in distributed deep learning. In *Advances in neural information processing systems*, pp. 1509–1519, 2017.

Yao, Z., Gholami, A., Arfeen, D., Liaw, R., Gonzalez, J., Keutzer, K., and Mahoney, M. Large batch size training of neural networks with adversarial training and second-order information. *arXiv preprint arXiv:1810.01021*, 2018a.

Yao, Z., Gholami, A., Lei, Q., Keutzer, K., and Mahoney, M. W. Hessian-based analysis of large batch training and robustness to adversaries. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 4949–4959. Curran Associates, Inc., 2018b.

Yin, D., Pananjady, A., Lam, M., Papailiopoulos, D., Ramchandran, K., and Bartlett, P. Gradient diversity empowers distributed learning. *arXiv preprint arXiv:1706.05699*, 143, 2017.

You, Y., Gitman, I., and Ginsburg, B. Scaling sgd batch size to 32k for imagenet training. *arXiv preprint arXiv:1708.03888*, 6, 2017.

Yu, M., Lin, Z., Narra, K., Li, S., Li, Y., Kim, N. S., Schwing, A., Annavaram, M., and Avestimehr, S. Gradiveq: Vector quantization for bandwidth-efficient gradient aggregation in distributed cnn training. *arXiv preprint arXiv:1811.03617*, 2018.

Zhang, Z., Cui, P., and Zhu, W. Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

A DETAILED EXPERIMENTAL SETTINGS

Data preprocessing For preprocessing the images of CIFAR-10 and CIFAR-100 datasets, we follow the standard data augmentation and normalization process. For data augmentation, random cropping and horizontal random flipping are used. Each color channel is normalized with it's mean and standard deviation. Where $\mu_r = 0.49, \mu_g = 0.48, \mu_b = 0.45$ are mean of the red, green and blue channels respectively. And $\sigma_r = 0.25, \sigma_g = 0.24, \sigma_b = 0.26$ are corresponding standard deviations. Each channel pixel is normalized by subtracting the mean value in this color channel and then divided by the standard deviation of this color channel. For pre-processing WIKITEXT-2 we used the default english tokenizer in Spacy (Honnibal & Montani, 2017).

Table 7. Hyperparameters

Dataset	CIFAR-10 and CIFAR-100	WIKITEXT-2
LR	0.1 × Number of Workers.	2.5 × Number of Workers
LR Decay	/10 at epoch 150 and 250	/10 at epoch 60 and 80
LR warmup	Linearly for the first 5 epochs, starting from 0.1	Linearly for the first 5 epochs, starting from 2.5
Total Epochs	300	90
Optimizer	Nesterov	Nesterov
Momentum	0.9	0.9
Repetition	3 times with different random seeds	3 times with different random seeds
Error Bars	95% Confidence Interval	95% Confidence Interval

Hyperparameters For training we used the standard hyper-parameters from prior work. We used POWERSGD with memory term as suggested in (Vogels et al., 2019) and used the same learning rate schedule. Table 7 provides details of the hyper-parameters used in our experiments. We used learning rate warmup as suggested by Goyal et al. (Goyal et al., 2017) for all our baselines as well as ACCORDION. We start with learning rate of 0.1 and linearly scale the learning rate 5 epochs to $0.1 \times \frac{BatchSize}{128}$.

Additional Details for Batch Size experiment When trying to run extremely large batch sizes on 4 *p3.2xlarge* we started running out of memory. To make sure that our communication overhead for each round remains same instead of using more GPU's, we simulated large batch size in Pytorch (Paszke et al., 2019). Which means we did multiple backward passes to accumulate the gradients before communicating and applying them to the weights.Moreover for training stability as done by (Yao et al., 2018a) we only allow ACCORDION to increase batch size.

B CONNECTION BETWEEN GRADIENT COMPRESSION AND BATCH SIZE

The connection between gradient compression and batch size tuning can be made more formal under the following assumption: "each stochastic gradient is the sum of a sparse mean and a dense noise", i.e.,

$$\nabla_w \ell(w; x_i, y_i) = \underbrace{\mathbb{E}_j \nabla_w \ell(w; x_j, y_j)}_{\text{sparse, large magnitudes}} + \underbrace{(\nabla_w \ell(w; x_i, y_i) - \mathbb{E}_j \nabla_w \ell(w; x_j, y_j))}_{\text{dense, small magnitudes}} \tag{2}$$

Under this assumption, we can see that "large batch gradient ≈ highly compressed gradient", as a large batch gradient will be close to $\mathbb{E}_j \nabla_w \ell(w; x_j, y_j)$ by the law of large numbers, and a highly compressed gradient will also pick up the same sparse components. Similarly, a small batch gradient is equivalent to weakly compressed gradient.

We show that the above assumption on gradient properties can hold for limited scenarios by considering a simple LASSO example.Consider a model whose goal is to minimize $1/2 \cdot \|Xw - y\|_2^2 + \lambda \|w\|$, where positive-class data points $x_+ \sim \mathcal{N}(\mu, \sigma^2 I)$, negative-class data points $x_- \sim \mathcal{N}(-\mu, \sigma^2 I)$, and $P(Y = +1) = P(Y = -1) = 1/2$. Here, w is sparse for a properly chosen value of λ due to the shrinkage operation (Tibshirani, 1996). Then, we have the following lemma, which implies that the gradient modeling described above holds w.h.p.

Lemma 1. *If μ is k_1-sparse and w is k_2-sparse, $\mathbb{E}_j \nabla_w \ell(w; x_j, y_j)$ is $k_1 + k_2$-sparse. Let us denote by γ the minimum absolute value of non-zero entries of $\mathbb{E}_j \nabla_w$. Then, for any positive integer n and $\epsilon > 0$, there exists a sufficiently small enough $\sigma > 0$ such that $\mathbb{P}(\max_{i \in [n]} \|\nabla_w \ell(w; x_i, y_i) - \mathbb{E}_j \nabla_w \ell(w; x_j, y_j)\|_\infty < \gamma) \geq 1 - \epsilon$.*

Proof. We first show that $\mathbb{E}_j \nabla_w \ell(w; x_j, y_j)$ is $k_1 + k_2$-sparse. Since $\nabla_w \ell(w; x_i, y_i) = x_i(x_i^\top w) - x_i y_i + \lambda \text{sign}(w)$, we have

$$\begin{aligned}\mathbb{E}\nabla_w \ell(w; x_j, y_j) &= \mathbb{E}[x_i(x_i^\top w) - x_i y_i + \lambda \text{sign}(w)] \\ &= \mathbb{E}[x_i x_i^\top] w + \lambda \text{sign}(w).\end{aligned}$$

Since $\mathbb{E}[x_i x_i^\top] = I + \mu\mu^\top$,

$$\begin{aligned}\mathbb{E}\nabla_w \ell(w; x_j, y_j) &= (I + \mu\mu^\top) w + \lambda \text{sign}(w) \\ &= w + \lambda \text{sign}(w) + \mu(\mu^\top w).\end{aligned}$$

The first two terms are k_1-sparse sharing the support. The sparsity of the last term is upper bounded by μ, hence k_2-sparse.

We now prove $\mathbb{P}(\max_{i\in[n]} \|\nabla_w \ell(w; x_i, y_i) - \mathbb{E}_j \nabla_w \ell(w; x_j, y_j)\|_\infty < \gamma) \geq 1 - \epsilon$. Note that it is sufficient to show $\mathbb{P}(\|\nabla_w \ell(w; x_i, y_i) - \mathbb{E}_j \nabla_w \ell(w; x_j, y_j)\|_\infty \geq \gamma) \leq \epsilon/n$ for all i. Therefore, it is sufficient to show $\mathbb{P}((\nabla_w \ell(w; x_i, y_i) - \mathbb{E}_j \nabla_w \ell(w; x_j, y_j))_j \geq \gamma) \leq \epsilon/(nd)$ for all i, j, where d is the dimension of the model parameter. By Chebyshev's inequality, we have $\mathbb{P}((\nabla_w \ell(w; x_i, y_i) - \mathbb{E}_j \nabla_w \ell(w; x_j, y_j))_j \geq \gamma) \leq \frac{\text{Var}[(\nabla_w \ell(w; x_i, y_i)_j]}{\gamma^2}$. It is easy to see that $\text{Var}[(\nabla_w \ell(w; x_i, y_i)_j] \leq (\sigma^4 + 2\|\mu\|^2_{\max}\sigma^2)\|w\|_2^2 + \sigma^2$, which converges to 0 as $\sigma \to 0$. Therefore, one can always find a small enough $\sigma > 0$ such that $\frac{(\sigma^4 + 2\|\mu\|^2_{\max}\sigma^2)\|w\|_2^2 + \sigma^2}{\gamma^2} = \frac{\epsilon}{nd}$, which is a sufficient condition for the desired inequality. □

C ACCORDION ON EXTREMELY LARGE BATCH SIZE

To push the limits of batch size scaling further we tried using ACCORDION for scaling CIFAR-10 on ResNet-18 to batch size of 16,384.We observed that using ACCORDION looses around (1.6%) accuracy compared to using batch size 512. Interestingly we also observe that when ACCORDION first switches the batch size there is a rapid drop, but then training immediately recovers.

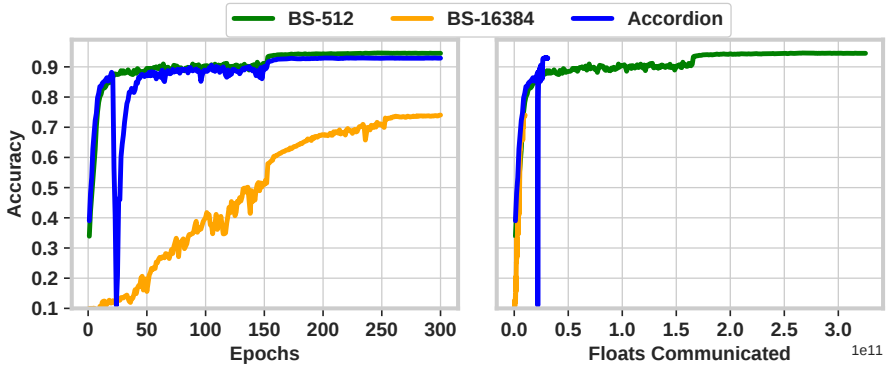


Figure 10. **Using Extremely Large Batch Size:** We observe that ACCORDION looses around 1.6% accuracy when we use batch size of 16,384. Showing ACCORDION can often prevent large accuracy losses while providing massive gains.

D RESULTS AND DETAILED ANALYSIS

We present detailed analysis with error bars for the results presented in Tables 1 to 4.

D.1 Language Model

Figure 11 shows ACCORDION's performance for training a 2 Layer LSTM on WIKITEXT-2. By automatically switching between TOPK99% and TOPK2% ACCORDION is able to bridge the perplexity score and achieve the same accuracy as TOPK99% with significantly less overall communication.

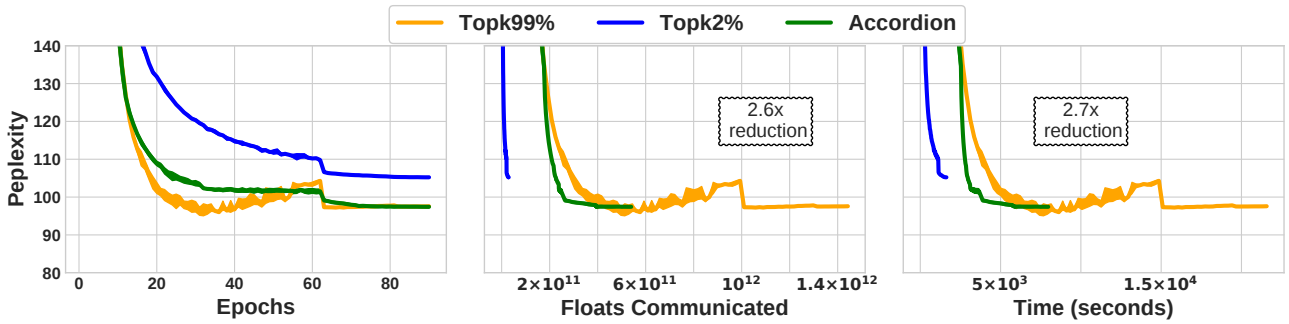
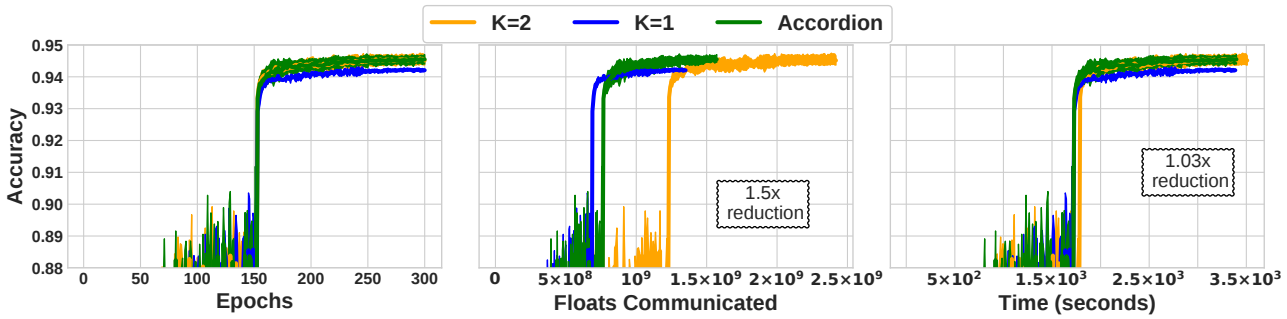


Figure 11. **ACCORDION using TOPK with $\ell_{\text{low}} =$ K 99% and $\ell_{\text{high}} =$ K 2% for training LSTM on WIKITEXT-2 (left:)** Perplexity vs Epochs, **(center:)** Perplexity vs Floats Communicated, **(right:)** Perplexity vs Time(seconds): ACCORDION significantly reduces total communication and training time compared to using $\ell_{\text{low}} =$ K 99% throughtout training

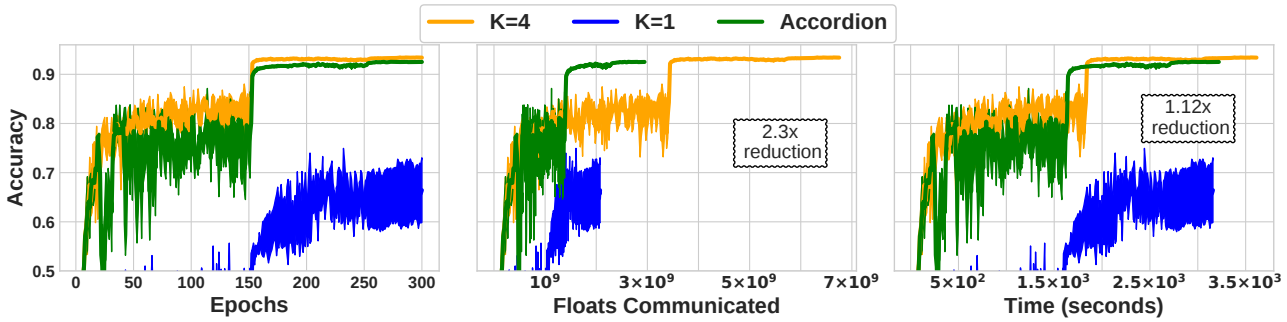
D.2 Computer Vision Models

We present graphs corresponding to the results stated in Tables 1 to 4 in the main text. In Figures 12 to 15 we provide details on ACCORDION's performance with error bars.

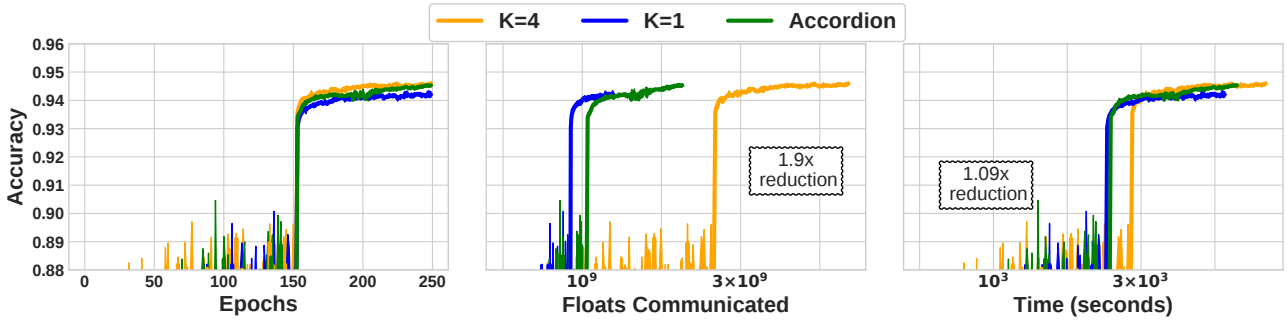
Figure 12. **ACCORDION on Computer Vision Models trained on CIFAR-10 using POWERSGD**



(a) ResNet-18 trained using POWERSGD with ℓ_{low} = Rank 4 and ℓ_{high} = Rank 1 for training

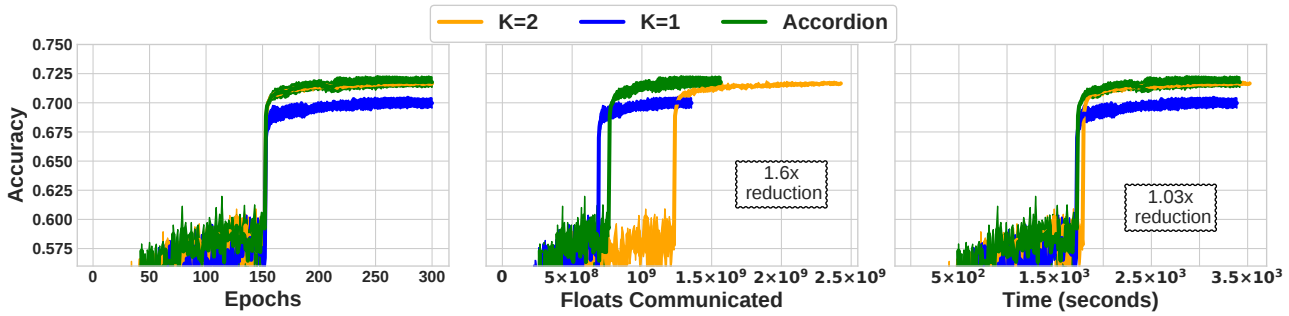


(b) VGG-19bn trained using POWERSGD with ℓ_{low} = Rank 4 and ℓ_{high} = Rank 1 for training

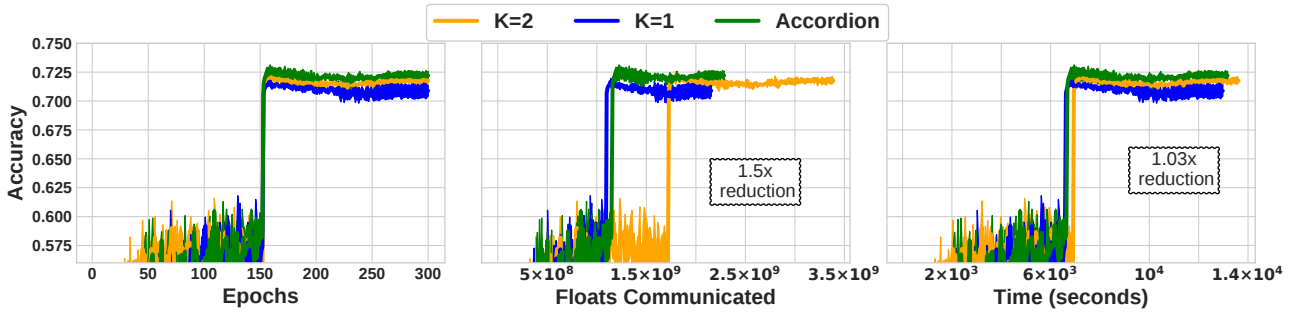


(c) SeNet trained using POWERSGD with ℓ_{low} = Rank 4 and ℓ_{high} = Rank 1 for training

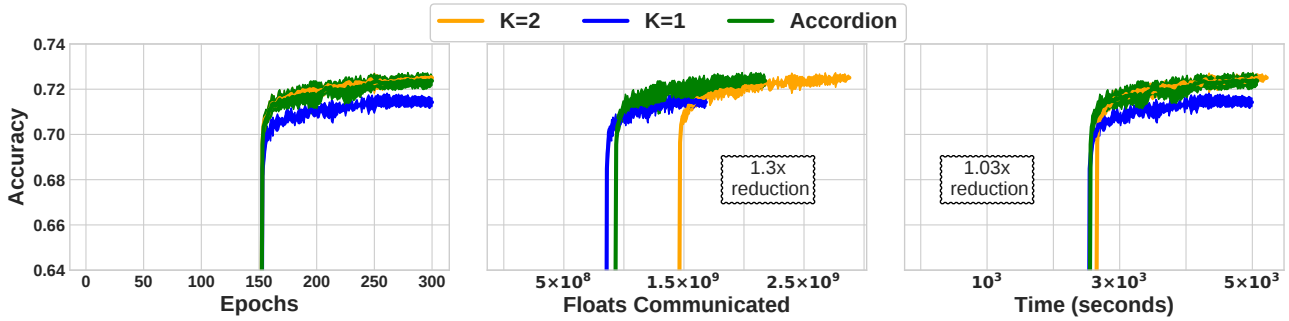
Figure 13. **ACCORDION on Computer Vision Models trained on CIFAR-100 using POWERSGD**



(a) ResNet-18 trained using POWERSGD with ℓ_{low} = Rank 2 and ℓ_{high} = Rank 1 for training

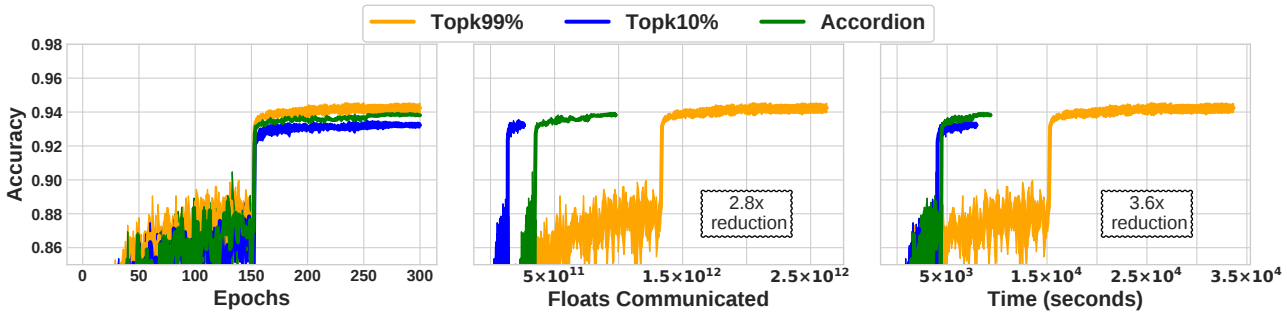


(b) DenseNet trained using POWERSGD with ℓ_{low} = Rank 2 and ℓ_{high} = Rank 1 for training

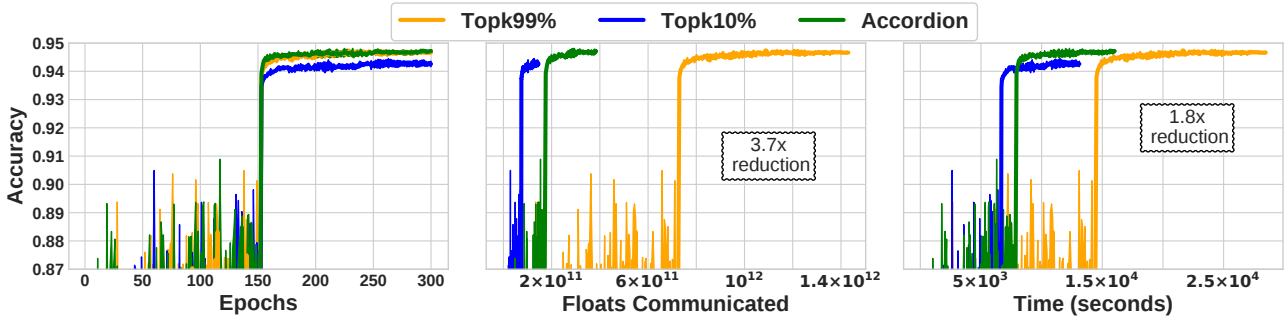


(c) SeNet trained using POWERSGD with ℓ_{low} = Rank 2 and ℓ_{high} = Rank 1 for training

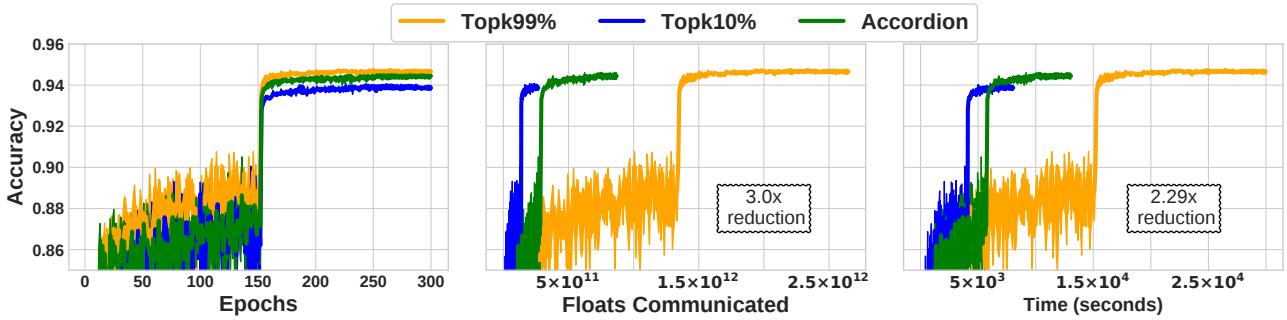
Figure 14. **ACCORDION on Computer Vision Models trained on CIFAR-10 using TOPK**



(a) ResNet-18 trained using TOPK with $\ell_{\text{low}} =$ K 99% and $\ell_{\text{high}} =$ K 10% for training

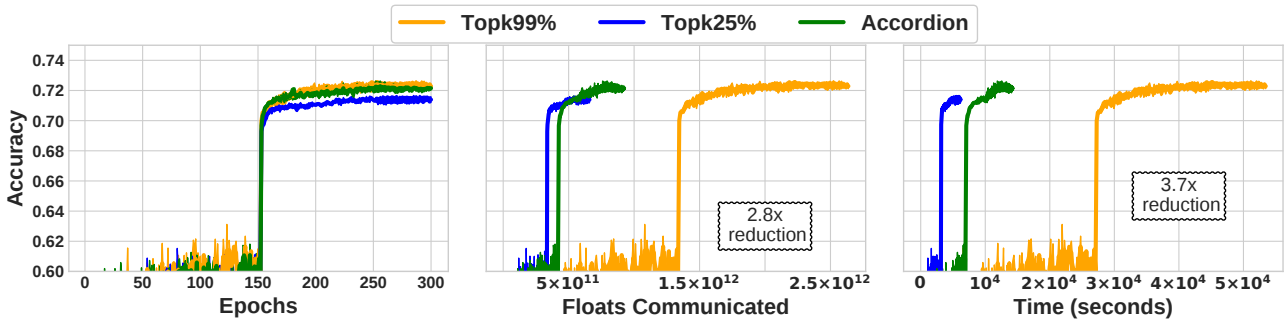


(b) GoogLeNet trained using TOPK with $\ell_{\text{low}} =$ K 99% and $\ell_{\text{high}} =$ K 10% for training

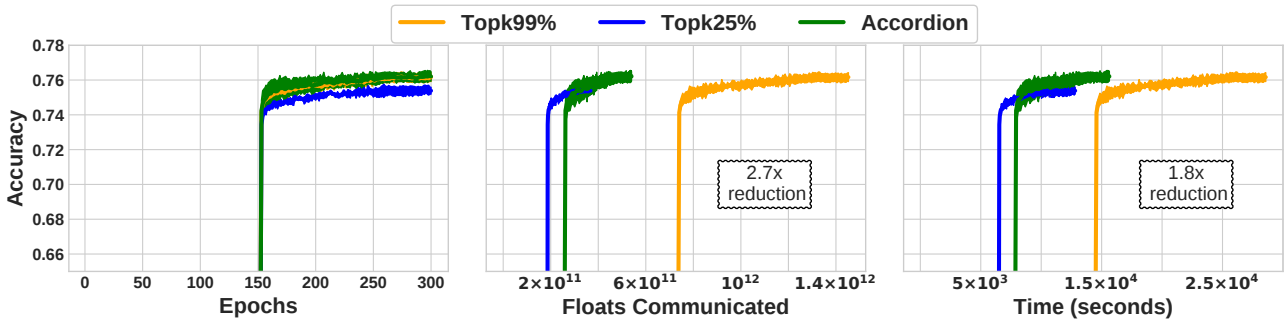


(c) SeNet trained using TOPK with $\ell_{\text{low}} =$ K 99% and $\ell_{\text{high}} =$ K 10% for training

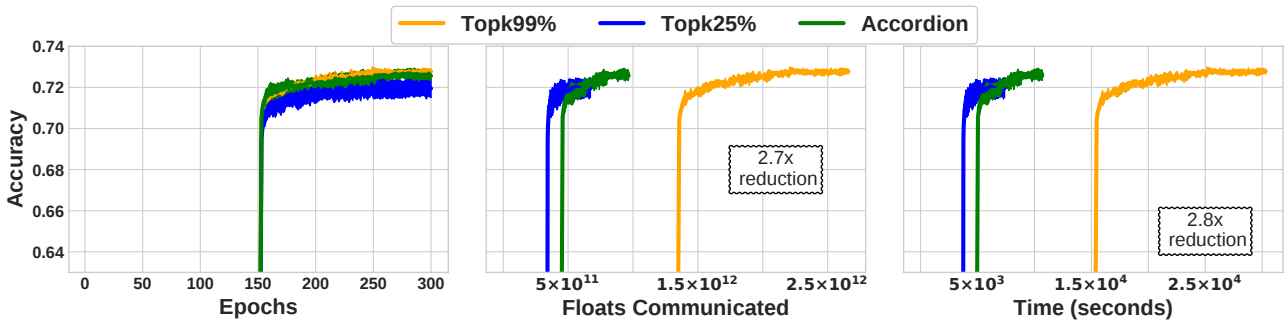
Figure 15. **ACCORDION on Computer Vision Models trained on CIFAR-100 using TOPK:**



(a) ResNet-18 trained using TOPK with $\ell_{\text{low}} =$ K 99% and $\ell_{\text{high}} =$ K 25% for training

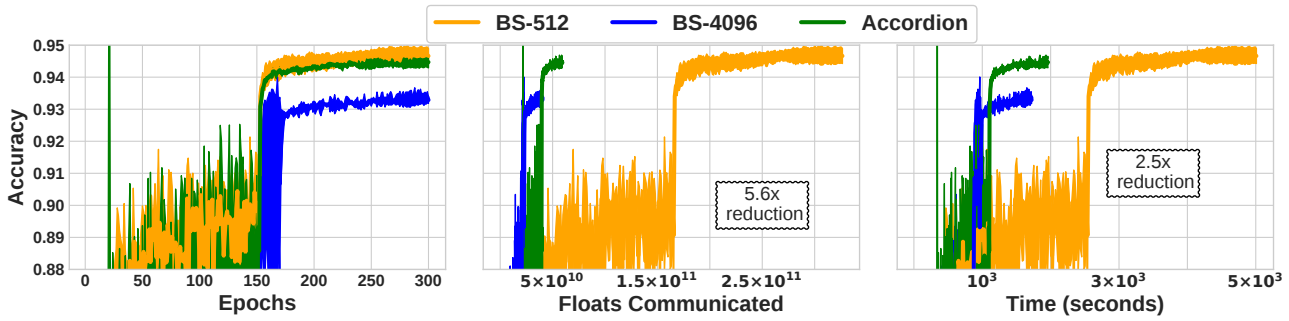


(b) GoogLeNet trained using TOPK with $\ell_{\text{low}} =$ K 99% and $\ell_{\text{high}} =$ K 25% for training

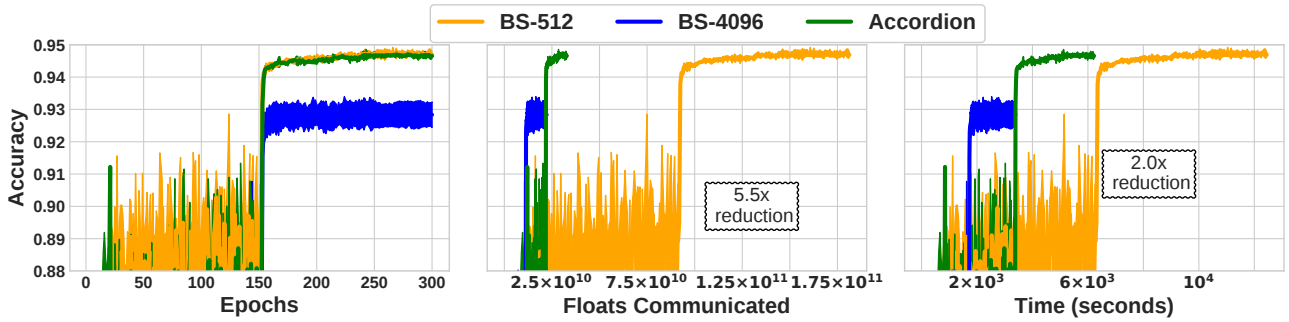


(c) SeNet trained using TOPK with $\ell_{\text{low}} =$ K 99% and $\ell_{\text{high}} =$ K 25% for training

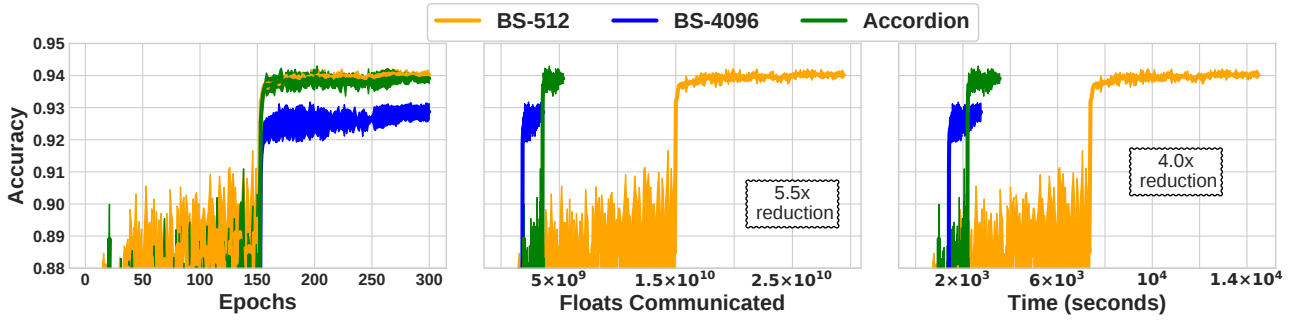
Figure 16. **ACCORDION on Computer Vision Models trained on CIFAR-10**



(a) ResNet-18 trained using Batch Size=512,Batch Size=4096 and ACCORDION



(b) GoogLeNet trained using Batch Size=512,Batch Size=4096 and ACCORDION

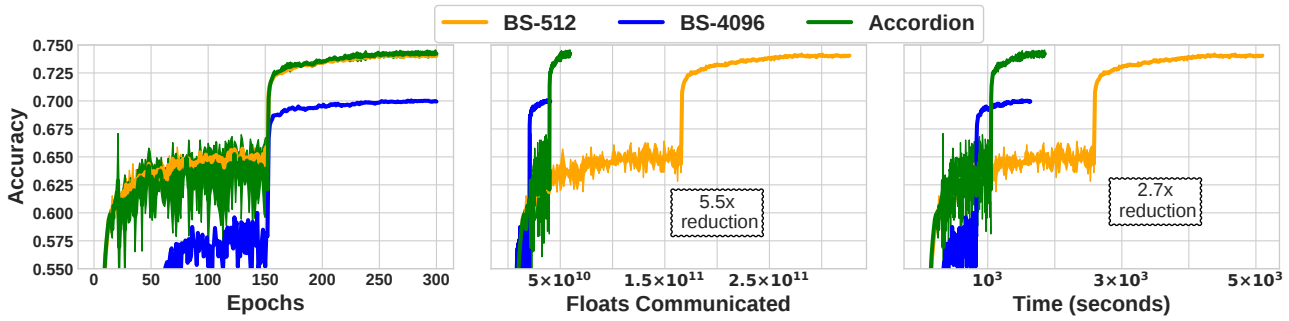


(c) DenseNet trained using Batch Size=512,Batch Size=4096 and ACCORDION

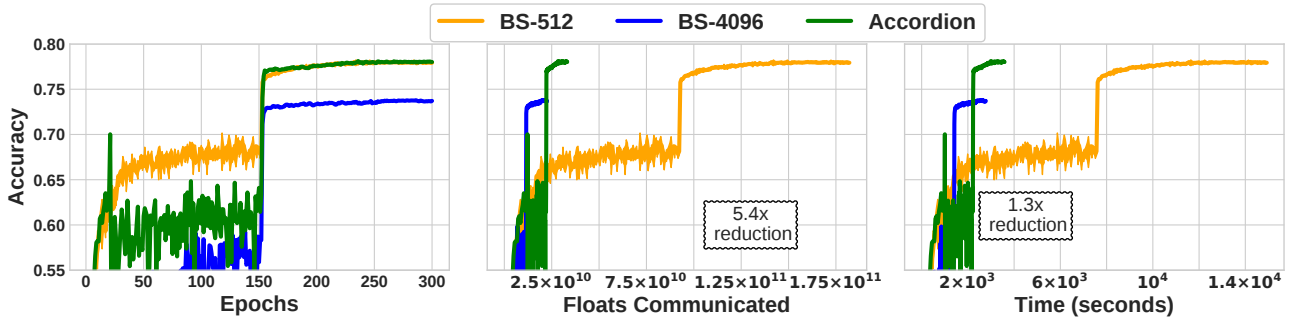
E DETAILED ANALYSIS OF BATCH SIZE RESULTS

In Figure 16 and 17 we provide detailed analysis for batch size. We show that, we ran experiments on CIFAR-10 and CIFAR-100. For three different CNN's, two recent CNN's with skip connections(ResNet-18, DenseNet) and one CNN without skip connections (Inception V1). Based on the findings of (Goyal et al., 2017; Devarakonda et al., 2017) we also modify the learning rate in the same proportion as the change in batch size.

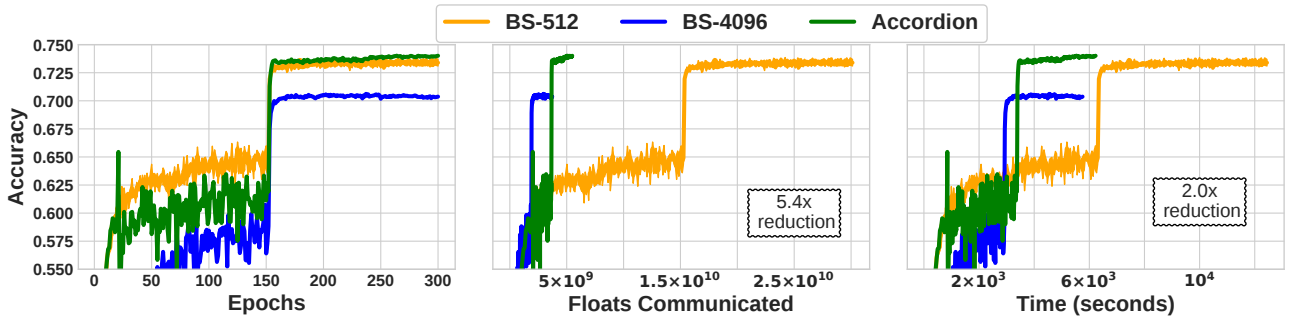
Figure 17. **ACCORDION on Computer Vision Models trained on CIFAR-100**



(a) ResNet-18 trained using Batch Size=512,Batch Size=4096 and ACCORDION



(b) GoogLeNet trained using Batch Size=512,Batch Size=4096 and ACCORDION



(c) DenseNet trained using Batch Size=512,Batch Size=4096 and ACCORDION

F COMPRESSION RATIO SELECTION OF ADASPARSE

In Figures 18 to 20 we show the compression ratio chosen by ACCORDION for different layers when training ResNet-18 on CIFAR-100 with POWERSGD as a gradient compressor. The layer numbers are associated with how PYTORCH indexes layers in the model. The missing layers numbers are 1 dimensional vectors which can not be compressed by POWERSGD.

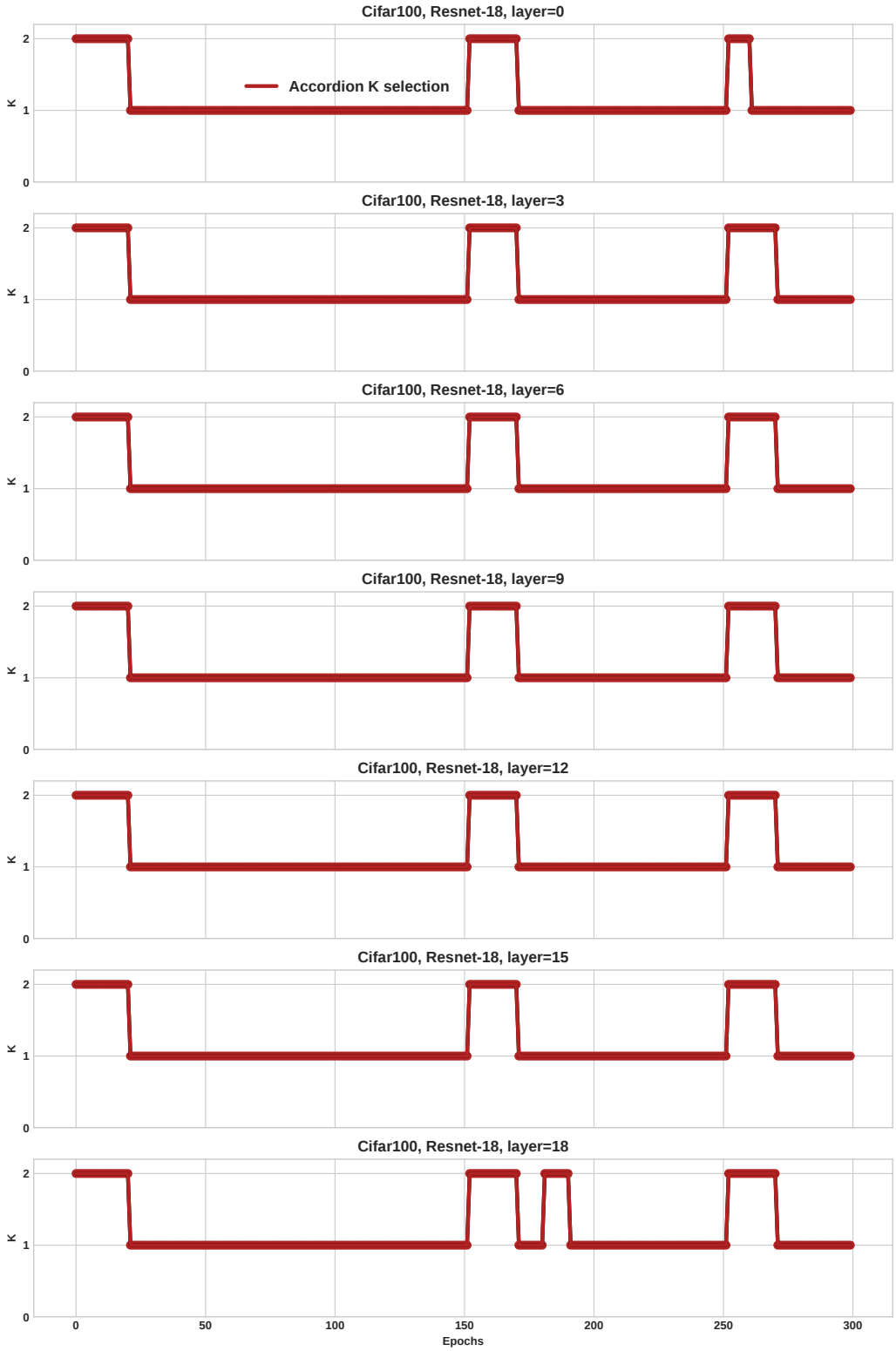


Figure 18. **Rank selected in Different Regions by ACCORDION when used with POWERSGD**

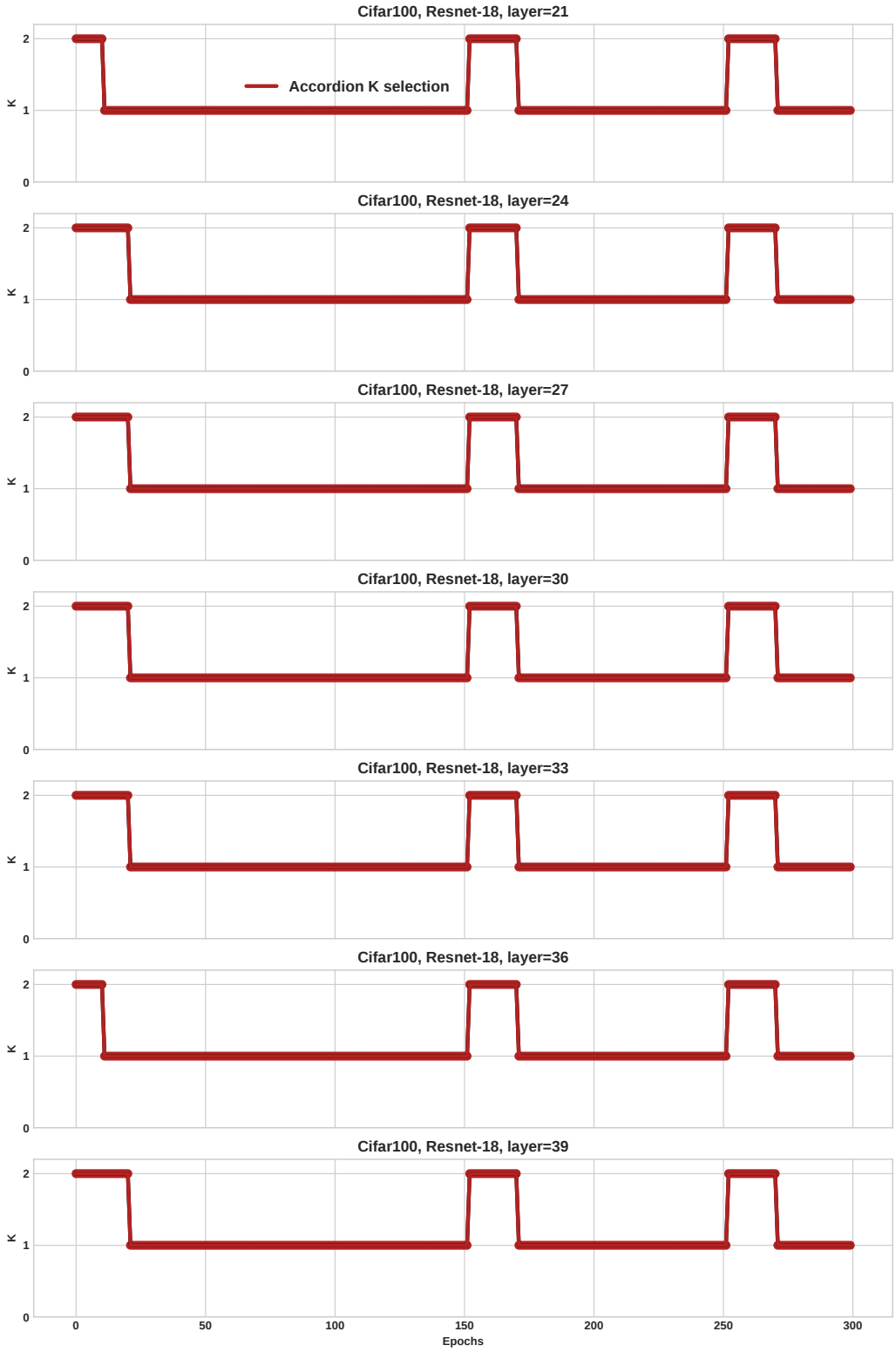


Figure 19. **Rank selected in Different Regions by ACCORDION when used with POWERSGD**

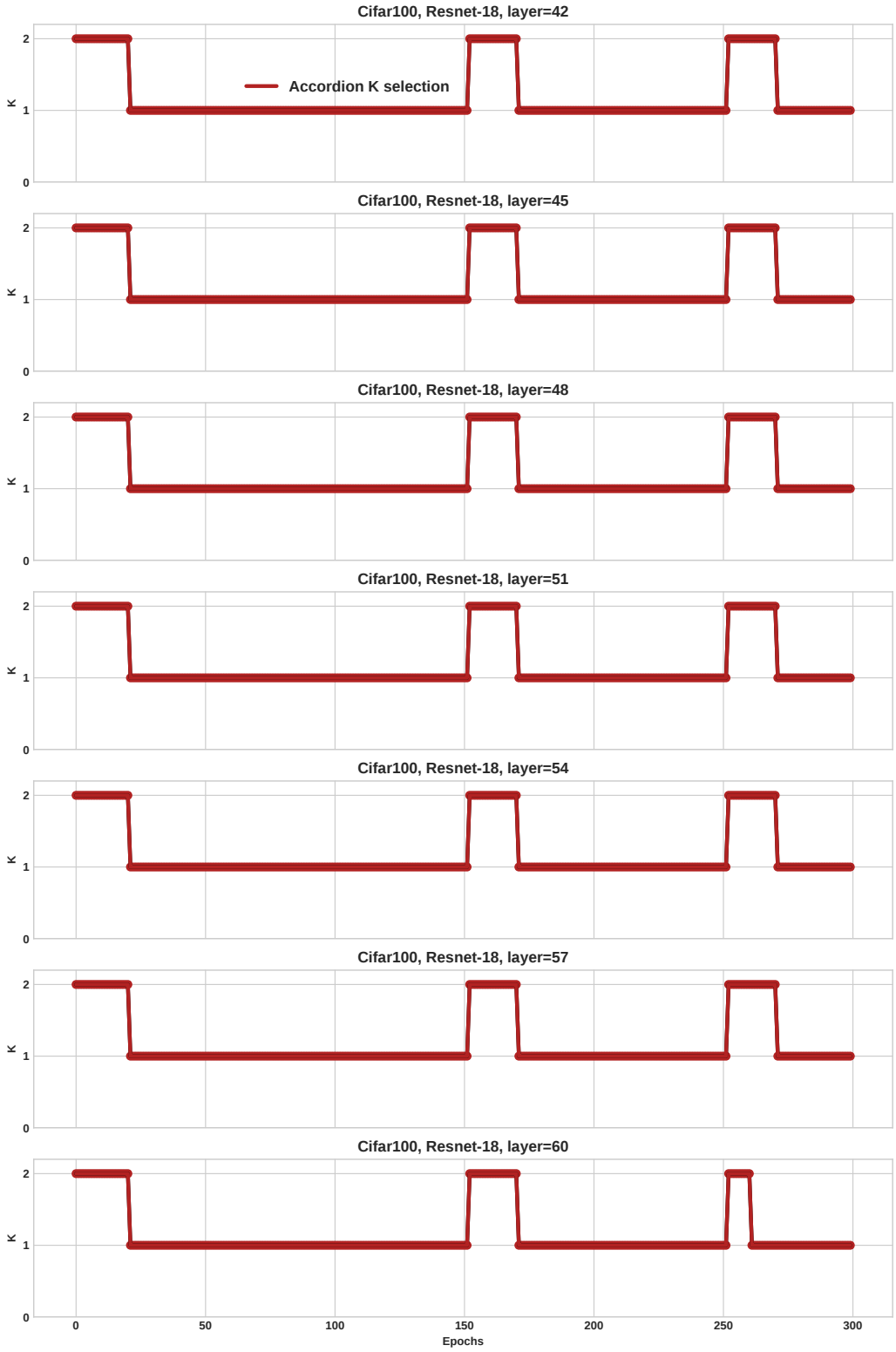


Figure 20. **Rank selected in Different Regions by ACCORDION when used with POWERSGD**

G MODEL DESCRIPTIONS

We use standard implementation for all the models. The model implementations for CIFAR-10 are borrowed from (pyt, a) and for CIFAR-100 are borrowed from (pyt, b). Here we present the total number of parameters in each of the model used.

Table 8. Total parameters when training CIFAR-10

Network	**Total Parameters**
ResNet-18	11173962
VGG-19bn	20565834
SeNet	11260354
WideResNet	36489290
DenseNet	1000618
GoogLeNet	6166250

Table 9. Total parameters when training CIFAR-100

Network	**Total Parameters**
ResNet-18	11220132
VGG-19bn	39327652
SeNet	11436256
WideResNet	36546980
DenseNet	1035268
GoogLeNet	6258500

Table 10. Total parameters when training WIKITEXT-2

Network	**Total Parameters**
2 Layer LSTM	28949319