# STEP: Learning N:M Structured Sparsity Masks from Scratch with Precondition

**Yucheng Lu** [1] **Shivani Agrawal** [2] **Suvinay Subramanian** [2] **Oleg Rybakov** [2]
**Christopher De Sa** [1] **Amir Yazdanbakhsh** [3]

## Abstract

Recent innovations on hardware (e.g. Nvidia A100) have motivated learning N:M structured sparsity masks from scratch for fast model inference. However, state-of-the-art learning recipes in this regime (e.g. SR-STE) are proposed for non-adaptive optimizers like momentum SGD, while incurring non-trivial accuracy drop for Adam-trained models like attention-based LLMs. In this paper, we first demonstrate such gap origins from poorly estimated second moment (i.e. variance) in Adam states given by the masked weights. We conjecture that learning N:M masks with Adam should take the critical regime of variance estimation into account. In light of this, we propose **STEP**, an Adam-aware recipe that learns N:M masks with two phases: first, **STEP** calculates a reliable variance estimate (*precondition phase*) and subsequently, the variance remains fixed and is used as a precondition to learn N:M masks (*mask-learning phase*). **STEP** automatically identifies the switching point of two phases by dynamically sampling variance changes over the training trajectory and testing the sample concentration. Empirically, we evaluate **STEP** and other baselines such as ASP and SR-STE on multiple tasks including CIFAR classification, machine translation and LLM fine-tuning (BERT-Base, GPT-2). We show **STEP** mitigates the accuracy drop of baseline recipes and is robust to aggressive structured sparsity ratios.

## 1. Introduction

Overparameterized Deep Neural Networks (DNNs) have shown promising performance on various applications, such as language modeling (Brown et al., 2020), translation (Vaswani et al., 2017) and image classification (Liu et al., 2021). However, modern DNNs usually contain millions of billions of parameters (e.g. BERT (Devlin et al., 2018) and GPT (Brown et al., 2020)), which hinders the inference scalability. Recent innovation on hardware architecture suggests structured sparsity is a promising way of alleviating this issue by deploying N:M masks during inference (N out of consecutive M elements in the the weight tensor are kept while others are pruned). N:M masks accelerate model inference with regular sparse structures (Pool, 2020; Fang et al., 2022). Compared to traditional unstructured sparsity (Frankle and Carbin, 2018; Lee et al., 2018; Evci et al., 2020) or channel/block structured sparsity algorithms (Wen et al., 2016; Li et al., 2016; He et al., 2017), adopting N:M masks has negligible evaluation degradation and progressively co-design algorithm (sparse matrix multiplication) and hardware (e.g. Nvidia Ampere Sparse Tensor Core), reaching a desirable trade-off.

Following this line of research, recent studies indicate it is critical (and also possible) to learn these N:M masks from scratch, without additional training or fine-tuning steps. Representative methods in this domain include SR-STE (Zhou et al., 2021), DominoSearch (Sun et al., 2021) and Decaying Mask (Kao et al., 2022), which sparsify the model weights during each forward pass in training to compute gradients, and update them to models. While these methods demonstrate promising results with momentum SGD, their performance over adaptive optimizers, such as Adam, is less satisfactory (Section 3). This implies the benefits of sparsity are largely traded-off by adaptivity in training, leading to slow convergence on many state-of-the-art models (Zhang et al., 2020). In light of this, in this paper we answer the question:

*Can we learn N:M structured sparsity masks with Adam, without model degradation?*

Motivated by the insights from recent studies on critical learning regime of Adam in a distributed learning environment (Tang et al., 2021; Lu et al., 2022), we first hypothesize that with masked weights, the back propagation leads to noisy gradients and gives a poorly estimated variance (running average of second moment gradients) in the

[1]Department of Computer Science, Cornell University [2]Google [3]Google DeepMind. Correspondence to: Yucheng Lu <yl2967@cornell.edu>.

*Proceedings of the $40^{th}$ International Conference on Machine Learning*, Honolulu, Hawaii, USA. PMLR 202, 2023. 

Adam states. It essentially breaks the proper scaling of the coordinate-wise learning rate.

To alleviate this, we propose **STEP** that learns N:M masks with two phases: 1) in the first phase, no mask is applied and **STEP** explores the gradient space to obtain a reliable variance estimate (*precondition phase*); 2) in the second phase, such estimate remains fixed and is used to learn N:M masks (*mask-learning phase*). While previous works have had similar ideas on two-phase training paradigm under the context of low-precision training (Tang et al., 2020; 2021; Lu et al., 2022), the switching point of two phases is still decided by heuristics or redundant hyperparameter tuning. In contrast, **STEP** leverages a novel **AutoSwitch** subroutine that samples the variance update along the training trajectory and tests their concentration.

Our contributions in this paper can be summarized as follows:

- We introduce **STEP**, a recipe for learning N:M structured sparsity masks from scratch with Adam. **STEP** addresses the accuracy drop of state-of-the-art recipes (e.g. SR-STE) with Adam. **STEP** involves a novel subroutine named **AutoSwitch**, which automatically separates the training into precondition and mask learning phases by dynamically testing variance concentration.
- We provide in-depth analysis on why using preconditioning in Adam is justifiable, and prove in theory that under the same conditions given in original Adam paper (Kingma and Ba, 2014), the precondition error from **STEP** remains bounded and the averaged accumulated approximation error is decreasing over time.
- We perform extensive experiments on CIFAR image classification, WMT machine translation, fine-tuning BERT on GLUE and GPT-2 on WikiText-2/-103 that **STEP** mitigates the accuracy drop of baseline algorithms, and is robust to aggressive structured sparsity ratios.

## 2. Related Work

**Recipes for Learning N:M Structured Sparsity Masks from Scratch.** With the proposition of Sparse Tensor Cores introduced in the NVIDIA Ampere GPU architecture (Mishra et al., 2021), there has been an increasing interest of learning N:M structured sparsity masks from scratch. Zhou et al. (2021) initiatively proposes SR-STE that leverages sparse refinement when evaluating gradients via masked weights (termed Straight Through Estimator). Subsequently, Sun et al. (2021) and Kao et al. (2022) extend SR-STE towards using adaptive N:M ratios across layers and steps. While these works focus on learning the N:M masks from scratch, other works have separate discussions. For instance, Holmes et al. (2021) proposes a general framework to learn the structured sparsity mask on a pre-trained model specifically. Hubara et al. (2021) aims to find N:M masks to speed up training rather than inference. Pool and Yu (2021) advocates a pre-permutation yields better results for N:M sparsity and Chmiel et al. (2022) discusses the structured sparsity on activations.

**Critical Learning Regime for Adam Variance.** The existence of a critical learning regime during neural network training has been observed by various studies (Achille et al., 2018; Frankle and Carbin, 2018; Gur-Ari et al., 2018). Many prior works including (Jastrzębski et al., 2018; Jastrzebski et al., 2020) highlight that the early phase of training with SGD determines the difficulty of entire training. Lately, studies including (Agarwal et al., 2021; Tang et al., 2020; 2021) suggest the critical learning regime also exists for Adam-type optimizers (Kingma and Ba, 2014) in a distributed learning environment. More specifically, it has been pointed out that if quantized communication is used in distributed Adam, then we must run dense Adam for the first few iterations to obtain a reliable variance, followed by iterations where quantization is actually applied (Tang et al., 2020; 2021; Li et al., 2021; Lu et al., 2022). Despite the similarity in heuristics to our works, accurately identifying the critical learning regime (i.e. precondition phase) is much more crucial in learning N:M masks: early exiting the precondition phase could lead to unreliable variance estimate while late exit could result in poorly-trained N:M masks. This makes the previous methods on hand-picking the phase length for preconditioning highly unreliable.

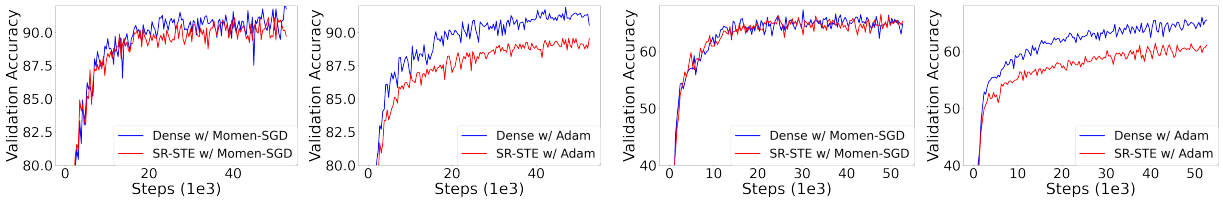
## 3. Preliminary

In this section, we give a full description on the problem formulation. We first provide an overview on the Adam updates and fundamentals to learn N:M masks from scratch with Straight Through Estimator (STE). We also introduce our main baseline SR-STE (Zhou et al., 2021), the state-of-the-art recipe to learn N:M masks. We conclude this section by showing naively applying SR-STE over Adam incurs non-trivial accuracy drop when training ResNet18 on CIFAR10 (He et al., 2016) and DenseNet121 on CIFAR100 (Huang et al., 2017).

**Overview of Adam Updates.** Model training in general can be formulated as an optimization problem, i.e., finding a set of target model weights $\boldsymbol{w}^* \in \mathbb{R}^d$ that minimizes the loss function:

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}\in\mathbb{R}^d} [f(\boldsymbol{w}) = \mathbb{E}_{\zeta\sim\mathcal{D}} f(\boldsymbol{w};\zeta)]. \tag{1}$$

where $\mathcal{D}$ denotes the training set and $f(\boldsymbol{w};\zeta)$ is the loss incurred over sample $\zeta$ given $d$-dimensional model parameters $\boldsymbol{w}$. The Adam optimizer (Kingma and Ba, 2014) solves this problem iteratively with an adaptive learning rate schedule. Concretely, with some initialized value $\boldsymbol{w}_1$, for any $t \geq 1$,




(a) ResNet18 on CIFAR10 (b) DenseNet121 on CIFAR100

Figure 1: Figures demonstrating the state-of-the-art N:M masks learning recipe SR-STE (Zhou et al., 2021) works with momentum SGD but fails to reach target accuracy when trained with Adam on CIFAR classification tasks. In this demonstration, 1:4 (N=1, M=4) sparsity is applied on all the model weights using the exact implementation from (Zhou et al., 2021). Note that here we are not comparing the performance between momentum SGD and Adam, but rather focus on the accuracy gap between dense and SR-STE under two different optimizers.

the update formula of Adam[1] can be summarized as:

$$\text{(Sample Gradient) } \boldsymbol{g}_t = \nabla f(\boldsymbol{w}_t;\zeta_t), \quad \zeta_t \sim \mathcal{D}, \tag{2}$$

$$\text{(Update } \boldsymbol{m}\text{) } \boldsymbol{m}_{t+1} = \beta_1 \boldsymbol{m}_t + (1-\beta_1)\boldsymbol{g}_t, \tag{3}$$

$$\text{(Update } \boldsymbol{v}\text{) } \boldsymbol{v}_{t+1} = \beta_2 \boldsymbol{v}_t + (1-\beta_2)(\boldsymbol{g}_t)^2, \tag{4}$$

$$\text{(Correct Bias) } \hat{\boldsymbol{m}}_{t+1} = \frac{\boldsymbol{m}_{t+1}}{1-\beta_1^t}, \tag{5}$$

$$\text{(Correct Bias) } \hat{\boldsymbol{v}}_{t+1} = \frac{\boldsymbol{v}_{t+1}}{1-\beta_2^t}, \tag{6}$$

$$\text{(Update Model) } \boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \underbrace{\frac{\gamma_t}{\sqrt{\hat{\boldsymbol{v}}_{t+1}+\epsilon}}}_{\text{adaptive learning rate}} \odot \hat{\boldsymbol{m}}_{t+1}, \tag{7}$$

where $\gamma_t$ is the learning rate at step $t$, $\epsilon$ is a small constant to prevent zero division, $\beta_1$ and $\beta_2$ are tunable decaying factors. The running average of first and second gradient moments $\boldsymbol{m}$ and $\boldsymbol{v}$ are usually referred to as *momentum* and *variance*, respectively. The Adam optimizer (and its variants) has been adopted as the folklore method to train many models since its proposition. In recent studies like (Zhang et al., 2020), it has been found that Adam is critical for many attention-based foundation models to achieve state-of-the-art model quality.

**Overview of SR-STE.** Learning N:M structured sparsity masks from scratch refers to generating a set of N:M masks at the end of model training, without any additional training steps, and applying these masks during inference. STE (Bengio et al., 2013) is a basic method to solve this problem by directly masking the model weights during forward passes, making the gradients mask-aware. This can be formally expressed as: $\forall t \geq 1$

$$\boldsymbol{g}_t = \nabla f(\Pi_t \odot \boldsymbol{w}_t;\zeta_t), \tag{8}$$

where $\Pi_t \in \mathbb{R}^d$ is an N:M mask obtained based on the magnitude of $\boldsymbol{w}_t$. Comparing Equation (2) and Equation (8), the main difference in STE is that the gradient is now computed on the masked weights, while the mask is $\boldsymbol{w}_t$ specific at any training step $t$.

Based on STE, SR-STE (Zhou et al., 2021) advocates a regularized version of gradients with masking. Specifically, with a given regularizing coefficient $\lambda$, SR-STE estimates the gradient as:

$$\boldsymbol{g}_t = \nabla f(\Pi_t \odot \boldsymbol{w}_t;\zeta_t) + \lambda(\mathbf{1}-\Pi_t)\odot \boldsymbol{w}_t, \tag{9}$$

where $\mathbf{1}$ denotes all-one vector in $\mathbb{R}^d$. It has been shown in (Zhou et al., 2021) that proper refinement and a well-tuned $\lambda$ mitigates the accuracy drop of momentum SGD over plain STE.

**Issue on SR-STE with Adam.** While the majority of results shown in (Zhou et al., 2021) demonstrates the effectiveness of SR-STE over momentum SGD, here we identify even on simple CIFAR tasks, SR-STE could lead to unsatisfactory sparse models when trained with Adam. We plot the results in Figure 1, which compares the performance of dense training and SR-STE on two models (ResNet18 and DenseNet121) on CIFAR10/100 datasets. We observe that when training a model with Adam, the masks learned by SR-STE incur non-trivial accuracy drop during model inference.

## 4. STEP: STE with Precondition

We proceed to introduce the approach of addressing the aforementioned issue of SR-STE with Adam. The intuition of our method is based on the observation on variance change during model training. We justify our approach with theory under the same condition in (Kingma and Ba, 2014), and illustrate its practicality.

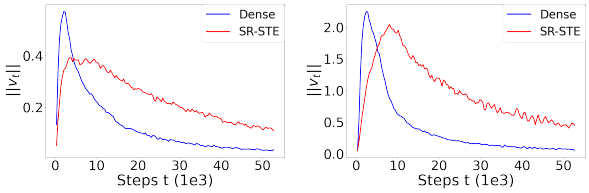
**A Closer Look at Variance Change.** Motivated by the recent studies on distributed Adam (Li et al., 2021; Tang et al., 2021; Lu et al., 2022), we take a closer look at the variance change in the previous tasks and plot them in Figure 2. We observe that while in both dense training and SR-STE, the variance norm first increases and then decreases, the norm in

[1]Note that in Adam, operations like division should act element-wise.

**Algorithm 1** Proposed **STEP** Algorithm

**Require:** Initial time step $t = 0$, initialized model weights $\boldsymbol{w}_0$, Adam-related hyperparameters: $\{(\beta_1,\beta_2), \epsilon$ for preventing zero division, initialized momentum and variance $\boldsymbol{m}_0 = \boldsymbol{0}, \boldsymbol{v}_0 = \boldsymbol{0}\}$.

**while** True **do**
  Sample the data batch $\zeta_t$.
  Compute stochastic gradient $\boldsymbol{g}_t = \nabla f(\boldsymbol{w}_t;\zeta_t)$.
  Update the momentum: $\boldsymbol{m}_{t+1} = \beta_1 \boldsymbol{m}_t + (1-\beta_1)\boldsymbol{g}_t$.
  Update the variance: $\boldsymbol{v}_{t+1} = \beta_2 \boldsymbol{v}_t + (1-\beta_2)(\boldsymbol{g}_t)^2$.
  Correct momentum bias: $\hat{\boldsymbol{m}}_{t+1} = \boldsymbol{m}_{t+1}/(1-\beta_1^t)$.
  Correct variance bias: $\hat{\boldsymbol{v}}_{t+1} = \boldsymbol{v}_{t+1}/(1-\beta_2^t)$.
  Update the weights: $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \gamma_t \hat{\boldsymbol{m}}_{t+1}/\sqrt{\hat{\boldsymbol{v}}_{t+1}+\epsilon}$.
  Update the time $t = t+1$.
  **if** $t$ is the switching point **then**
    Set the preconditioned variance $\boldsymbol{v}^* = \boldsymbol{v}_t$ and **break**.
  **end if**
**end while**
**while** $t < T$ **do**
  Sample the data batch $\zeta_t$.
  Compute N:M mask $\Pi_t$ based on the current weights $\boldsymbol{w}_t$.
  Compute stochastic gradient $\boldsymbol{g}_t = \nabla f(\Pi_t \odot \boldsymbol{w}_t;\zeta_t)$.
  Update the momentum: $\boldsymbol{m}_{t+1} = \beta_1 \boldsymbol{m}_t + (1-\beta_1)\boldsymbol{g}_t$.
  Correct momentum bias: $\hat{\boldsymbol{m}}_{t+1} = \boldsymbol{m}_{t+1}/(1-\beta_1^t)$.
  Update the weights: $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \gamma_t \hat{\boldsymbol{m}}_{t+1}/\sqrt{\boldsymbol{v}^*+\epsilon}$.
  Update the time $t = t+1$.
**end while**
Compute N:M mask $\Pi_T$ based on the current weights $\boldsymbol{w}_T$.
**return** $\Pi_T \odot \boldsymbol{w}_T$ for inference.




(a) ResNet18 on CIFAR10 (b) DenseNet121 on CIFAR100

Figure 2: Figure showing variance $\boldsymbol{v}_t$ (running average of second moment) change in the Adam states, in the CIFAR tasks shown in Figure 1. In dense training, the variance gradually becomes small in magnitude, which suggests the model converges. In contrast, in SR-STE, the variance norm remains large, which suggests the gradients are noisy even in later stage of the training, and thus it scales down the adaptive learning rates.

SR-STE remains large at later stage of learning. This implies the noise obtained in the gradients remains large and essentially scales down the learning rate (Kingma and Ba, 2014).

This motivates us to think extensively on the previous success in distributed learning: can we first run dense Adam to obtain a reliable variance, and then learn the N:M masks over the preconditioned variance? While this is mainly based on heuristics in previous works, we next illustrate it is well-justified in theory.

**Theoretical Motivation.** To motivate preconditioned variance, we start from the original objective of having a variance scaler on the learning rate. In the original Adam paper (Kingma and Ba, 2014), it is shown that $\boldsymbol{v}_t$ is advocated to capture the expectation of the gradient magnitude at step $t$. In fact, Kingma and Ba (2014) provably shows that if the gradient square $\boldsymbol{g}_t^2$ is stationary, i.e. $\mathbb{E}[\boldsymbol{g}_i^2] = \mathbb{E}[\boldsymbol{g}_j^2]$ for any $i$ and $j$, then $\mathbb{E}[\hat{\boldsymbol{v}}_t] = \mathbb{E}[\boldsymbol{g}_t^2]$ so that $\hat{\boldsymbol{v}}_t$ can be used as an estimator for $\boldsymbol{g}_t^2$. Following this intuition, we next prove that under the same condition, the averaged approximation error of leveraging a preconditioned variance estimate is decreasing over time.

**Theorem 1.** *Suppose $\boldsymbol{g}_t^2$ is stationary and has bounded norm $\|\boldsymbol{g}_t^2\|_\infty \le G$ for some constant $G > 0$. Given a sufficient precondition step $t_0$ such that $t_0 > \log_{\beta_2}\left(1 - \frac{1}{\sqrt{2}}\right)$, then for any step $t > t_0$ it holds with probability at least $1-\delta$,*

$$\|\hat{\boldsymbol{v}}_t - \hat{\boldsymbol{v}}_{t_0}\|_\infty < \sqrt{4G^2(1-\beta_2)^2(t-t_0)\log\left(\frac{2}{\delta}\right)}.$$

Theorem 1 provides the worst-case accumulated error of using preconditioned $\boldsymbol{v}_{t_0}$ to estimate $\boldsymbol{v}_t$ ($\forall t > t_0$). Observing the bound given in Theorem 1, conditioned on $t_0$, the maximal accumulated change to a variance coordinate is sublinear to time $t - t_0$. This suggests when we use $\boldsymbol{v}_{t_0}$ to estimate $\boldsymbol{v}_t$ for any $t > 0$, the average error obtained in each step is decreasing over time with rate $O(1/\sqrt{t-t_0})$.

On the other hand, the coefficient $(1-\beta_2)^2$ is a very small number both theoretically and empirically. In theory, it is provably shown that to ensure Adam convergence, $1-\beta_2$ has to be small enough such that $1-\beta_2 = O(N^{-3})$, where $N$ is the size of the training dataset (Zhang et al., 2022), and having a larger $1-\beta_2$ could lead to divergence. In practice $\beta_2$ is often set to a value such that $(1-\beta_2)^2$ reduces $t - t_0$ by orders of magnitude: For instance, the default setting of $\beta_2$ is 0.999 given in the original Adam paper (Kingma and Ba, 2014) and most of the deep learning libraries (Paszke et al., 2019; Heek et al., 2020), leading to $(1-\beta_2)^2 = 10^{-6}$; on foundation models like GPT-3 and Megatron, $(1-\beta_2)^2$ is around $10^{-4}$ (Brown et al., 2020; Smith et al., 2022).

Building upon this, the overall structure of **STEP** algorithm is shown in Algorithm 1 that separates the training into two phases. In the first phase (the first while loop), the normal Adam is used and the variance estimate is actively updated; in the second phase (the second while loop), the variance estimate obtained from phase I is then used as a precondition to learn the mask with Straight Through Estimator (STE).

## 5. Auto Switch Between two Phases

In the previous section, we've discussed the theoretical motivation of using preconditioned variance on learning N:M masks with Adam. However, the central question is still left

**Algorithm 2** Proposed **AutoSwitch** subroutine for **STEP**

**Require:** Sample size $T_w = \lfloor (1-\beta_2)^{-1} \rfloor$ given by **STEP**, the current step $t$, (Optional: lower bound $T_{\min}$ and upper bound $T_{\max}$ for clipping).

1: Compute the current sample on the variance change:

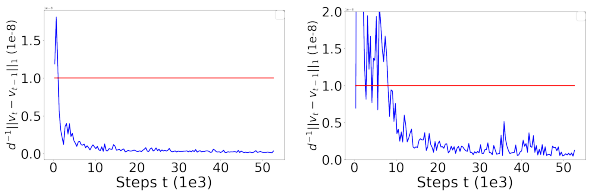
$$\textbf{Option I: } Z_t = d^{-1} \|\boldsymbol{v}_t - \boldsymbol{v}_{t-1}\|_1;$$

$$\textbf{Option II: } Z_t = \exp(d^{-1} \|\log(\boldsymbol{v}_t - \boldsymbol{v}_{t-1})\|_1).$$

2: Estimate mean over the sliding window:

$$\bar{Z} = T_w^{-1} \sum_{j=t-T_w+1}^{t} Z_j.$$

3: **if** (Optional) Use Clipping **then**
4: **return** $t > T_{\max}$ **or** $\bar{Z} < \epsilon$ **and** $t > T_{\min}$.
5: **else**
6: **return** $\bar{Z} < \epsilon$.
7: **end if**




(a) ResNet18 on CIFAR10 (b) DenseNet121 on CIFAR100

Figure 3: Figure showing per-coordinated variance difference $d^{-1}\|\boldsymbol{v}_t - \boldsymbol{v}_{t-1}\|_1$ over steps (in blue curves), in the CIFAR tasks shown in Figure 1. We also plot the $\epsilon$ (in the red line). We observe the update to each coordinate of the variance is quickly dominated by the $\epsilon$.

open: how should we set the switching point $t_0$ in Theorem 1? As partially discussed in Section 1, while identifying reliable Adam variance during training is an established problem, most of the existing methods solve this via heuristics or hyperparameter tuning (Tang et al., 2021; Li et al., 2021; Lu et al., 2022). In this section, we introduce **AutoSwitch**, a subroutine that automatically decides the switching point between precondition and mask learning phases by testing the variance change concentration along the training trajectory.

**Baseline Methods and Their Limitations.** We start with the methods in the literature on identifying the switching point. A straightforward way to do this is leveraging standard hyperparameter tuning protocol such as grid search or random search (Bergstra and Bengio, 2012): setting a few candidate steps and iterate over them and choose the one yielding best performance. However, adding hyperparameters heavily relies on heuristics and requires certain domain knowledge for practitioners.

There have been a few efforts on identifying a good switching point by monitoring the variance metrics. The first is to monitor the relative error as proposed in (Agarwal et al., 2021), which identifies step $t$ as the end of the critical regime if:

$$\frac{\big|\|\boldsymbol{v}_t\| - \|\boldsymbol{v}_{t-1}\|\big|}{\|\boldsymbol{v}_{t-1}\|} < 0.5, \tag{10}$$

where the bound $0.5$ is given by (Agarwal et al., 2021). The intuition is to use the tensor norm difference to approximate the tensor difference (note that storing $\boldsymbol{v}_t$ and $\boldsymbol{v}_{t-1}$ directly could incur non-trival memory overhead due to the high-dimensionality). Another similar method is proposed in (Tang et al., 2021), which suggests a staleness comparison on the variance norm. Concretely, Tang et al. (2021) identifies step $t$ as the end of the critical regime if:

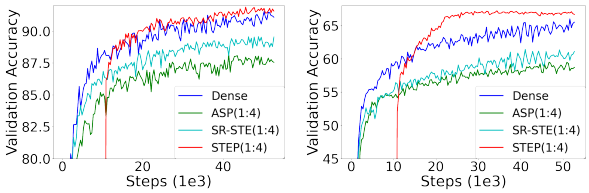
$$\frac{\|\boldsymbol{v}_t\|_1}{\big\|\boldsymbol{v}_{t-\lfloor (1-\beta_2)^{-1} \rfloor}\big\|_1} > 0.96, \tag{11}$$

where the criteria $0.96$ is provided by (Tang et al., 2021).

The baseline methods (Equation (10) and (11)) are limited in practice in three-fold: (i) when evaluating the switching point $t$, it can be easily affected by the noise at step $t$; (ii) Although both of the methods require relative metrics, the thresholds are still hand-picked, and thus introducing additional noise to the criterion; (iii) Both of the methods use the tensor norm over all the coordinates. On one hand, norm can be a good indicator for status of variance but not for variance changes. On the other hand, the switching point can easily be mistakenly missed due to the outliers among the coordinates, especially on large models, where the order of variance magnitude varies significantly (Xiong et al., 2020; Liu et al., 2020).

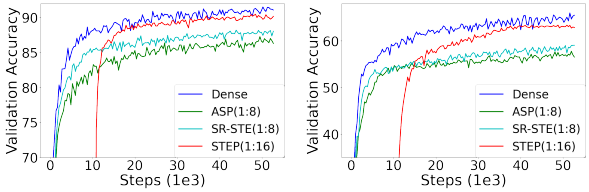
**AutoSwitch.** The main procedures of **AutoSwitch** are summarized in Algorithm 2. To cope with the gradient noise and outlier coordinates, **AutoSwitch** samples over time $t$ the per-coordinate variance change via arithmetic mean (**Option I**) or geometric mean (**Option II**). While geometric mean is robust to outliers, in practice we found arithmetic mean is sufficient for deciding the switching point. We set the sampling window length to be $\lfloor (1-\beta_2)^{-1} \rfloor$. This quantity is motivated from the Markov Chain theory: the mixing time of the Markov Chain defined over $\boldsymbol{v}_t$ is asymptotically $\tilde{O}\Big(\frac{1}{1-\beta_2}\Big)$.

While sampling mitigates the noise from single step evaluation, it still remains unclear what metric we should be applying to decide the phase length. Note that in the baseline works (Equation (10) and (11)), hand-picking values are applied. Ideally, we should leverage some metrics from the Adam optimizer that is adapted to each task. Based on this, **AutoSwitch** uses the $\epsilon$ from Adam as the signal. The $\epsilon$ is originally used in Adam to prevent zero division. In some research it has been found that it largely decides the model convergence (Yuan and Gao, 2020). To justify our motivation, we plot the per-coordinate variance change and $\epsilon$ in Figure 3. We observe the update to each coordinate of the variance is quickly dominated by the $\epsilon$ as the training proceeds.




(a) ResNet18 on CIFAR10 (b) DenseNet121 on CIFAR100

Figure 4: Figure showing how **STEP** mitigates the gap of baseline algorithm ASP (Mishra et al., 2021) and SR-STE (Zhou et al., 2021). In this experiment, 1:4 sparsity is used. The switching point of **STEP** is decided by the **AutoSwitch** subroutine. Note that during the precondition phase of **STEP**, the model does not involve the mask learning while the model is evaluated with sparsity (for fair comparison to baseline models). And thus the evaluation accuracy during that phase is low compared to the mask learning phase.




(a) ResNet18 on CIFAR10 (b) DenseNet121 on CIFAR100

Figure 5: Figure comparing the performance of **STEP** under aggressive sparsity ratio. Comparing the results with Figure 4, it suggests the **STEP** recipe is robust to aggressive sparsity ratio up to 1:16, while baselines degrade the evaluation accuracy at 1:8.

**Clipping for Tight Training Budget.** While Algorithm 2 provides a statistical way of identifying the switching point, in practice, varying training budgets (e.g. model fine-tuning) are usually considered. We can use clipping to clamp a computed switching point $t_0$ between given $T_{\min}$ and $T_{\max}$. The clipping bounds are two optional variables that regularize the **AutoSwitch** subroutine. By default, we suggest using $T_{\min} = 0.1T$ and $T_{\max} = 0.5T$, these two values are motivated by Geweke's convergence diagnostic in MCMC theory (Geweke et al., 1991). Recall that the update of $\boldsymbol{v}_t$ forms a markov chain, and so in practice, the concentration of the first 10% and last 50% of the chain can be used as a good indicator on the convergence (Geweke et al., 1991).

## 6. Experiment

In this section we evaluate the effectiveness of proposed **STEP** and **AutoSwitch** on various tasks, comparing it to other baseline recipes of learning N:M masks. We also show that **STEP** can be easily extended to incorporate other techniques such as layer-wise sparsity (Sun et al., 2021). All of the experiments run on a Google Cloud TPUv3-8 virtual machine.

Table 1: Comparing **AutoSwitch** (Algorithm 2) with two baseline approaches Equation (10) (Agarwal et al., 2021) and (11) (Tang et al., 2021). We measure the average change within 1k steps after the precondition $t_0$ identified by different approaches: $10^{-3}\sum_{t=t_0}^{t_0+1000}\|\boldsymbol{v}_{t+1}-\boldsymbol{v}_t\|_1$. A lower number indicates better estimation for the switching points. The numbers for each experiment are averaged over 5 different random seeds.

| Task | Eq. (10) | Eq. (11) | **AS** |
|---|---|---|---|
| ResNet18/CF10 | 1.58e-1 | 5.58e-2 | **0.79e-2** |
| DenseNet121/CF100 | 5.26e-1 | 1.28e-2 | **0.46e-2** |
| BERT-Large (PreT) | 4.92e-6 | 2.71e-7 | **2.28e-7** |

**Overview of Tasks.** Throughout these sections, we adopt the following tasks for the evaluation: (1) Training various vision models (ResNet18, Densenet121) on CIFAR10/100 dataset (Krizhevsky et al., 2009). (2) Finetuning BERT-Base(Devlin et al., 2018) on the GLUE benchmark (Wang et al., 2018). (3) Training a 6-layer Transformer model on the WMT17 De-En Translation task following (Vaswani et al., 2017). (4) Finetuning GPT-2 model (Radford et al., 2019) on Wikitext-2 and Wikitext-103 (Merity et al., 2016).

**Hyperparameters.** We apply the grid search over the following hyperparameters on each task. Notice that we only tune the hyperparameters for the baselines, but not for **STEP**. That is, **STEP** reuses the hyperparameters tuned for SR-STE. This suggests **STEP** can provide in-place improvement over the baseline recipes. For all the Adam-specific hyperparameters we adopt the default values: $\{\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1e-8\}$. For the CIFAR tasks, we adopted batch size 128 and tune the learning rate from $\{1e-4, 5e-5, 1e-5\}$; for BERT and GPT-2 fine-tuning we follow (Tang et al., 2021) and tune batch size from $\{8,16,32\}$ and learning rate from $\{1e-4,5e-5,1e-5\}$; for WMT machine translation we follow the exact setup[2] of (Vaswani et al., 2017) and (Kao et al., 2022).

**The Effectiveness of AutoSwitch.** We start from evaluating the effectiveness of **AutoSwitch** over baseline methods as introduced in Section 5. Concretely, we compare Algorithm 2 with Equation (10) proposed by (Agarwal et al., 2021) and Equation (11) proposed by (Tang et al., 2021). For each task, we first profile the $\|\boldsymbol{v}_t\|_2$, $\|\boldsymbol{v}_t\|_1$ and $\|\boldsymbol{v}_{t+1}-\boldsymbol{v}_t\|_1$ for all the $t \geq 1$ since these suffice for running the three approaches. Then for any $t_0$ as a precondition step found by each method, we compute the average variance change in the

[2] A more detailed description can be found in Section 4 (Kao et al., 2022).

Table 2: Finetuning BERT-Base on the GLUE development set. The original results are from (Devlin et al., 2018). The Dense results are reproduced by ours with no sparsity. For different recipes (ASP, SR-STE and **STEP**), 2:4 sparsity is applied on all the linear modules (including attention, intermediate and output layer of BERT.) The scores are the median scores over 10 runs with different seeds. We observe compared to baselines, **STEP** has a negligible drop on the average score compared to the dense counterpart.

| | RTE | MRPC | STS-B | CoLA | SST-2 | QNLI | QQP | MNLI-m | MNLI-mm | Avg Score |
|---|---|---|---|---|---|---|---|---|---|---|
| Original | 66.4 | 84.8 | 85.8 | 52.1 | 93.5 | 90.5 | 89.2 | 84.6 | 83.4 | 81.1 |
| Dense | 65.0 | 85.1 | 85.2 | 51.0 | 92.3 | 91.1 | 91.0 | 84.6 | 83.6 | 81.0 |
| ASP | 57.4 | 79.2 | 81.7 | 47.2 | 88.5 | 83.7 | 84.8 | 80.6 | 79.5 | 75.8 |
| SR-STE | 55.6 | 81.3 | 88.2 | 47.8 | 90.2 | 86.6 | 90.1 | 82.1 | 82.9 | 78.3 |
| **STEP** | 62.4 | 84.7 | 88.7 | 50.4 | 91.8 | 89.2 | 90.9 | 84.2 | 83.9 | **80.7** |

Table 3: Training different language modeling tasks on Wikitext-2(-103). For different recipes (ASP, SR-STE and **STEP**), 2:4 sparsity is applied on all the `Conv1D` modules of GPT2. The numbers are averaged evaluation perplexity over 10 runs with different seeds.

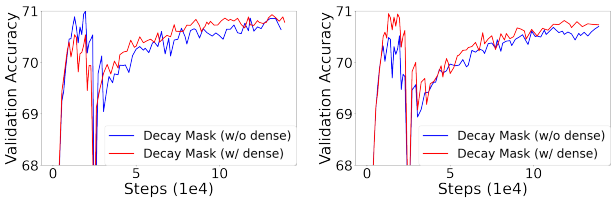
| | Wikitext-2 | Wikitext-103 |
|---|---|---|
| Dense | 21.15 | 16.57 |
| ASP | 37.09 | 26.29 |
| SR-STE | 28.54 | 18.93 |
| **STEP** | **23.85** | **17.02** |

Table 4: Extension of **STEP** to layer-wise N:M masks learning. The N:M sparsity ratios are decided in a per-layer fashion following the strategy given in (Sun et al., 2021). The numbers in this table are averaged over 5 runs. The results suggest **STEP** can provide in-place improvement when combined with per-layer structured sparsity.

| | N:M | RN-CF10 | DN-CF100 |
|---|---|---|---|
| Dense | / | 91.56 | 65.62 |
| DS | Mixed N:8 | 89.94 | 64.88 |
| DS+**STEP** | Mixed N:8 | **91.42** | **65.71** |
| DS | Mixed N:16 | 87.08 | 62.13 |
| DS+**STEP** | Mixed N:16 | **90.93** | **65.04** |
| DS | Mixed N:32 | 85.37 | 60.47 |
| DS+**STEP** | Mixed N:32 | **90.12** | **64.91** |

next 1k steps, i.e., $10^{-3}\sum_{t=t_0}^{t_0+1000}\|\boldsymbol{v}_{t+1}-\boldsymbol{v}_t\|_1$ as measuring the reliability of preconditioned variance. Intuitively, a smaller average variance change implies better preconditioning. We summarize the results in Table 1, the results suggest **AutoSwitch** is able to identify variance with subtle changes in the following steps compared to the other two baselines.

**Comparing with Baselines.** We now evaluate the performance of **STEP** with the following baseline recipes: Dense (no mask is learnt), ASP (Mishra et al., 2021) and SR-STE (Zhou et al., 2021). The comparison is carried out on three tasks: training ResNet18 and Densenet121 from scratch on CIFAR10/100; finetuning BERT-Base on GLUE; and finetuning GPT2 on Wikitext-2/-103. For all the recipes, we apply 2:4 sparsity (Pool, 2020) to all the modules. More concretely: for ResNet and DenseNet, the sparsity is applied on all the `Conv2D` layers; for BERT-Base, all the `Linear` modules in attention, intermediate and output layers are sparsified; in GPT-2, the sparsity is applied on all the `Conv1D` modules. We summarize the results in Figure 4, Table 2 and 3. The results consistently suggest under the same sparsity ratio, **STEP** is able to mitigate the accuracy drop between baseline recipes (ASP and SR-STE) and dense training. Perhaps surprisingly, we found in the DenseNet task, **STEP** achieves higher validation accuracy compared to the dense training.

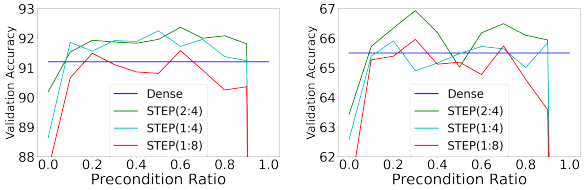



(a) N:M=1:32 (b) N:M=1:64

Figure 6: Ablation Study on Decaying Mask. We follow the setting of (Kao et al., 2022) and train the 6-layer Transformer model on the WMT17 De-En translation task. To shows the importance of preconditioning with dense updates. We include the results and compare the Decaying Mask recipe with and without the dense training phase.

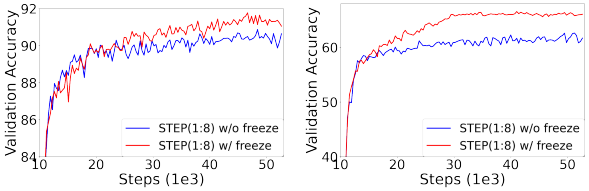
**Robustness to Aggressive Structured Pruning.** We extend the previous experiments on pre-training ResNet18 and DenseNet121 with different sparsity ratios, using **STEP** recipes. We summarize the results in Figure 4, we observe up to N:M=1:16, **STEP** recipe has negligible accuracy drop compared to the dense training, while other recipes have non-trivial evaluation accuracy gap at 1:8.

**Ablation Study I: Layer-wise Pruning.** We now demonstrate that **STEP** can be trivially extended to layer-wise SR-STE as considered in DominoSearch (Sun et al., 2021). We




(a) ResNet18 on CIFAR10 (b) DenseNet121 on CIFAR100

Figure 7: Ablation study on different precondition phase length. The X-axis denotes the ratio of precondition phase length over the total number of training steps; while the Y-axis denotes the evaluation accuracy of the output model at the end. We observe that the switching point between precondition and mask learning phase is quite flexible.




(a) ResNet18 on CIFAR10 (b) DenseNet121 on CIFAR100

Figure 8: Ablation study on comparing with and without updating variance term during the mask learning phase. The curves suggest freezing (fixing) the preconditioned variance during the mask learning phase is crucial.

now run the **STEP** and **AutoSwitch** following a per-module fashion, with per-layer sparsity ratio determined by the DominoSearch algorithm (Sun et al., 2021). We summarize the results of using plain DominoSearch (DS) and DS combined with **STEP** in Table 4. The results there suggest combined with **STEP**, DominoSearch can have more stable results, especially over aggressive N:M ratios. More concretely, when the sparsity ratios are increased to N:32, the original DominoSearch already incurs over 5% accuracy drop while with **STEP**, the accuracy drop is generally around 1% on both ResNet and DenseNet. Notice that **STEP** does not modify the dynamic sparsity ratio assignment strategy as used in the original DominoSearch. This, on the other hand, implies **STEP** provides in-place improvement over layer-wise sparsity.

**Ablation Study II: Decaying Mask.** In this experiment, we conduct an ablation study on a recently proposed recipe named Decaying Mask (Kao et al., 2022). The recipe proceeds as follows: first run dense training for some iterations, and then start the sparse training phase. At the beginning of the sparse training phase, it starts with M-1:M structured sparsity. As training progresses, Decaying Mask increases the sparsification degree by applying N:M structured sparsity at different decaying intervals, where $N = \lfloor \frac{M}{2^s} \rfloor$.

Note that the original Decaying Mask recipe already includes the dense training phase. In this ablation study, we follow the setup of (Kao et al., 2022) and compare how Decaying Mask behaves with and without its dense training phase. We summarize the results in Figure 6. It suggests if no dense training is performed at the beginning of the recipe, there will be a certain accuracy drop even if the sparsity ratio is gradually decreased. This, again, substantiates the motivation of **STEP** recipe.

**Ablation Study III: Varying Preconditioning Phase Length.** We continue investigating the effect of preconditioning phase length on the final model accuracy. We repeat the CIFAR experiments on two vision models and rerun the **STEP** algorithm with different precondition phase length. We summarize the results in Figure 7. We observe that **STEP** is able to achieve dense accuracy when the ratio of preconditioning phase is between 10% and 80% (despite the fact that **AutoSwitch** decides the ending point to be around 20%). This suggests the switching point in **STEP** is quite flexible over the entire training trajectory, and is robust to the potential noise in the **AutoSwitch** subroutine.

**Ablation Study IV: Why Fixing the Variance.** Note that in the original **STEP** Algorithm, the variance remains fixed during the masking learning phase. A natural question to this would be: does it help if we keep updating the variance using the gradients computed on the sparsified model? In practice, we observe this in fact has negative impact. We rerun the ResNet/DenseNet experiments with two variants: original **STEP** and **STEP** where variance is updated in the second phase. We summarize the results in Figure 8. It suggests keeping updating the variance with gradients computed on masked weights reduces the final evaluation accuracy, which implies the noise level in gradients remains high during mask learning, even in the later stage of training.

## 7. Conclusion

In this paper, we identify the state-of-the-art recipe SR-STE incurs non-trivial model degradation when applied in Adam-based model training. We propose an algorithm named **STEP** that separates the training into two phases, where in the first phase, the Adam optimizer computes a reliable second moment (variance) estimate; while in the second phase, such variance remains fixed and is used as a precondition to learn the N:M structured sparsity masks. We also propose a subroutine named **AutoSwitch** that automatically determines the switching point of two phases. Compared to other approaches, **AutoSwitch** shows stable and reliable estimation. Empirically we evaluate **STEP** on various benchmarks including text classification, image

classification and language modeling. We demonstrate **STEP** mitigates the accuracy drop compared to other recipes and is robust to aggressive sparsity ratios.

## Acknowledgements

CDS gratefully acknowledges the support from NSF-2046760 CAREER. YL gratefully acknowledges the support of the Meta PhD Fellowship. We extend our gratitude towards James Laudon, Cliff Young, Google Research Cerebra Team and extended Google Research, Brain Team for their feedback and comments. We also would like to thank Google Cerebra Team for hosting this research and proving TPU Research Cloud; and Google University Relations Team for the conference funding. We further would like to thank Blaise Aguera-Arcas, Jihwan Lee and anonymous reviewers from ICML 2023 for feedback on the early draft of this work.

## References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Jeff Pool. Accelerating sparsity in the nvidia ampere architecture. *GTC 2020*, 2020.

Chao Fang, Aojun Zhou, and Zhongfeng Wang. An Algorithm–Hardware Co-Optimized Framework for Accelerating N: M Sparse Transformers. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 30(11): 1573–1586, 2022.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*, 2018.

Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, pages 2943–2952. PMLR, 2020.

Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. *Advances in neural information processing systems*, 29, 2016.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1389–1397, 2017.

Aojun Zhou, Yukun Ma, Junnan Zhu, Jianbo Liu, Zhijie Zhang, Kun Yuan, Wenxiu Sun, and Hongsheng Li. Learning n: m fine-grained structured sparse neural networks from scratch. *arXiv preprint arXiv:2102.04010*, 2021.

Wei Sun, Aojun Zhou, Sander Stuijk, Rob Wijnhoven, Andrew O Nelson, Henk Corporaal, et al. DominoSearch: Find layer-wise fine-grained N: M sparse schemes from dense neural networks. *Advances in Neural Information Processing Systems*, 34:20721–20732, 2021.

Sheng-Chun Kao, Amir Yazdanbakhsh, Suvinay Subramanian, Shivani Agrawal, Utku Evci, and Tushar Krishna. Training Recipe for N: M Structured Sparsity with Decaying Pruning Mask. *arXiv preprint arXiv:2209.07617*, 2022.

Jingzhao Zhang, Sai Praneeth Karimireddy, Andreas Veit, Seungyeon Kim, Sashank Reddi, Sanjiv Kumar, and Suvrit Sra. Why are adaptive methods good for attention models? *Advances in Neural Information Processing Systems*, 33:15383–15393, 2020.

Hanlin Tang, Shaoduo Gan, Ammar Ahmad Awan, Samyam Rajbhandari, Conglong Li, Xiangru Lian, Ji Liu, Ce Zhang, and Yuxiong He. 1-bit adam: Communication efficient large-scale training with adam's convergence speed. In *International Conference on Machine Learning*, pages 10118–10129. PMLR, 2021.

Yucheng Lu, Conglong Li, Minjia Zhang, Christopher De Sa, and Yuxiong He. Maximizing Communication Efficiency for Large-scale Training via 0/1 Adam. *arXiv preprint arXiv:2202.06009*, 2022.

Hanlin Tang, Shaoduo Gan, Samyam Rajbhandari, Xiangru Lian, Ji Liu, Yuxiong He, and Ce Zhang. APMSqueeze: A Communication Efficient Adam-Preconditioned Momentum SGD Algorithm. *arXiv preprint arXiv:2008.11343*, 2020.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Asit Mishra, Jorge Albericio Latorre, Jeff Pool, Darko Stosic, Dusan Stosic, Ganesh Venkatesh, Chong Yu, and Paulius Micikevicius. Accelerating sparse deep neural networks. *arXiv preprint arXiv:2104.08378*, 2021.

Connor Holmes, Minjia Zhang, Yuxiong He, and Bo Wu. NxMTransformer: Semi-Structured Sparsification for Natural Language Understanding via ADMM. *Advances in Neural Information Processing Systems*, 34:1818–1830, 2021.

Itay Hubara, Brian Chmiel, Moshe Island, Ron Banner, Joseph Naor, and Daniel Soudry. Accelerated sparse neural training: A provable and efficient method to find n: m transposable masks. *Advances in Neural Information Processing Systems*, 34:21099–21111, 2021.

Jeff Pool and Chong Yu. Channel permutations for n: m sparsity. *Advances in Neural Information Processing Systems*, 34:13316–13327, 2021.

Brian Chmiel, Itay Hubara, Ron Banner, and Daniel Soudry. Optimal Fine-Grained N: M sparsity for Activations and Neural Gradients. *arXiv preprint arXiv:2203.10991*, 2022.

Alessandro Achille, Matteo Rovere, and Stefano Soatto. Critical learning periods in deep networks. In *International Conference on Learning Representations*, 2018.

Guy Gur-Ari, Daniel A Roberts, and Ethan Dyer. Gradient descent happens in a tiny subspace. *arXiv preprint arXiv:1812.04754*, 2018.

Stanisław Jastrzębski, Zachary Kenton, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. On the relation between the sharpest directions of DNN loss and the SGD step length. *arXiv preprint arXiv:1807.05031*, 2018.

Stanislaw Jastrzebski, Maciej Szymczak, Stanislav Fort, Devansh Arpit, Jacek Tabor, Kyunghyun Cho, and Krzysztof Geras. The break-even point on optimization trajectories of deep neural networks. *arXiv preprint arXiv:2002.09572*, 2020.

Saurabh Agarwal, Hongyi Wang, Kangwook Lee, Shivaram Venkataraman, and Dimitris Papailiopoulos. Adaptive Gradient Communication via Critical Learning Regime Identification. *Proceedings of Machine Learning and Systems*, 3:55–80, 2021.

Conglong Li, Ammar Ahmad Awan, Hanlin Tang, Samyam Rajbhandari, and Yuxiong He. 1-bit LAMB: Communication Efficient Large-Scale Large-Batch Training with LAMB's Convergence Speed. *arXiv preprint arXiv:2104.06069*, 2021.

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

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Yushun Zhang, Congliang Chen, Naichen Shi, Ruoyu Sun, and Zhi-Quan Luo. Adam can converge without any modification on update rules. *arXiv preprint arXiv:2208.09632*, 2022.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. Flax: A neural network library and ecosystem for JAX, 2020. URL http://github.com/google/flax.

Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, et al. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model. *arXiv preprint arXiv:2201.11990*, 2022.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pages 10524–10533. PMLR, 2020.

Liyuan Liu, Xiaodong Liu, Jianfeng Gao, Weizhu Chen, and Jiawei Han. Understanding the difficulty of training transformers. *arXiv preprint arXiv:2004.08249*, 2020.

Wei Yuan and Kai-Xin Gao. Eadam optimizer: How epsilon impact adam. *arXiv preprint arXiv:2011.02150*, 2020.

John F Geweke et al. Evaluating the accuracy of sampling-based approaches to the calculation of posterior moments. Technical report, Federal Reserve Bank of Minneapolis, 1991.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer Sentinel Mixture Models, 2016.

Martin J Wainwright. *High-dimensional statistics: A non-asymptotic viewpoint*, volume 48. Cambridge University Press, 2019.

# A. Technical Proof

## A.1. Proof to Theorem 1

*Proof.* We first define the filtration $\mathcal{F}_t$ over step $t\in\{1,\cdots,T\}$, where the randomness come from the sampling of the data point $\zeta_t$. And next we show the update for each coordinate of $\boldsymbol{v}_t$ is a martingale difference sequence. From the update of Adam, we get:

$$\begin{aligned}\hat{\boldsymbol{v}}_{t+1}-\hat{\boldsymbol{v}}_t&=\frac{\boldsymbol{v}_{t+1}}{1-\beta_2^{t+1}}-\frac{\boldsymbol{v}_t}{1-\beta_2^t}\\&=\frac{1}{1-\beta_2^{t+1}}\left(\boldsymbol{v}_{t+1}-\frac{1-\beta_2^{t+1}}{1-\beta_2^t}\boldsymbol{v}_t\right)\\&=\frac{1}{1-\beta_2^{t+1}}\left[\beta_2\boldsymbol{v}_t+(1-\beta_2)\boldsymbol{g}_t^2-\frac{1-\beta_2^{t+1}}{1-\beta_2^t}\boldsymbol{v}_t\right]\\&=\frac{1}{1-\beta_2^{t+1}}\left[(1-\beta_2)\cdot\left(\boldsymbol{g}_t^2-\frac{\boldsymbol{v}_t}{1-\beta_2^t}\right)\right].\end{aligned}$$

Take expectation with respect to the filtration, we obtain

$$\begin{aligned}\mathbb{E}[\hat{\boldsymbol{v}}_{t+1}-\hat{\boldsymbol{v}}_t|\mathcal{F}_t]&=\mathbb{E}\left[\frac{1-\beta_2}{1-\beta_2^{t+1}}\left(\boldsymbol{g}_t^2-\frac{\boldsymbol{v}_t}{1-\beta_2^t}\right)\Big|\mathcal{F}_t\right]\\&=\frac{1-\beta_2}{1-\beta_2^{t+1}}\mathbb{E}\left[\boldsymbol{g}_t^2-\frac{\boldsymbol{v}_t}{1-\beta_2^t}\Big|\mathcal{F}_t\right].\end{aligned}$$

Note that

$$\mathbb{E}[\boldsymbol{v}_t]=\mathbb{E}\left[(1-\beta_2)\sum_{j=1}^{t}\beta_2^{t-j}\boldsymbol{g}_j^2\right]=(1-\beta_2^t)\mathbb{E}[\boldsymbol{g}_t^2].$$

Push it back, we know for each $i\in[d]$,

$$\mathbb{E}\left[\boldsymbol{e}_i^\top(\hat{\boldsymbol{v}}_{t+1}-\hat{\boldsymbol{v}}_t)|\mathcal{F}_t\right]=0. \tag{12}$$

On the other hand, for each $i\in[d]$,

$$\left|\boldsymbol{e}_i^\top(\hat{\boldsymbol{v}}_{t+1}-\hat{\boldsymbol{v}}_t)\right|=\frac{1-\beta_2}{1-\beta_2^{t+1}}\left|\boldsymbol{e}_i^\top\left(\boldsymbol{g}_t^2-\frac{\boldsymbol{v}_t}{1-\beta_2^t}\right)\right|$$

Note that both $\boldsymbol{e}_i^\top\boldsymbol{g}_t^2$ and $\frac{\boldsymbol{e}_i^\top\boldsymbol{v}_t}{1-\beta_2^t}$ is non-negative. Considering that

$$\frac{\boldsymbol{e}_i^\top\boldsymbol{v}_t}{1-\beta_2^t}=\frac{1-\beta_2}{1-\beta_2^t}\sum_{j=0}^{t}\beta_2^{t-j}\boldsymbol{e}_i^\top\boldsymbol{g}_j^2\le G.$$

And so

$$\left|\boldsymbol{e}_i^\top(\hat{\boldsymbol{v}}_{t+1}-\hat{\boldsymbol{v}}_t)\right|\le\frac{1-\beta_2}{1-\beta_2^{t+1}}G\le\frac{1-\beta_2}{1-\beta_2^{t_0}}G\le\sqrt{2}(1-\beta_2)G, \tag{13}$$

where we apply the fact that $t>t_0$ and $t_0>\frac{\log(1/2)}{\log(\beta_2)}$. Considering Equation (12) and (13), we know it is a martingale difference sequence. Now we apply the Azuma-Hoeffding Inequality (Wainwright, 2019), and get for any $i\in[d]$,

$$\begin{aligned}\mathbb{P}\left[\left|\sum_{k=t_0}^{t-1}\boldsymbol{e}_i^\top(\hat{\boldsymbol{v}}_{k+1}-\hat{\boldsymbol{v}}_k)\ge c\right|\right]&\le 2\exp\left(-\frac{c^2}{2\sum_{k=t_0}^{t-1}\left(\sqrt{2}(1-\beta_2)G\right)^2}\right)\\&=2\exp\left(-\frac{c^2}{4G^2(1-\beta_2)^2(t-t_0)}\right).\end{aligned}$$

Set the R.H.S. as $\delta$, we obtain

$$c=\sqrt{4G^2(1-\beta_2)^2(t-t_0)\log\left(\frac{2}{\delta}\right)}.$$

Finally we get

$$\|\hat{\boldsymbol{v}}_t-\hat{\boldsymbol{v}}_{t_0}\|_\infty<\sqrt{4G^2(1-\beta_2)^2(t-t_0)\log\left(\frac{2}{\delta}\right)},$$

as desired. That completes the proof

□

Table 5: Potential memory savings for different sparsity ratios of two of the large models (BERT-Base, GPT-2). The numbers are assumed with 2-byte per parameter.

| | Dense | 2:4 | 1:4 | 1:16 |
|---|---|---|---|---|
| BERT-Base | 216.75 MB | 131.82 MB (1.65×) | 89.35 MB (2.43×) | 57.50 MB (3.76×) |
| GPT-2 | 247.06 MB | 162.13 MB (1.52×) | 119.66 MB (2.06×) | 87.81 MB (2.81×) |

## B. Memory Saving

We estimate the potential memory savings for different sparsity ratios of two of the large models (BERT-Base, GPT-2) used in the experimental section, and summarize the numbers in Table 5. In all cases, we assume 2-byte per parameter and include the required metadata to represent each sparsity pattern.