We thank the reviewers for their feedback. Our paper will be updated to reflect the responses below.
ALL: *Novelty?* Our contribution are: (1) Demonstrating training is highly sensitive to sparsifying back-propagated output error gradients; (2) A simple training algorithm, sparse in forward and backward passes, with activation and weight sparsity that fits emerging sparse accelerators. *Why SWAT outperforms related work?* We believe this is because SWAT can perform dynamic topology exploration: Backpropagation with sparse weights and activations approximates backpropagation on a network with sparse connectivity and sparsely activated neurons. The gradients generated during back-propagation minimize loss for the current sparse connectivity. However, each iteration SWAT generates a potentially new sparse network using the sparsifying function. Non-active weights are also updated thus capturing fine-grained temporal importance of connectivity during training. Figure 7 and 8 show the importance of unmasked gradient updates and dynamic exploration of connectivity.
Reviewer 1: **(1)** When sparsifying Equation 2 and 3, we expect combinations sparsifying both gradients and weights and/or activations (i.e., (g,a), (g,w), (g,w,a)) would underperform as sparsifying gradients alone harms convergence. Also, as sparsifying weights or activations alone will not sparsify both equations, we skip these combinations too. **(2)** DSG loses considerable accuracy even at low sparsity. E.g., for ResNet18 on ImageNet at 50% sparsity DSG suffers an accuracy loss of 4.6%.
Reviewer 2: **(1)** *"Drastic drop due to sparse activations in forward pass"*: In Figure 1 we isolate the forward and backward pass and examine sensitivity of training to sparsifying only the forward pass. Notably, this means we use the full activation for the backward pass. So, in Figure 1 the backward pass is not optimizing the network for sparse activation. Typical filter pruning methods use sparse activation in the backward pass so the network adapts for activation sparsity. Kurtz et al. (ICML 2020) and Georgiadis (CVPR 2019) show how to increase activation sparsity in the forward pass. Combining such techniques with SWAT may reduce FLOPS without increasing error and thus may be a good direction for future work. **(2)** *"Never faced such drastic drops from sparse output gradients"*: We believe there is a misunderstanding: GMP, STR, CS and RigL use a sparse weight gradient during the parameter update stage whereas in Figure 2 we sparsify the back-propagated error gradient. The back-propagated output error gradient is sparsified before performing the convolutions to generate weight and input gradients. The generated weight gradient is dense because the convolution between a sparse activation and dense back-propagated error gradient tensor yields a dense result. This back-propagated error gradient is not dropped in STR, CS, and GMP but rather dense back-propagated error gradients are used to generate weight gradients that are masked during the parameter update stage. Thus, STR, CS, GMP only update the active parameters. **(3)** Yes, sparsity budget proposed by Kusupati et al. (ICML 2020) could be used for more speedup. **(4)** Yes, the different channels will be selected for different filters and there is overhead for introducing structured sparsity since the L1 response of channels is computed. However, the FLOP reduction due to high sparsity is more than the overhead of computing the L1 response. Moreover, the idea of periodically computing the Top-K channel selection is applicable. **(5)** SWAT doesn't have dense phase. Base model accuracy can be calculated from the accuracy drop. E.g., for ResNet-50 on ImageNet, the Top-1 accuracy is 76.8. **(6)** Yes, in general uniform will have lower flop than ERK especially when the input resolution is high. ERK generally applies lower sparsity at initial layer which have significant computation especially true for imagenet. However, in Table 2, ERK is more efficient because the initial layer has small computation due to small input resolution (CIFAR-10) and computationally expensive layers have higher sparsity. Similarly, we observe DST allocates higher sparsity to more computationally expensive layers and quickly reaches a sparsity where overall computation is low. **(7)** We will add discussion on Li et al. 2017 and follow-up work on structured sparsity.
Reviewer 3: **(1)** *The speedup gap is small:* The theoretical speedup by the earlier method would be around $\frac{1}{1-0.67} = 3.03\times$ whereas with our method the theoretical speedup would be $\frac{1}{1-0.76} = 4.16\times$. Theoretical speedup does not fully capture the benefit of SWAT since SWAT also reduces memory footprint during training. **(2)** *Methodology for cycle count estimation?* We will add relevant detail from the supplementary material. Simulator counts the cycles taken to spatially map and schedule the computation present in each layer. The memory hierarchy is similar to the DaDianNo architecture. *Table 1*: (i) The column represents whether the input gradient and weight gradient computation is sparse or not. (ii) Second, convolution between sparse input activation and dense back-propagated error gradient tensor generates a dense gradient for weight and the dense gradient is used in parameter update. (iii) Yes, related work can be adapted for structured sparsity.
Reviewer 4: Selecting the sparsity budget of SWAT (e.g., using ERK) is not our main contribution, but rather demonstrates SWAT can readily leverage such techniques. We have covered a range of configurations in Table 4, 5 6 and 7 in the main paper and Table 1 and 2 of the supplemental. Joint optimization of backward pass activation and weight sparsity is an good direction for future work.