Train Flat, Then Compress: Sharpness-Aware Minimization Learns More Compressible Models

Clara Na, Sanket Vaibhav Mehta, Emma Strubell
Language Technologies Institute, Carnegie Mellon University

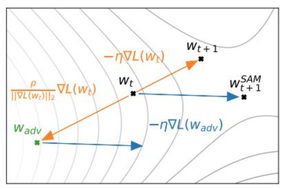
Abstract

Model compression by way of parameter pruning, quantization, or distillation has recently gained popularity as an approach for reducing the computational requirements of modern deep neural network models for NLP. Inspired by previous work suggesting a connection between simpler, more generalizable models and those that lie within flat basins in the loss landscape, we hypothesize that optimizing for flat minima should lead to simpler parameterizations and thus more compressible models. We propose to combine sharpness-aware minimization with various task-specific model compression methods, including iterative magnitude pruning, structured pruning with a distillation objective, and post-training dynamic quantization. In our experiments, we show that optimizing for flat minima consistently leads to greater compressibility of parameters compared to standard Adam optimization when fine-tuning BERT models, leading to higher rates of compression with little to no loss in accuracy on the GLUE text classification and SQuAD question answering benchmarks.

Background

(BQ0): How do we train flat? **Sharpness-Aware Minimization (SAM)**

Figure 1: (left) Error rate reduction obtained by switching to SAM. Each point is a different dataset / model / data augmentation. (middle) A sharp minimum to which a ResNet trained with SGD converged. (right) A wide minimum to which the same ResNet trained with SAM converged.



$$\min_{\boldsymbol{w}} L_{\mathcal{S}}^{SAM}(\boldsymbol{w}) + \lambda\|\boldsymbol{w}\|_2^2 \quad \text{where} \quad L_{\mathcal{S}}^{SAM}(\boldsymbol{w}) \triangleq \max_{\|\boldsymbol{\epsilon}\|_p \leq \rho} L_{\mathcal{S}}(\boldsymbol{w}+\boldsymbol{\epsilon})$$

Foret, Pierre, et al. "Sharpness-aware minimization for efficiently improving generalization." ICLR 2021.

(BQ1): How do we compress? **Many ways…**

- Pruning
  - We use unstructured iterative magnitude pruning (IMP); a proportion of individual parameters is pruned repeatedly based on absolute magnitude
  - Lottery ticket hypothesis (Frankle and Carbin, 2019): Models are overparameterized… ∃ sparse subnetworks that can be trained to reach the full model's accuracy in the same number of steps
  - Either w/ or w/o *rewinding* remaining weights to pre-trained init between iterations
- Distillation
  - A smaller student model learns to mimic a teacher model's outputs
  - We use a SoTA method that prunes coarse + fine-grained (CoFi) substructures from a student model using a distillation objective (Xia et al., 2022)
- Quantization
  - Learn/calibrate transformations that map `float32` params into smaller (e.g. `int8`) representations for inference
  - We use a standard PyTorch implementation of Post-Training Dynamic Quantization

Methods

Models: pre-trained BERT models (🤗 $\text{BERT}_{\text{base}}$, + $\text{BERT}_{\text{large}}$ & $\text{RoBERTa}_{\text{base}}$)

Tasks: Text Classification (from GLUE), and Question Answering (SQuAD)

Optimizers: vanilla Adam (Kingma and Ba., 2014) vs. SAM (Foret et al., 2021) (with Adam base optimizer)

Metrics: accuracy metrics reported by related work at reference sparsity levels

Hyperparameters

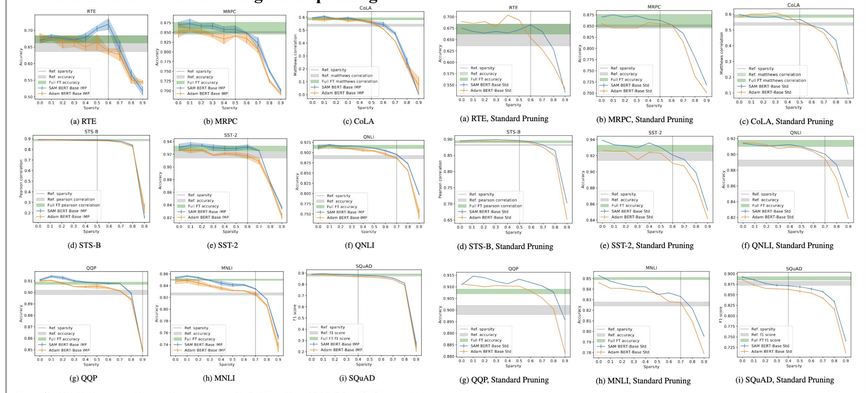
- ρ = 0.05 for size of epsilon-neighborhood in SAM
- Matched Chen et al. (2020) for unstructured IMP experiments
  - Pruned 10% of weights between each of 10 fine-tuning iterations -> 90% sparsity
  - We experiment both with and without *rewind* to $\text{BERT}_{\text{base}}$ initialization
  - Different: we set Adam w.d. = 0 in order to remove potential confound of L2 reg.
- Matched Xia et al. (2022) for structured/distillation (CoFi) experiments
  - Different: we compare teacher models trained with SAM vs vanilla Adam

Experiments

(RQ0): Does SAM help make models more robust to compression?

Yes! Models trained with SAM are more robust to:

Unstructured iterative magnitude pruning…



(a) RTE (b) MRPC (c) CoLA (a) RTE, Standard Pruning (b) MRPC, Standard Pruning (c) CoLA, Standard Pruning

(d) STS-B (e) SST-2 (f) QNLI (d) STS-B, Standard Pruning (e) SST-2, Standard Pruning (f) QNLI, Standard Pruning

(g) QQP (h) MNLI (i) SQuAD (g) QQP, Standard Pruning (h) MNLI, Standard Pruning (i) SQuAD, Standard Pruning

Figure 2: Individual plots showing sparsity vs. accuracy for GLUE tasks and SQuAD in $\text{BERT}_{\text{base}}$ models compressed via IMP with rewind to $\text{BERT}_{\text{base}}$ initialization. The vertical lines and gray horizontal bands mark reference sparsity and "winning ticket" evaluation metric values that were obtained by Chen et al. (2020). The green horizontal bands mark the initial performance of our full fine-tuned models.

Figure 3: Individual plots showing sparsity vs. accuracy for GLUE tasks and SQuAD in $\text{BERT}_{\text{base}}$ models compressed with standard pruning (IMP with no rewinding of weights). The vertical lines and gray horizontal bands mark reference sparsity and "winning ticket" evaluation metric values that were obtained by Chen et al. (2020). The green horizontal bands mark the initial performance of our full fine-tuned models.

… ✅ with… …and ✅ without…

…rewind to pre-trained $\text{BERT}_{\text{base}}$ initialization between iterations

✅ Structured pruning with a distillation objective

Dataset	Optim.	Teacher Acc.	Pruned Acc.
SST-2	Adam	$92.7_{0.1}$	$90.2_{0.5}$
(67k)	SAM	$93.1_{0.6}$	$\mathbf{91.3_{0.3}}$
QNLI	Adam	$91.5_{0.1}$	$85.9_{0.4}$
(105k)	SAM	$91.3_{0.6}$	$\mathbf{86.9_{0.4}}$
QQP	Adam	$91.0_{0.1}$	$90.0_{0.1}$
(364k)	SAM	$91.1_{0.1}$	$90.1_{0.1}$
MNLI	Adam	$84.7_{0.4}$	$80.2_{0.4}$
(393k)	SAM	$85.3_{0.2}$	$80.60.1$

Table 1: Comparison between pruned models obtained using teacher models fine-tuned with Adam and SAM optimizers. Numbers reported are $\text{mean}_{\pm\text{stddev}}$ ($n = 3$) for evaluation metrics on the development set. Compressed models are trained to reach 95% sparsity using optimal values for λ and finetuning learning rate from Xia et al. (2022)'s structured pruning setting.

✅ Post-training quantization

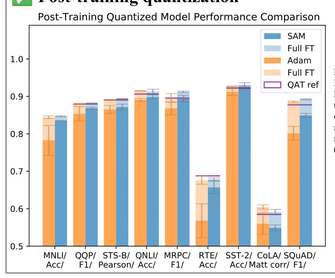
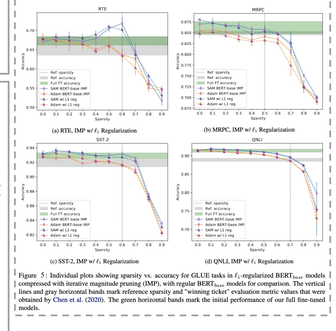


Figure 4: We compare full fine-tuned and quantized BERT-base models optimized with SAM and Adam. Error bars show standard deviations for $n = 3$. Additionally, we show that applying a simpler post-training dynamic quantization technique on a SAM-optimized model can approach the reported performance of a model quantized through quantization-aware training (QAT) (Zafrir et al., 2019).

(RQ0.5): Wait! Is SAM just doing the same thing as L1 regularization?

Probably not!



(a) RTE, IMP w/ ℓ_1 Regularization (b) MRPC, IMP w/ ℓ_1 Regularization

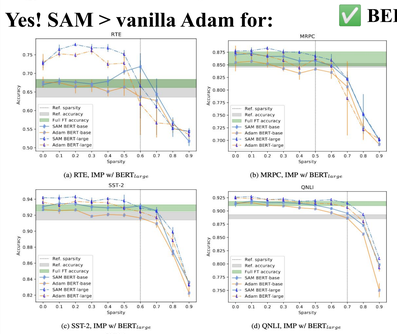
(c) SST-2, IMP w/ ℓ_1 Regularization (d) QNLI, IMP w/ ℓ_1 Regularization

Figure 5: Individual plots showing sparsity vs. accuracy for GLUE tasks in ℓ_1-regularized $\text{BERT}_{\text{base}}$ models compressed with iterative magnitude pruning (IMP), with regular $\text{BERT}_{\text{base}}$ models for comparison. The vertical lines and gray horizontal bands mark reference sparsity and "winning ticket" evaluation metric values that were obtained by Chen et al. (2020). The green horizontal bands mark the initial performance of our full fine-tuned models.

(RQ1): Does SAM benefit compressibility across different model initializations and sizes?

Yes! SAM > vanilla Adam for:

✅ **$\text{BERT}_{\text{base}}$** ⬆️⬆️⬆️



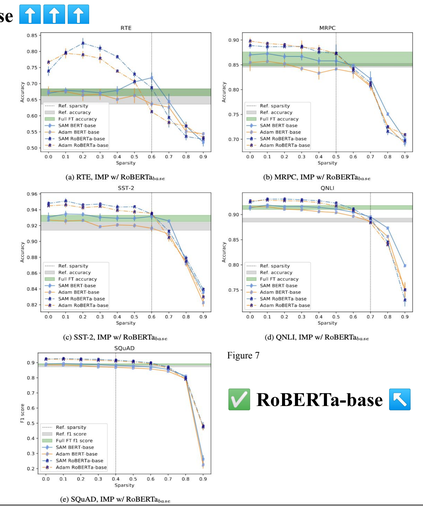
(a) RTE, IMP w/ $\text{BERT}_{\text{large}}$ (b) MRPC, IMP w/ $\text{BERT}_{\text{large}}$

(c) SST-2, IMP w/ $\text{BERT}_{\text{large}}$ (d) QNLI, IMP w/ $\text{BERT}_{\text{large}}$

(e) SQuAD, IMP w/ $\text{BERT}_{\text{large}}$

Figure 6

✅ **$\text{BERT}_{\text{large}}$** ↖️



(a) RTE, IMP w/ $\text{RoBERTa}_{\text{base}}$ (b) MRPC, IMP w/ $\text{RoBERTa}_{\text{base}}$

(c) SST-2, IMP w/ $\text{RoBERTa}_{\text{base}}$ (d) QNLI, IMP w/ $\text{RoBERTa}_{\text{base}}$

(e) SQuAD, IMP w/ $\text{RoBERTa}_{\text{base}}$

Figure 7

✅ **$\text{RoBERTa}_{\text{base}}$** ↖️

Experiments (cont'd)

*(RQ2): **How** does SAM influence model compressibility?*

SAM learns more compressible "winning ticket" *structures*

Dataset	Ticket	Optim.	Accuracy
RTE	Random	Adam	54.9 ± 1.3
(60%)	Random	SAM	55.4 ± 1.6
	Adam	Adam	63.7 ± 0.9
	Adam	SAM	61.7 ± 2.4
	SAM	Adam	70.2 ± 1.9
	SAM	SAM	71.8 ± 1.7
MRPC	Random	Adam	70.8 ± 0.8
(50%)	Random	SAM	70.1 ± 0.2
	Adam	Adam	84.2 ± 0.1
	Adam	SAM	85.3 ± 0.5
	SAM	Adam	85.3 ± 1.3
	SAM	SAM	85.7 ± 0.8

Table 2: For RTE and MRPC at their reference sparsity values, we fine-tune using 1) SAM and 2) Adam optimizers from pre-trained BERT-base initializations using only the remaining weights based on a) a random mask, b) an Adam-learned mask, and c) a SAM-learned mask.

*(RQ3): How does SAM affect compressed model **generalization?***

SAM-learned tickets transfer better across tasks

Figure 8: Heatmaps indicating the absolute difference in target task performance between SAM and Adam tickets when transferring tickets across tasks, fine-tuned with either SAM (left) or Adam (right) optimizers during IMP. Values > 0 indicate the extent to which SAM tickets transferred better than Adam tickets; Values < 0 indicate where Adam tickets transferred better than SAM. Overall, SAM tickets transfer better regardless of the final fine-tuning optimizer.

Figure 9: Heatmaps indicating the absolute difference in target task performance between SAM and Adam optimizers during fine-tuning when transferring tickets across tasks. Values greater than 0 indicate the extent to which SAM optimizer worked better than Adam; Values less than 0 indicate where Adam optimizer worked better than SAM; Values close to 0 indicate little difference.

Discussion

Key findings

- Fine-tuning with SAM improves compressibility of BERT models across a variety of settings, compared to vanilla Adam optimization
- Specifically, SAM helps us *find winning lottery tickets* during iterative magnitude pruning that even a vanilla Adam optimizer can learn on successfully:
  - The *structure* of a winning ticket learned by SAM is more important than the *optimization* over a given ticket
  - The winning tickets learned by SAM *generalize better* to other tasks
  - On the other hand, SAM does not always help us optimize a given winning ticket

(DQ0): What's the catch?

- Standard SAM is slower… bilevel optimization can be costly
- Lottery ticket-style IMP is expensive (fine-tuning x *n* iterations)
- Hardware does not necessarily support inference or training speedups proportional to size in compressed networks

(DQ1): What's next?

- SAM works, but what about more efficient alternatives, or different methods of obtaining flatness (e.g. stochastic weight averaging)?
- Do these results hold in much larger models? Other architectures? Other types of tasks or modalities?
- What role could pre-training with SAM play?