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

Clara Na **Sanket Vaibhav Mehta** **Emma Strubell**
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
{csna, svmehta, estrubel}@cs.cmu.edu

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 prior works suggesting a connection between simpler, more generalizable models and those that lie within wider loss basins, we hypothesize that optimizing for flat minima should lead to simpler parameterizations and thus more compressible models. We propose to combine sharpness-aware minimization (SAM) with various task-specific model compression methods, including iterative magnitude pruning (IMP), structured pruning with a distillation objective, and post-training dynamic quantization. Empirically, we show that optimizing for flatter minima consistently leads to greater compressibility of parameters compared to vanilla Adam when fine-tuning BERT models, with little to no loss in accuracy on the GLUE text classification and SQuAD question answering benchmarks. Moreover, SAM *finds superior winning tickets* during IMP that 1) are amenable to vanilla Adam optimization, and 2) transfer more effectively across tasks.[1]

1 Introduction

Recent advances in hardware, modeling, and optimization for deep neural networks have enabled training of substantially larger models on massive amounts of unlabeled data, leading to corresponding improvements in accuracy across a variety of tasks in NLP (Devlin et al., 2019; Brown et al., 2020; Raffel et al., 2020). Unfortunately, this sudden increase in scale of state-of-the-art models also has adverse consequences, such as reducing equity of access (Yu, 2020; Ahmed and Wahed, 2020) and increasing computational and energy requirements (Strubell et al., 2019; Dodge et al., 2022).

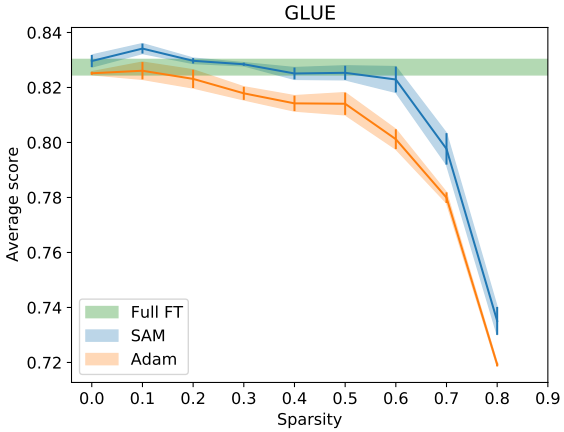


Figure 1: Average score over all GLUE tasks as a function of sparsity (% of parameters pruned) of BERT_{base} through unstructured iterative magnitude pruning. We compare the baseline Adam optimized model's performance to our model, optimized to prefer flat minima via SAM. The green horizontal bands mark initial performance of our full fine-tuned (FT) models. SAM outperforms baseline Adam across all sparsity values.

In response, *model compression* has emerged as a dominant approach to improving memory and inference efficiency in neural network models, including approaches such as knowledge distillation (Buciluǎ et al., 2006; Hinton et al., 2014; Jiao et al., 2020), model quantization (Vanhoucke et al., 2011; Shen et al., 2020) and pruning (LeCun et al., 1989; Chen et al., 2020; Xia et al., 2022).

The vast majority of work on model compression focuses on methods for selecting how and where to reduce (via quantization or distillation) or remove (by pruning) model parameters without sacrificing end-task accuracy. While these approaches are usually simple to implement and work relatively well for maintaining overall accuracy on considered benchmarks, recent work has revealed that these commonly used compression methods can result in negative impacts on model behavior that are not necessarily captured by current performance met-

[1]Code is available at https://github.com/clarana/train-flat-compress

rics. For example, in the field of computer vision, pruning has been shown to sacrifice performance on long-tail examples in order to preserve accuracy on more frequent phenomena (Hooker et al., 2020) and reduce robustness to out-of-distribution examples (Liebenwein et al., 2021). At the same time, it has also been shown that compression can sometimes have a regularizing effect, improving generalization in some settings (Ahia et al., 2021). Clearly, more work is needed better understand the relationship between compression and generalizability, and to develop improved compression methods that are informed by this knowledge.

Meanwhile, there is a growing body of work investigating curvature of the loss landscape and its relation to generalization in deep neural models. Hochreiter and Schmidhuber (1997) first defined *flat minima* as regions in parameter space where error remains relatively stable despite perturbations in parameter values, arguing that models in flat minima should correspond to simpler, more generalizable functions. More recently, Wu et al. (2017) further showed that wide loss basins correspond to low-complexity solutions,[2] and it has been shown empirically that directly optimizing for solutions in flat minima leads to improved generalization on a wide range of supervised learning tasks in both vision (Foret et al., 2021) and language (Bahri et al., 2022) modalities.

Inspired by the above results connecting wider loss regions to simpler, more generalizable models, in this work we aim to advance our understanding of model compression by examining the relationship between flat minima and compressibility in fine-tuned language models. Intuitively, model parameters in neighborhoods having uniformly low loss values should be more robust to perturbations, such as those resulting from model compression, compared to models in sharper regions, since changes to parameter values should lead to minimal change in loss with respect to the main training objective. Empirically and theoretically, previous work has also linked generalizability with compressibility, finding that neural networks whose weights reflect simpler, more general hypotheses may be more robust to compression, and compression itself can act as a regularization mechanism (Zhou et al., 2019; Liang et al., 2021; Kuhn et al., 2021). Li et al. (2020) showed that larger pre-trained language models, which are known to genereralize better, are also more compressible. Further, Bartoldson et al. (2020) connect flatness to generalization in pruned models, showing that pruning noise correlates positively with measures of flatness in a CNN for computer vision.

We investigate the relationship between flat minima and compression in large pre-trained language models by directly optimizing for flat minima during language model fine-tuning using *sharpness-aware minimization* (SAM; Foret et al. (2021)). Through extensive experiments on the GLUE text classification and SQuAD question answering benchmarks, we show that fine-tuning BERT models with SAM leads to optima in flatter basins, and compressing those models consistently results in higher model accuracy at the same level of compression compared to standard Adam-optimized baselines. Our results hold across multiple BERT variants and sizes (Devlin et al., 2019; Liu et al., 2019) and a wide variety of compression methods: Iterative magnitude pruning with and without rewinding, a structured pruning procedure that also employs knowledge distillation, and an off-the-shelf method for post-training quantization. We also show that sparse subnetworks (*winning tickets*; Frankle and Carbin (2019)) discovered by SAM are better transferable across tasks, suggesting improved generalizability. Our findings shed light on a promising new avenue for obtaining practical improvements in model compression.

2 Methods

Broadly, we are interested in understanding: 1) Are models in flatter minima more compressible? 2) If so, why? 3) Beyond task-specific accuracy, what properties do flat, compressed models have?

To provide empirical answers to these questions, our experimental setup is as follows. We fine-tune pre-trained language models on standard benchmarks using both "vanilla" Adam optimization and Sharpness-Aware Minimization (§2.1). We then experiment with a variety of strategies for compressing those fine-tuned models: iterative magnitude pruning (unstructured) with and without rewinding (§2.2.1), structured pruning using ℓ_0 regularization and a distillation obejctive (§2.2.2), and post-training dynamic quantization (§2.3). We evaluate model end-task accuracy at different compression rates, and compare that accuracy to the full (uncompressed) model and across experimental settings,

[2] Wu et al. (2017) demonstrate this theoretically for two layer feed-forward networks, and empirically for larger networks applied to computer vision.

such as varying the pre-trained language model, and transferring initializations across tasks.

2.1 Flat Minima

Sharpness-Aware Minimization (SAM). To explicitly encourage flatter loss basins, we employ the SAM (Foret et al., 2021) procedure. Given a loss function $f(w)$, SAM strives to find parameters that lie in the neighborhood with uniformly low loss by optimizing the following minimax objective:

$$\min_{w} \max_{||\epsilon||_2 \leq \rho} f(w+\epsilon) \tag{1}$$

where the maximization (or neighborhood) region is an ℓ^p ball with radius ρ for $p = 2$ in Equation (1). The gradient of the result of the above (inner) maximization problem can be approximated as:

$$\approx \nabla_w f(w)\Big|_{w+\hat{\epsilon}(\mathbf{w})} + \frac{\partial \hat{\epsilon}(\mathbf{w})}{w} \nabla_w f(w)\Big|_{w+\hat{\epsilon}(\mathbf{w})}$$
$$\text{where, } \hat{\epsilon}(\mathbf{w}) = \rho \nabla_w f(w) / \|\nabla_w f(w)\|_2$$

Foret et al. (2021) showcase that one can simplify the optimization problem without compromising the algorithm's effectiveness by dropping the second order term in the gradient, leaving us with:

$$\nabla_w \max_{||\epsilon||_2 \leq \rho} f(w+\epsilon) \approx \nabla_w f(w)\Big|_{w+\hat{\epsilon}(\mathbf{w})} \tag{2}$$

Intuitively, SAM takes a gradient step at each iteration based on the gradient estimated at the parameters yielding the highest loss ($w + \hat{\epsilon}(w)$) in an ℓ^p neighborhood around the current parameters (w).

Stochastic Weight Averaging (SWA). Although, we use SAM to optimize for flatness, there exist other alternatives to promote flatness like Stochastic Weight Averaging (Izmailov et al., 2018). SWA performs an equal average of model checkpoints along the optimization trajectory to find flatter solutions compared to vanilla optimization. We also consider SWA for our experimentation (§3.4.7).

Sharpness Metric. To verify that SAM and SWA indeed leads to flatter minima as compared to Adam, we compute a *sharpness metric* (Keskar et al., 2017), which estimates the flatness by computing the maximum value of $f(w)$ within a neighborhood region controlled by the hyperparameter ϵ. Following (Keskar et al., 2017; Mehta et al., 2021), the neighborhood region is defined as:

$$C_\epsilon = \{z \in R^p : -\epsilon(|(A^+ w)_i| + 1) \leq z_i \leq \epsilon(|(A^+ w)_i| + 1) \forall i \in \{1 \dots p\}\} \tag{3}$$

where R^p is a random subspace of the entire parameter space R^n constructed using a projection matrix $A \in R^{n \times p}$, and A^+ is the pseudo inverse of A. Concretely, the sharpness metric (lower corresponds to flatter minima) is computed as follows:

$$\phi_{w,f} := \frac{\max_{z \in C_\epsilon} f(w + Az) - f(w)}{1 + f(w)} \times 100 \tag{4}$$

To qualitatively verify for flatness, we also visualize loss contours (see Appendix A.2).

2.2 Pruning

We investigate compressibility primarily in an unstructured pruning setting. Given a network $\mathcal{N}$ and weights $\mathbf{w}$, we wish to prune a subset of individual weights to leave only a subset, $\mathbf{w}'$. A successfully pruned model has $|\mathbf{w}'| \ll |\mathbf{w}|$ while retaining good performance on the task(s) of interest.

2.2.1 Iterative Magnitude Pruning

Typically, $\mathbf{w}'$ is found through an iterative process where $\mathcal{N}$ is trained and pruned incrementally, either until some target sparsity is reached or until some larger-than-desired performance drop is observed, and a common criterion selects weights with the smallest absolute magnitude to be pruned at each iteration. In the standard pruning scenario (Renda et al., 2020a; Han et al., 2015), training simply resumes with the remaining weights after each iteration of pruning. Previous work (Renda et al., 2020a) presents evidence that rewinding remaining weights to earlier learned values may be beneficial for compressibility.

Frankle and Carbin (2019) present a formulation of iterative magnitude pruning (IMP) as a way to obtain sparse "winning tickets" $\mathbf{w}'$ that can be trained from initialization to reach performance to match that of the original full network $\mathcal{N}$ while using significantly fewer parameters. In IMP, model parameters are repeatedly reset to their original initialized values after pruning, before the next iteration of training. Reverting weights to values from an earlier point during training is also known as *rewinding* (Frankle et al., 2020). Following Chen et al. (2020), we consider both standard (no rewinding) and lottery ticket-style IMP (with rewinding) settings. In alignment with the paradigm of pre-training and fine-tuning, we treat the pre-trained model's weights as the initial weights to which parameters are reset at each iteration.

2.2.2 Structured Pruning

We also explore flat minima in a recently proposed structured pruning setting: Xia et al. (2022) incorporate a layerwise distillation objective into their structured pruning process, which dynamically maps layers between teacher and student models as structured units of varying granularity are incrementally pruned in the student model via an ℓ_0 regularizer. In our experiments, we vary only optimizer used to fine-tune the *teacher model* and compare downstream compression performance.

2.3 Post-Training Quantization

We compare performance of Int8 quantized BERT$_{base}$ models fine-tuned with Adam- and SAM-optimized models. Using a standard PyTorch implementation[3], we perform *post-training dynamic quantization*, where full-size (32-bit) floating point model weights are statically mapped to a lower precision (in our case, 8-bit integer) representation after training, and activations are dynamically reduced in precision during inference.

3 Experimentation

3.1 Research Questions

In this section, we describe a series of experiments and analyses aimed at answering the following research questions:

Q0 Does SAM help make models more robust to compression? (§3.4.1, §3.4.4, §3.4.5)

Q1 Does SAM benefit compressibility across different model initializations and sizes? (§3.4.6)

Q2 *How* does SAM influence model compressibility? (§3.4.2)

Q3 How does SAM affect compressed model properties beyond single-task accuracy? (§3.4.2, §3.4.3)

Q4 How does flatness in general, beyond SAM specifically, influence compressibility? (§3.4.7)

3.2 Datasets and Metrics

We consider eight tasks from the standard GLUE (Wang et al., 2018) benchmark for our experimentation, as well as SQuAD v1.1 (Rajpurkar et al., 2016). The GLUE datasets include MNLI (Williams et al., 2018), QQP[4], STS-B (Cer et al., 2017), QNLI (Wang et al., 2018), MRPC (Dolan and Brockett, 2005), RTE (Wang et al., 2018), SST-2 (Socher et al., 2013), and CoLA (Warstadt et al., 2019). For all experiments unless otherwise noted, we follow prior work (Chen et al., 2020) and report validation set accuracy for QQP, QNLI, MRPC, RTE, SST-2, matched accuracy for MNLI, Matthew's correlation for CoLA, Pearson correlation for STS-B, and F1 score for SQuAD. We also make use of a sharpness metric (§A.2) to quantify the flatness of the basins that our models lie in.

3.3 Implementation Details

For all experiments described in this section, we fine-tune publicly available pre-trained BERT model weights (Wolf et al., 2019). We use the uncased BERT$_{base}$ model for all experiments except when otherwise noted. For our iterative magnitude pruning experiments, we mainly set hyperparameters as mentioned by Chen et al. (2020) and follow a similar general procedure for iterative magnitude pruning, pruning an absolute 10% of prunable weights over 9 iterations to reach 90% sparsity in order to facilitate direct comparisons. Appendix A.4 contains further implementation details including hyperparameters used and explanations of when our methods differ. For our SAM optimizer, we use Adam (Kingma and Ba, 2014) as our base optimizer, and following (Mehta et al., 2021; Bahri et al., 2022) set ρ to 0.05. Appendix A.2 contains implementation details for SWA.

3.4 Results

3.4.1 Iterative Magnitude Pruning (Q0)

With rewind to BERT$_{base}$ We investigate the SAM procedure's effectiveness in uncovering winning tickets (Frankle and Carbin, 2019). The **IMP** section of Table 1 shows that **optimizing with SAM throughout iterative magnitude pruning allows pruned models to retain higher performance** at reference sparsity levels when compared to models trained with vanilla Adam optimizer.

The plots in Figure 2 show evaluation metrics over successive IMP iterations for individual GLUE tasks. We see that although initial performance of Adam- and SAM-optimized models is usually comparable, promoting flat minima during

[3] https://pytorch.org/docs/stable/quantization.html

[4] https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs

	Dataset Sparsity Metric	MNLI 70% Match/Mismatch acc.	QQP 90% Acc.	STS-B 50% Pearson Cor.	QNLI 70% Acc.	MRPC 50% Acc.	RTE 60% Acc.	SST-2 60% Acc.	CoLA 50% Matthew's Cor.	SQuAD 40% F1
Full	Adam	$84.6_{0.1}/83.6_{0.3}$	$89.1_{0.1}$	$84.0_{0.4}$	$90.8_{0.2}$	$82.6_{1.4}$	$67.0_{1.4}$	$93.2_{0.6}$	$53.3_{1.2}$	$88.5_{0.2}$
	SAM	$85.0_{0.1}/84.5_{0.1}$	$89.2_{0.2}$	$84.7_{0.1}$	$90.8_{0.4}$	$82.9_{1.3}$	$65.5_{0.8}$	$93.6_{0.1}$	$54.1_{1.3}$	$89.2_{0.1}$
IMP	Adam	$82.3_{0.3}/81.4_{0.2}$	$83.0_{0.0}$	$83.3_{0.3}$	$88.6_{0.3}$	$81.5_{1.3}$	$63.3_{2.8}$	$92.2_{0.4}$	$47.7_{1.5}$	$86.9_{0.3}$
	SAM	$\mathbf{83.2_{0.2}/82.5_{0.4}}$	$\mathbf{85.4_{0.1}}$	$\mathbf{84.1_{0.1}}$	$\mathbf{89.4_{0.2}}$	$\mathbf{83.6_{0.2}}$	$\mathbf{65.0_{1.5}}$	$\mathbf{92.9_{0.5}}$	$\mathbf{49.5_{2.0}}$	$\mathbf{87.8_{0.2}}$
	*SWA	$83.1_{0.1}/82.1_{0.1}$	$\boldsymbol{87.7_{0.2}}$	$\boldsymbol{85.3_{0.4}}$	$89.0_{0.2}$	$\boldsymbol{83.7_{0.1}}$	$\boldsymbol{67.4_{0.4}}$	$92.7_{0.3}$	$\boldsymbol{50.9_{1.0}}$	–
Std	Adam	$82.4_{0.3}/81.1_{0.5}$	$87.2_{0.1}$	$83.9_{0.1}$	$88.8_{0.1}$	$82.9_{1.0}$	$64.4_{1.6}$	$92.3_{0.2}$	$50.9_{0.2}$	$86.3_{0.2}$
	SAM	$83.2_{0.1}/81.8_{0.1}$	$87.4_{0.4}$	$84.6_{0.1}$	$89.4_{0.1}$	$81.9_{1.2}$	$64.3_{0.8}$	$93.0_{0.6}$	$51.3_{2.0}$	$87.1_{0.2}$

Table 1: At full size and at Chen et al. (2020)'s reference sparsities, we report task-specific metrics for Adam and SAM-optimized BERT$_{base}$ models in their (1) Iterative Magnitude Pruning (**IMP**) and (2) **Std** pruning settings. We report mean and standard deviation calculated over 3 random seeds. All GLUE results are reported for test sets. We report results on the development set for SQuAD as test set evaluation is unavailable for v1.1. Table 5 in Appendix A.5 contains comparison with reference (**Ref**) values using development sets. *Additionally, we report test accuracy metrics in IMP models trained with stochastic weight averaging (SWA), another optimization method which empirically leads to flatter minima. We observe that optimizing with SAM or SWA throughout iterative magnitude pruning allows pruned models to retain higher performance at reference sparsity levels when compared to models trained with vanilla Adam optimizer.

IMP with SAM leads to either 1) higher performance compared to Adam at Chen et al. (2020)'s reference sparsity level[5] or 2) performance comparable to the full sized model at higher sparsity levels, if not both, for all but one smaller GLUE task (CoLA).

Moreover, for some tasks (RTE, SST-2), **the final pruned model's accuracy tends to be higher than the full BERT$_{base}$ fine-tuned model's**. This is an especially striking result given that we reset remaining weights to the BERT$_{base}$ initialization after each successive iteration of pruning; there is no progressive learning of weights from iteration to iteration. Instead, we reach higher accuracy simply by optimizing over the learned substructures rather than the full network.

Both SAM and vanilla Adam induce marginal improvements compared to the full fine-tuned model with 10% pruning for some tasks (e.g. 2b, 2f, 2g), which we might attribute to slight pruning acting as a structure regularizer. However, only SAM-optimized models sometimes continue to exhibit improvements in much later stages of pruning.

"Standard pruning" without rewind We also investigate SAM and vanilla Adam optimizers in the standard pruning setting, where we continue training immediately after pruning in each iteration, without resetting remaining weights to pre-trained BERT$_{base}$ initialization. The **Std** section of Table 1 shows these results.

SAM-optimized models retain more of the full sized model's performance at Chen et al. (2020)'s reference sparsity levels. However, the trend is not more stark in this setting, which hints at the structure of winning tickets found by SAM playing an important role.

We more directly investigate *how* SAM benefits model compressibility in Section 3.4.2, but we also take care to rule out the possibility that SAM is simply acting as an implicit ℓ_1 regularizer (A.7).

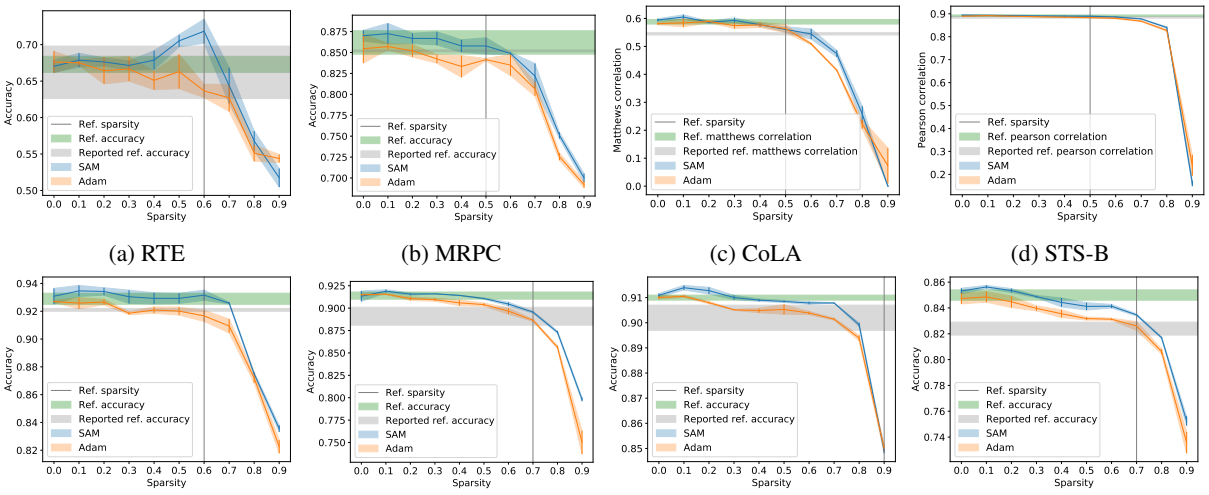
3.4.2 Analysis: Answering the Structure vs. Optimization Question (Q2, Q3)

In this experimental setting, we aim to disentangle the effects of the *structure*[6] of the pruning masks learned using different optimizers, from the *optimization* over given substructures using different optimizers. Figure 3 displays our results.

We observe that, in general, subnetworks learned through IMP greatly outperform random subnetworks of the same sparsity when trained. Subnetworks found with SAM tend to outperform those found with vanilla Adam, especially with subsequent optimization with Adam. Furthermore, training any given IMP-learned subnetwork with SAM yields modest improvements in accuracy compared to fine-tuning with vanilla Adam.

[5] Our Adam optimized models often differ in performance from Chen et al. (2020)'s at iteration 0 – the fairest comparison is between our implementation of Adam and SAM optimized models, and our reference performance bands in Figure 2 are often much higher than originally reported.

[6] In this work, we refer to "winning tickets" and "structures" thereof interchangeably, although, as observed by Frankle and Carbin (2019), winning tickets are conditioned on models' (in our case, pre-trained) initializations.



(a) RTE (b) MRPC (c) CoLA (d) STS-B

(e) SST-2 (f) QNLI (g) QQP (h) MNLI

Figure 2: Individual plots showing sparsity vs. task metrics (validation set) for GLUE throughout IMP. 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 (uncompressed) models.

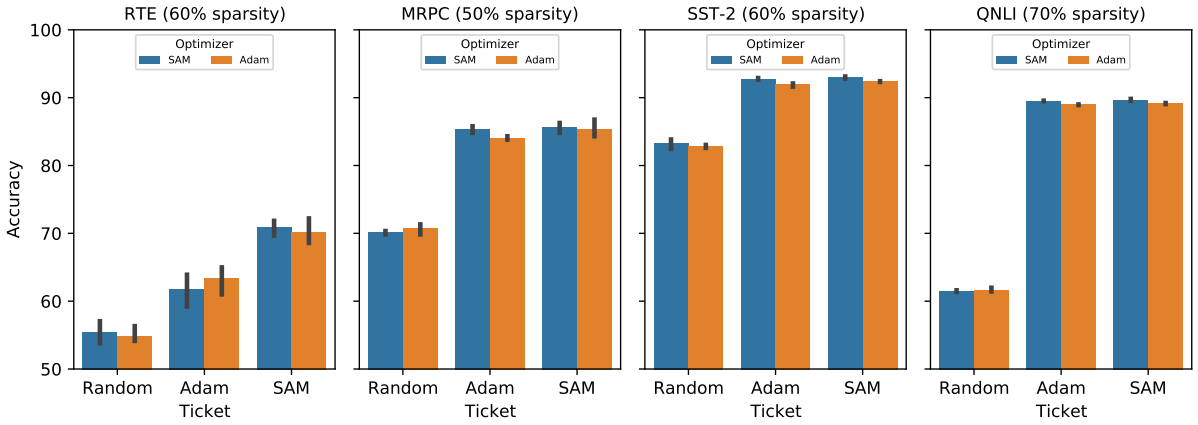


Figure 3: We compare optimizers' *learned tickets*, as well as their performance in *optimization over a given ticket*. For select GLUE tasks at their reference sparsity values, we fine-tune pruned subnetworks of pre-trained BERT_{base} initializations networks based on 1) random masks, 2) Adam-learned masks, and 3) SAM-learned masks, using a) SAM and b) Adam optimizers. SAM optimization over SAM- and Adam-learned winning tickets tends to yield marginal improvements compared to Adam optimization. Comparing bar heights from left to right within each figure allows us to see that, at least when Adam is used for final fine-tuning, a Random- $\ll$ Adam- $<$ SAM-learned masks. Exact values are reported in Table 6, and Figure 12 in Appendix shows an alternative view of the data.

3.4.3 Analysis: Comparing Transferability of Winning Tickets (Q3)

We explore the extent to which SAM tickets are more or less transferable across tasks compared to tickets discovered by Adam (Figure 4). For consistency, we use 70% sparsity tickets for all tasks evaluated. SAM-learned tickets tend to transfer better across tasks than Adam-learned tickets. This complements our findings in §3.4.2, which can be interpreted as a study on transferability of winning tickets between optimizers instead of across tasks.

We also compare SAM versus Adam as an optimizer for fine-tuning in this setting, and found that SAM did *not* seem to work better overall as an optimizer, given a ticket for a different task (see Figure 13 in Appendix). Note that this does not directly contradict our results from §3.4.2; SAM optimization does typically benefit same-task performance (see diagonals in these figures).

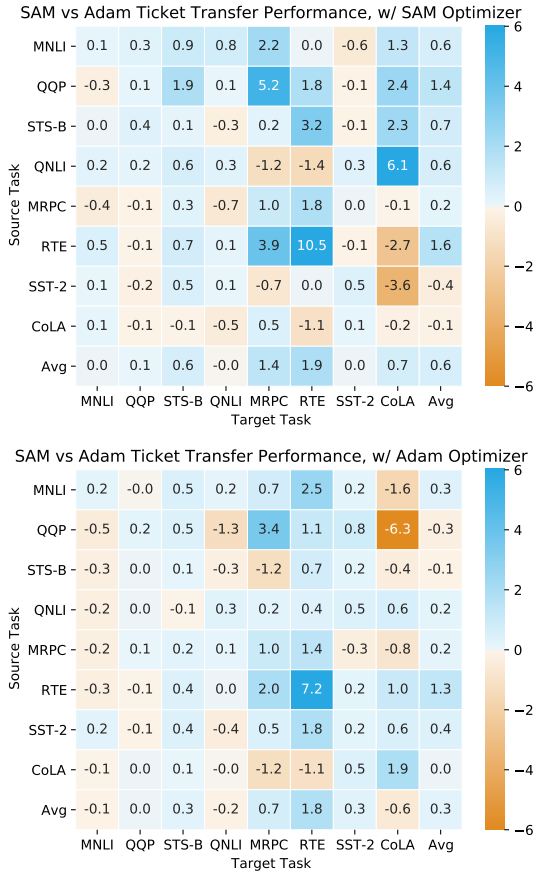


Figure 4: Heatmaps indicating the *difference* in target task performance between 70% sparsity SAM and Adam tickets when transferring tickets across tasks, fine-tuned with either SAM (top) or Adam (bottom) optimizers during IMP. Values > 0 indicate the extent to which SAM tickets transferred better than Adam tickets; Values < 0 indicate where Adam tickets transferred better than SAM. Overall, SAM tickets transfer better regardless of the final fine-tuning optimizer. Note that the positive values along the diagonal indicate superior SAM ticket performance in the single task setting, even with "transfer" between optimizers.

3.4.4 Structured Pruning (Q0)

We use full-size fine-tuned BERT$_{base}$ models from §3.4.1 as teacher models for the layerwise distillation objective used throughout the structured pruning procedure. The pruning procedure itself is the same as Xia et al. (2022)'s, regardless of whether the teacher model was originally trained using SAM or vanilla Adam. Appendix A.10 contains further implementation details.

In Table 2, we show that SAM-optimized teacher models improve compressed student model performance in this structured pruning setting. SAM outperforming a vanilla Adam optimizer in this setting is a particularly desirable goal from a practical standpoint. First, unlike in a full IMP setting, we only use SAM for a single fine-tuning in the pruning pipeline, and so we only incur the computational overhead associated with SAM for a fraction of the overall pruning process. Second, training time for CoFi's pruning process itself has a tenfold speedup compared to the TinyBERT baseline (Jiao et al., 2020). Finally, the pruned models obtained in this setting perform inference as quickly as TinyBERT, which amounts to a tenfold speedup compared to the full BERT$_{base}$ model.

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	$85.3_{0.2}$	$80.6_{0.1}$

Table 2: Comparison between pruned models obtained using teacher models fine-tuned with Adam and SAM optimizers. Numbers reported are means$_{stddev}$ ($n = 3$) 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.

3.4.5 Post-Training Quantization (Q0)

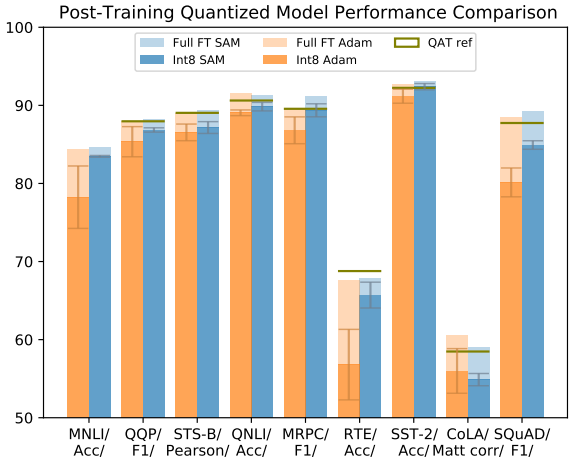


Figure 5: We compare full fine-tuned and quantized BERT-base models optimized with SAM and Adam. Error bars show standard deviations for $n = 3$. Additionally, we show that applying a simpler post-training dynamic quantization technique on a SAM-optimized model can approach the reported performance of a model quantized through quantization-aware training (QAT) (Zafrir et al., 2019).

From Figure 5 (see A.11 for actual numbers), we

can make a few key observations: 1) SAM-trained models retain higher task performance after quantization compared to Adam-trained models. 2) SAM-trained models' higher performance is also more *stable* across random seeds. Finally, 3) for some tasks, our SAM-trained models compressed with simple post-training dynamic quantization meet or approach the performance of Zafrir et al. (2019)'s models quantized through quantization-aware training (QAT), which tends to yield better compressed models but is more complex to implement and use. The benefits of quantization vary under different settings and hardware, but our BERT$_{base}$ models quantized to Int8 precision have a 2.5x reduction in storage requirements and 1.5-2x faster inference.

3.4.6 Evaluating Other Models (Q1)

In order to explore the applicability of our findings across model sizes and other BERT variants, we experiment with SAM and Adam optimizers on BERT$_{large}$ and RoBERTa$_{base}$ models in the IMP setting of §3.4.1. We find that, indeed, SAM-optimized models fare better than Adam-optimized models in these other BERT variants. A.12 includes details relevant for reproducibility, as well as task-specific results.

More generally, our proposal to "train flat" with SAM is compatible with Li et al. (2020)'s recommendation to "train large, then compress", and with starting with a better-performing model before compression. Starting with BERT$_{large}$ or RoBERTa$_{base}$ can lead to clearly higher compressed accuracy at similar target sparsity levels and rarely leads to significantly *worse* performance. However, the higher performance often does not simply follow a parallel pattern as in BERT$_{base}$ models throughout pruning; the initial performance gaps between BERT$_{large}$ and BERT$_{base}$ models tend to be preserved slightly more reliably at higher sparsity levels than the gaps between RoBERTa$_{base}$ and BERT$_{base}$ models. This prompts further investigation into the properties of the subnetworks found in these BERT variants and appeals to the potential compressibility of even larger models when flatness-optimized.

3.4.7 Stochastic Weight Averaging (Q4)

We conduct experiments matching the IMP setting with rewind from §3.4.1 using stochastic weight averaging (SWA). In Table 1, we report results on GLUE test set for SWA with IMP. Similar to SAM, we observe that SWA is superior to Adam optimized models at reference sparsity levels.

4 Related Work

In this section, we draw connections to key related work. For a more general description of related work, please refer to A.1 in Appendix.

First, we briefly recount previous work that we consider for our experimentation. We use Adam (Kingma and Ba, 2014) as a base optimizer, with comparisons between vanilla Adam and the addition of Sharpness-Aware-Minimization (Foret et al., 2021). We make further comparisons in a subset of our experiments with Stochastic Weight Averaging (Izmailov et al., 2018) as an alternative method of inducing flatness. Using Keskar et al. (2017)'s ϵ-sharpness metric, and based on Mehta et al. (2021)'s implementation, we verify that SAM and SWA induce flatness. We fine-tune BERT models (Devlin et al., 2019; Liu et al., 2019) for GLUE (Wang et al., 2018) and SQuAD (Rajpurkar et al., 2016) language tasks. The compression methods we couple with "training flat" include: 1) unstructured IMP to find winning lottery tickets (Frankle and Carbin, 2019), referring to Chen et al. (2020)'s implementation and reported results for making direct comparisons; 2) Xia et al. (2022)'s structured pruning method with a distillation objective; and 3) off-the-shelf post-training quantization, which we compare with reported results from Zafrir et al. (2019)'s quantization-aware training method.

Flatness and generalization Prior work has investigated the connection between flat minima and generalization (i.e., the gap between training accuracy and holdout set accuracy), starting with Hochreiter and Schmidhuber (1997).

Subsequent work has continued on this front, exploring notions of sharpness and their predictiveness of generalization under different conditions (Bisla et al., 2022), as well as empirical evaluations of flatness-inducing methods and their effects on generalization. Jiang* et al. (2020) find that sharpness is empirically predictive of generalization, including in particular a perturbation magnitude-aware metric very similar to the ϵ-sharpness metric introduced in (Keskar et al., 2017) and used in our paper. In this work, however, separate from generalization, we primarily focus on investigating the underexplored relationship between flatness and *compression*.

Flatness and compression Previous work has mentioned flatness in the context of pruning. In fact, Hochreiter and Schmidhuber (1997)'s orig-

inal "Flat Minimum Search" algorithm is explicitly designed to prune units, weights, and input lines as a mechanism for finding flat minima. LeCun et al. (1989) also propose pruning unimportant weights as a way to obtain improved generalization, although without any notions of flatness.

Since these earlier works, the deep learning landscape has changed such that effective model compression itself is now often a goal; model efficiency in terms of size and latency is a common priority.

To the best of our knowledge, we are the first to directly relate loss landscape flatness with model compress*ibility*. We view our results as complementary to the concurrent work by Paul et al. (2022), who study properties of winning tickets found during iterative magnitude pruning and find that IMP itself preferentially prunes parameters lying along flatter directions in the loss landscape; they also theorize that flatter landscapes allow for more aggressive pruning. We note that the optimizer, data, and model architectures they use are different from ours. While Paul et al. (2022) use Stochastic Gradient Descent for image classification tasks on ResNet architectures, we use Adam as a base optimizer for text classification and question answering tasks with BERT architectures. Nonetheless, to the extent that findings from both papers can generalize to other settings, Paul et al. (2022) lay theoretical groundwork which supports our explicit suggestion to "train flat" as a strategy for *inducing* greater compressibility in neural models.

5 Discussion

We show that in general, SAM helps models retain higher accuracy on a variety of language tasks at higher sparsity levels. This holds true in multiple unstructured iterative magnitude pruning settings, as well as in a structured pruning setting with a distillation objective. Moreover, our additional experiments and analyses point to SAM-learned *structures* playing an important role in compressibility, as well as transferring well across tasks.

5.1 Future work

Beyond SAM and SWA (Q4) In this paper, we explore (Foret et al., 2021)'s Sharpness-Aware-Minimization procedure specifically as a method for directly reaching flat minima and conduct additional comparisons with stochastic weight averaging (Izmailov et al., 2018). However, other methods such as entropy SGD (Chaudhari et al., 2017) and label noise SGD (Damian et al., 2021) have also been shown to encourage convergence to flat minima. Further exploration would provide clarity on the role of flatness in general in model compressibility versus properties specific to SAM and SWA.

The role of pre-training (Q4) The BERT models we fine-tune in this work are pre-trained on various self-supervised auxiliary objectives with the goal of learning useful general representations of the English language. Recent work (Mehta et al., 2021) has found that, empirically, pre-training is associated with convergence to flatter minima than obtained by training on the same task to the same accuracy from a random initialization. Subsequent work could compare compressibility of pre-trained models versus models trained from random initialization, as well as investigate the potential for further improving compressibility through flat *pre*-training. Inducing flatness during pre-training would facilitate further experimentation with end-task-agnostic knowledge distillation.

Other evaluations (Q3) In general, we evaluate model performance in terms of task-specific metrics (on development/test splits) throughout this work. However, since compression is associated with negative consequences for model behavior not captured by task-specific accuracy (Hooker et al., 2020; Liebenwein et al., 2021), there is a particular need for work to understand and influence these qualities. Ribeiro et al. (2020) propose behavioral testing of NLP models to evaluate specific capabilities such as robustness to typos and simple coference resolution. Xu et al. (2021) propose measures of probability and label *loyalty* and robustness to input perturbations for evaluating compressed models beyond preserved accuracy. We briefly discuss preliminary observations of model behaviors with respect to Ribeiro et al. (2020)'s Checklist items in §A.13 in Appendix, but we emphasize that more work is needed to understand behaviors and behavior shifts of vanilla Adam and SAM models before and throughout pruning.

Limitations

Many of the limitations of our work have to do with its computational requirements. First, the standard implementation of Sharpness-Aware Minimization that we use incurs significant computational overhead, so further investigation into adaptive (Kwon

et al., 2021) and more efficient (Du et al., 2022b,a) variations of SAM is warranted before considering adoption of our methods in practice. Meanwhile, we control for the number of training steps and sparsity level of our models without regard to wall-clock time in our experiments, but fine-tuning BERT$_{base}$ with standard SAM typically results in 1.5x-2x slower optimization steps. In general, many of our approaches are not strategies that can simply be applied off the shelf for practical benefits. In particular, current hardware and frameworks do not typically support reliable and proportionate efficiency gains from quantization to arbitrary precision and unstructured magnitude pruning. Moreover, the computation and storage requirements for our iterative magnitude pruning scheme are tenfold compared to the typical single fine-tuning performed on a pre-trained language model, due to the ten iterations performed and checkpoints saved. We benefited from access to a large compute cluster with dozens of GPUs, including A6000, a100, v100, RTX3090, RTX8000, and 2080Ti GPUs. We estimate having used at least 1000 GPU hours across experiments for this work, which inherently limits the full reproducibility of our results in limited compute scenarios.

Additionally, although we focus on optimizing for flatness directly in our experimentation with SAM (Foret et al., 2021), other methods, such as weight averaging (Izmailov et al., 2018) (which we explore in less detail), entropy SGD (Chaudhari et al., 2017), and label noise SGD (Damian et al., 2021), have also been shown to encourage convergence to flat minima. Without explicitly exploring compressibility in models that have reached flat minima in alternative ways, we refrain from making strong claims about flat minima in general in this work.

Furthermore, the tasks we train our models on are limited to sentence classification and question answering tasks, all in only the English language. We evaluate our methods using only BERT variants: pre-trained language models trained with a token masking objective.

Finally, although our measures of end-task accuracy and degree of compression are standard and useful for evaluating and comparing compressed models, they do not provide a complete picture of model behaviors and capabilities. Other desirable characteristics in compressed models (and models in general) include but are not limited to robustness to distribution shift, stability against catastrophic forgetting, and fairness in performance across demographic groups..

Ethics Statement

We reiterate that we are not proposing that our strategies or models be adopted off the shelf as is. This is especially true because our work does not include rigorous analysis of our compressed models' properties and behaviors outside of task accuracy, resilience to compression, and transferability to other tasks. Detailed study of properties such as long-tail performance, robustness to data distribution shifts, and fairness in performance across demographic groups, for example, which have important real-world implications, is outside of the scope of our current work. However, preliminary evaluations on certain model behaviors and properties suggests that many of our models which achieve high end-task performance are vulnerable to simple perturbations in data and lack basic desirable linguistic capabilities (although not necessarily more so than is "typical" of language models (Ribeiro et al., 2020; Xu et al., 2021).

Acknowledgements

We are grateful to our anonymous reviewers, both of whom provided thoughtful and helpful feedback on extremely short notice. We would like to thank COMEDY (**CO**horts of **M**aarten Sap, **E**mma Strubell, **D**aniel Fried, and **Y**onatan Bisk) lab members for sharing insights and intuitions during initial discussions; Nupoor Gandhi, Jared Fernandez, Jeremiah Milbauer, Zhisong Zhang, and Josh Zhanson also gave constructive feedback on drafts and figures. We would like to acknowledge CMU Workhorse and TIR groups for providing compute resources for this work. Outside of CMU, we are appreciative of Mengzhou Xia for helping us reproduce structured pruning experiments using CoFi. This project is funded in part by DSO National Laboratories.

References

Orevaoghene Ahia, Julia Kreutzer, and Sara Hooker. 2021. The low-resource double bind: An empirical study of pruning for low-resource machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3316–3333, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Nur Ahmed and Muntasir Wahed. 2020. The de-democratization of ai: Deep learning and the compute divide in artificial intelligence research.

Dara Bahri, Hossein Mobahi, and Yi Tay. 2022. Sharpness-aware minimization improves language model generalization. In *Annual Conference of the Association for Computational Linguistics (ACL)*, Dublin, Ireland.

Haoli Bai, Wei Zhang, Lu Hou, Lifeng Shang, Jin Jin, Xin Jiang, Qun Liu, Michael Lyu, and Irwin King. 2021. BinaryBERT: Pushing the limit of BERT quantization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4334–4348, Online. Association for Computational Linguistics.

Brian Bartoldson, Ari Morcos, Adrian Barbu, and Gordon Erlebacher. 2020. The generalization-stability tradeoff in neural network pruning. *Advances in Neural Information Processing Systems*, 33:20852–20864.

Aishwarya Bhandare, Vamsi Sripathi, Deepthi Karkada, Vivek Menon, Sun Choi, Kushal Datta, and Vikram Saletore. 2019. Efficient 8-bit quantization of transformer neural machine language translation model. *arXiv preprint arXiv:1906.00532*.

Devansh Bisla, Jing Wang, and Anna Choromanska. 2022. Low-pass filtering sgd for recovering flat optima in the deep learning optimization landscape. In *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 8299–8339. PMLR.

Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. 2020. What is the state of neural network pruning? In *Proceedings of Machine Learning and Systems (MLSys*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, page 535–541, New York, NY, USA. Association for Computing Machinery.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*.

Pratik Chaudhari, Anna Choromańska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer T. Chayes, Levent Sagun, and Riccardo Zecchina. 2017. Entropy-sgd: Biasing gradient descent into wide valleys. *ArXiv*, abs/1611.01838.

Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. 2020. The lottery ticket hypothesis for pre-trained bert networks. In *Advances in Neural Information Processing Systems*, volume 33, pages 15834–15846. Curran Associates, Inc.

Xiaohan Chen, Yu Cheng, Shuohang Wang, Zhe Gan, Zhangyang Wang, and Jingjing Liu. 2021. EarlyBERT: Efficient BERT training via early-bird lottery tickets. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2195–2207, Online. Association for Computational Linguistics.

Alex Damian, Tengyu Ma, and Jason D. Lee. 2021. Label noise SGD provably prefers flat global minimizers. In *Advances in Neural Information Processing Systems*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

James Diffenderfer and Bhavya Kailkhura. 2021. Multi-prize lottery ticket hypothesis: Finding accurate binary neural networks by pruning a randomly weighted network. In *International Conference on Learning Representations*.

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. 2017. Sharp minima can generalize for deep nets. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1019–1028. PMLR.

Jesse Dodge, Taylor Prewitt, Remi Tachet des Combes, Erika Odmark, Roy Schwartz, Emma Strubell, Alexandra Sasha Luccioni, Noah A. Smith, Nicole DeCario, and Will Buchanan. 2022. Measuring the carbon intensity of ai in cloud instances. In *ACM Conference on Fairness, Accountability, and Transparency (ACM FAccT)*.

Bill Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Third International Workshop on Paraphrasing (IWP2005)*.

Jiawei Du, Hanshu Yan, Jiashi Feng, Joey Tianyi Zhou, Liangli Zhen, Rick Siow Mong Goh, and Vincent Tan. 2022a. Efficient sharpness-aware minimization for improved training of neural networks. In *International Conference on Learning Representations*.

Jiawei Du, Daquan Zhou, Jiashi Feng, Vincent Y. F. Tan, and Joey Tianyi Zhou. 2022b. Sharpness-aware training for free.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. 2021. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*.

Jonathan Frankle and Michael Carbin. 2019. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. 2020. Linear mode connectivity and the lottery ticket hypothesis. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org.

Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. 2021. A survey of quantization methods for efficient neural network inference.

R.M. Gray and D.L. Neuhoff. 1998. Quantization. *IEEE Transactions on Information Theory*, 44(6):2325–2383.

Demi Guo, Alexander Rush, and Yoon Kim. 2021. Parameter-efficient transfer learning with diff pruning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4884–4896, Online. Association for Computational Linguistics.

Song Han, Huizi Mao, and William J. Dally. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding.

Yaru Hao, Li Dong, Furu Wei, and Ke Xu. 2019. Visualizing and understanding the effectiveness of bert. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4134–4143.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2014. Distilling the knowledge in a neural network. In *NIPS 2014 Deep Learning Workshop*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Flat minima. *Neural computation*, 9(1):1–42.

Sara Hooker, Nyalleng Moorosi, Gregory Clark, Samy Bengio, and Emily Denton. 2020. Characterising bias in compressed models. In *Fifth Workshop on Human Interpretability in Machine Learning (WHI)*.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. 2018. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew G. Howard, Hartwig Adam, and Dmitry Kalenichenko. 2018. Quantization and training of neural networks for efficient integer-arithmetic-only inference. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2704–2713.

Yiding Jiang*, Behnam Neyshabur*, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. 2020. Fantastic generalization measures and where to find them. In *International Conference on Learning Representations*.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online. Association for Computational Linguistics.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 2017. On large-batch training for deep learning: Generalization gap and sharp minima. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Sehoon Kim, Amir Gholami, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. 2021. I-bert: Integer-only bert quantization. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5506–5518. PMLR.

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

Lorenz Kuhn, Clare Lyle, Aidan N. Gomez, Jonas Rothfuss, and Yarin Gal. 2021. Robustness to pruning predicts generalization in deep neural networks.

Jungmin Kwon, Jeongseop Kim, Hyunseo Park, and In Kwon Choi. 2021. Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5905–5914. PMLR.

François Lagunas, Ella Charlaix, Victor Sanh, and Alexander Rush. 2021. Block pruning for faster transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10619–10629, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yann LeCun, John Denker, and Sara Solla. 1989. Optimal brain damage. In *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann.

Zhuohan Li, Eric Wallace, Sheng Shen, Kevin Lin, Kurt Keutzer, Dan Klein, and Joey Gonzalez. 2020. Train large, then compress: Rethinking model size for efficient training and inference of transformers. In *ICML*, pages 5958–5968.

Chen Liang, Simiao Zuo, Minshuo Chen, Haoming Jiang, Xiaodong Liu, Pengcheng He, Tuo Zhao, and Weizhu Chen. 2021. Super tickets in pre-trained language models: From model compression to improving generalization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6524–6538, Online. Association for Computational Linguistics.

Lucas Liebenwein, Cenk Baykal, Brandon Carter, David Gifford, and Daniela Rus. 2021. Lost in pruning: The effects of pruning neural networks beyond test accuracy. In *Proceedings of Machine Learning and Systems*, volume 3, pages 93–138.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

Christos Louizos, Max Welling, and Diederik P. Kingma. 2018. Learning sparse neural networks through l_0 regularization. In *International Conference on Learning Representations*.

Sanket Vaibhav Mehta, Darshan Patil, Sarath Chandar, and Emma Strubell. 2021. An empirical investigation of the role of pre-training in lifelong learning. *arXiv preprint arXiv:2112.09153*.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. 2020. What is being transferred in transfer learning? In *Advances in Neural Information Processing Systems*, volume 33, pages 512–523. Curran Associates, Inc.

Mansheej Paul, Feng Chen, Brett W. Larsen, Jonathan Frankle, Surya Ganguli, and Gintare Karolina Dziugaite. 2022. Unmasking the lottery ticket hypothesis: What's encoded in a winning ticket's mask?

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Alex Renda, Jonathan Frankle, and Michael Carbin. 2020a. Comparing rewinding and fine-tuning in neural network pruning. In *International Conference on Learning Representations*.

Alex Renda, Jonathan Frankle, and Michael Carbin. 2020b. Comparing rewinding and fine-tuning in neural network pruning. In *International Conference on Learning Representations*.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

Victor Sanh, Thomas Wolf, and Alexander Rush. 2020. Movement pruning: Adaptive sparsity by fine-tuning. In *Advances in Neural Information Processing Systems*, volume 33, pages 20378–20389. Curran Associates, Inc.

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. 2020. Q-bert: Hessian based ultra low precision quantization of bert. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8815–8821.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for BERT model compression. In *Proceedings of the 2019 Conference on*

Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 4323–4332, Hong Kong, China. Association for Computational Linguistics.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. MobileBERT: a compact task-agnostic BERT for resource-limited devices. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2158–2170, Online. Association for Computational Linguistics.

Vincent Vanhoucke, Andrew Senior, and Mark Z. Mao. 2011. Improving the speed of neural networks on cpus. In *Deep Learning and Unsupervised Feature Learning Workshop, NIPS 2011*.

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

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Lei Wu, Zhanxing Zhu, and Weinan E. 2017. Towards understanding generalization of deep learning: Perspective of loss landscapes. In *ICML Workshop on Principled Approaches to Deep Learning (PADL)*.

Mengzhou Xia, Zexuan Zhong, and Danqi Chen. 2022. Structured pruning learns compact and accurate models. In *Association for Computational Linguistics (ACL)*.

Canwen Xu, Wangchunshu Zhou, Tao Ge, Ke Xu, Julian McAuley, and Furu Wei. 2021. Beyond preserved accuracy: Evaluating loyalty and robustness of BERT compression. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10653–10659, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Peter K. Yu. 2020. The algorithmic divide and equality in the age of artificial intelligence. *Florida Law Review*, 72:331–89.

Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. 2019. Q8bert: Quantized 8bit bert. *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing - NeurIPS Edition (EMC2-NIPS)*, pages 36–39.

Chunting Zhou, Jiatao Gu, and Graham Neubig. 2020. Understanding knowledge distillation in non-autoregressive machine translation. In *International Conference on Learning Representations*.

Wenda Zhou, Victor Veitch, Morgane Austern, Ryan P. Adams, and Peter Orbanz. 2019. Non-vacuous generalization bounds at the imagenet scale: a PAC-bayesian compression approach. In *International Conference on Learning Representations*.

A Appendix

A.1 Extended Related Work

Model compression in NLP Approaches for model compression aim to replicate the end-task performance of a large, accurate model while requiring fewer parameters and floating-point operations, and many approaches for model compression have been successfully applied to tasks in NLP. Specific model compression techniques include *pruning*, where individual model parameters (unstructured pruning) or entire weight matrices (structured pruning) are removed entirely (LeCun et al., 1989; Blalock et al., 2020; Sanh et al., 2020; Lagunas et al., 2021), *quantization*, where model weights, activations, gradients, and/or add-multiply accumulators are reduced in precision from 32-bit floating point representations to floating or fixed-point representations as low as one or two bits (Gray and Neuhoff, 1998; Vanhoucke et al., 2011; Gholami et al., 2021), and *knowledge distillation* where a smaller model is trained to replicate the predictions, and often intermediate embedded representations, of a larger model (Buciluǎ et al., 2006; Hinton et al., 2014; Sanh et al., 2019; Jiao et al., 2020).

Unstructured pruning can achieve some of the highest sparsity levels using various criteria and schedules for determining which parameters to prune (Frankle and Carbin, 2019; Chen et al., 2020; Sanh et al., 2020; Guo et al., 2021), though sparsity patterns resulting from unstructured pruning often do not result in latency reduction on modern accelerator hardware. Work in structured pruning has explored removing entire parameter matrices such as self-attention heads, hidden units, and entire layers (Michel et al., 2019; Lagunas et al., 2021; Xia et al., 2022), with basic underlying hardware constraints in mind. Pruning is often combined with a distillation objective, which provides complementary gains, likely by reducing complexity of the dataset (Zhou et al., 2020).

Distillation is a prominent, practical method for compression that is widely used in NLP. Work on distillation in NLP has focused largely on the task-agnostic setting of compressing general-purpose pre-trained models such as BERT (Sanh et al., 2019; Sun et al., 2019, 2020) but task-specific distillation has also been reported to work well (Jiao et al., 2020).

Approaches for model quantization can be categorized into post-training quantization, where general-purpose models are quantized at test-time (Jacob et al., 2018; Bhandare et al., 2019; Kim et al., 2021), and quantization-aware training, where models incorporate simulated quantization error during training in order to learn more quantizable parameters (Zafrir et al., 2019; Bai et al., 2021). Quantization-aware training tends to lead to higher accuracy quantized inference, but post-training quantization can be applied on-the-fly to any model at inference time.

In this work we experiment with a variety of compression methods including structured and unstructured pruning (Xia et al., 2022; Chen et al., 2020), and out-of-the-box INT8 post-training dynamic quantization,[7] highlighting both practical (structured pruning, quantization) and theoretical (unstructured pruning) findings. While we do not experiment with distillation directly, our chosen structured pruning method also incorporates a distillation objective.

Learning compressible models Most closely related to our work are methods for learning compressible models, and the study of what makes models more compressible. Quantization-aware training is an example of such training for compressibility, and training for sparsity using ℓ_0 regularization (Louizos et al., 2018) is a parallel method for pruning.

Learning sparse models from scratch has proven difficult, despite the fact that deep neural networks are vastly over-parameterized. Frankle and Carbin (2019) formalized the *Lottery Ticket Hypothesis*, which posits that large, overparameterized neural network models contain sparse subnetworks, or *winning tickets*, that can be trained from scratch (or close to it; see Frankle et al. (2020)) to match the end-task performance of the full model. This influential work has spurred much research into better understanding neural network models, including pre-trained language models, from the perspective of winning tickets (Chen et al., 2020; Renda et al., 2020b; Diffenderfer and Kailkhura, 2021) and how to leverage winning tickets to perform better model compression (Chen et al., 2021; Liang et al., 2021). Li et al. (2020) showed that larger pre-trained language models are more compressible than smaller ones, which they hypothesize is related to larger models being more likely to contain winning tickets.

[7] https://pytorch.org/tutorials/recipes/recipes/dynamic_quantization.html

Flat minima in neural networks. Hochreiter and Schmidhuber (1997) were among the first to discuss the relationship between flat basins in the loss landscape and generalization in neural networks, defining a flat minimum as "a large connected region in weight space where the error remains approximately constant." We use the ϵ-sharpness definition of Keskar et al. (2017) which defines sharpness as maximum loss within a neighborhood bounded by ϵ. Others have used Hessian-based measures to identify high minima with high curvature (Chaudhari et al., 2017). Note that care is needed when making inferences based on current measurements of sharpness, which is an active area of research; it has been shown that flat minima defined in this way can be rescaled to sharp minima (and still generalize) (Dinh et al., 2017).

Most previous results related to flat minima have focused on generalizability (Hao et al., 2019; Neyshabur et al., 2020). A common explanation for the good generalization of models converged to flat minima is that flatter models are less complex (Wu et al., 2017).

Flat minima may be particularly of interest in NLP, where a pretrain-then-finetune paradigm is often employed to leverage general representations learned from an extensive pre-training process during a much shorter fine-tuning process on a more specific end task. Indeed, pre-training provides a flat prior that can provide benefits in the contexts of lifelong learning (Mehta et al., 2021) and generalization (Hao et al., 2019; Bahri et al., 2022).

A.2 Flat Minima: Additional details

Stochastic Weight Averaging (SWA). We consider last 50% of the model checkpoints for equal averaging. Specifically, for RTE, MRPC, STS-B, and CoLA we fine-tune for 10 epochs and average 5 checkpoints from epochs 6 to 10. For SST-2, QNLI, QQP, and MNLI we fine-tune for 3 epochs and retain checkpoints after every 0.5 epochs and equal average checkpoints after $2, 2.5, 3$ epochs. Izmailov et al. (2018) suggests modified learning rate scheduler like cyclical or constant so that towards the later stage of training, the underlying optimizer explores diverse solutions and average over them would lead to flatter solution. To simulate this behavior, for all our SWA experiments, we set our initial learning rate to a high value of $8e-5$ and linearly decay it to 0 with no warmup steps.

Evaluating sharpness. Keskar et al. (2017) propose a computationally feasible metric for measuring the sharpness of a minimizer over an ϵ-neighborhood in the loss landscape. We report sharpness metrics (lower value means flatter low loss region) in Tables 3 and 4. Overall we see that SAM and SWA optimized models have significantly smaller sharpness values (or flatter low loss regions) as compared to vanilla Adam optimizer, thus, providing convincing evidence that these methods indeed find flatter solutions.

Loss contours. In Figure 6, we visualize contour plots of the loss landscape for the QQP task to qualitatively compare the sharpness of the solutions that Adam- and SAM-optimized BERT$_{base}$ models find. We observe that the SAM-optimized model sits in a noticeably flatter, wider basin than the Adam-optimized model when both are fitted with their respective classifier heads. These analyses verify that SAM indeed leads to flatter minima in comparison to Adam.

A.3 Non-Iterative Unstructured Magnitude Pruning

In a preliminary analysis, we subject full-size fine-tuned BERT$_{base}$ models to one-shot unstructured magnitude pruning and evaluate on the same task *without any subsequent training*. Figure 7 displays development set accuracies at sparsity levels of increments of 5%, up to 60% of prunable parameters masked to 0. Accuracy values are plotted at averages over $n = 3$ seeds for each of the sparsity levels and GLUE tasks displayed (SST-2, QNLI, MRPC, RTE). Interestingly, even under this non-iterative pruning setting, performance does not drop off noticeably in either SAM or Adam-optimized models until at least around 30% sparsity for the GLUE tasks displayed. Similarly to all iterative pruning settings we explore, models optimized with SAM retain full model size accuracies at higher sparsity levels than their Adam-optimized counterparts. The SAM-optimized RTE models at $35-40\%$ sparsity have *higher* accuracy than the full-sized uncompressed model, reminiscent of the pattern we observe in Figure 2a, albeit at lower sparsity levels.

A.4 Iterative Magnitude Pruning Reproducibility and Hyperparameters

Following Chen et al. (2020), we use a maximum sequence length of 128, batch size of 32, learning rate to $2e-5$, and linear decay of learning rate from

epsilon (ϵ)	Dataset	MNLI	QQP	STS-B	QNLI	MRPC	RTE	SST-2	CoLA
5×10^{-3}	Adam	$28.3_{3.6}$	$34.7_{5.0}$	$160.9_{34.3}$	$30.1_{6.0}$	$50.8_{24.8}$	$49.0_{5.6}$	$29.9_{10.3}$	$38.3_{3.7}$
	SAM	$14.2_{0.9}$	$9.3_{1.7}$	$45.8_{1.3}$	$17.8_{3.4}$	$40.4_{5.3}$	$28.4_{8.7}$	$13.7_{2.3}$	$29.4_{9.8}$
1×10^{-3}	Adam	$5.0_{0.6}$	$6.5_{0.9}$	$11.8_{3.1}$	$6.6_{2.5}$	$6.1_{1.0}$	$11.9_{3.0}$	$4.5_{0.6}$	$9.5_{2.2}$
	SAM	$2.6_{0.2}$	$1.9_{0.3}$	$4.5_{0.4}$	$3.5_{1.3}$	$7.0_{0.7}$	$4.3_{2.4}$	$2.2_{0.1}$	$6.8_{2.8}$
5×10^{-4}	Adam	$2.3_{0.2}$	$3.4_{0.2}$	$4.3_{1.2}$	$3.2_{1.5}$	$2.8_{0.3}$	$6.5_{2.1}$	$2.3_{0.4}$	$5.6_{2.0}$
	SAM	$1.3_{0.1}$	$0.9_{0.1}$	$1.9_{0.3}$	$1.5_{0.3}$	$3.2_{0.6}$	$2.1_{1.1}$	$1.0_{0.1}$	$3.4_{1.4}$

Table 3: Evaluating sharpness metric for vanilla Adam and SAM optimized models at full size (i.e., no compression). We observe that SAM optimized models have significantly lower sharpness values (lower corresponds to flatter minima) compared to vanilla Adam. These results provide quantitative evidence that SAM indeed leads to flatter loss basins.

epsilon (ϵ)	Dataset Sparsity	MNLI 70%	QQP 90%	STS-B 50%	QNLI 70%	MRPC 50%	RTE 60%	SST-2 60%	CoLA 50%
5×10^{-3}	Adam	$56.4_{1.0}$	$137.9_{40.4}$	$232.5_{30.0}$	$23.8_{3.9}$	$42.3_{26.9}$	$65.7_{8.8}$	$85.8_{33.8}$	$33.8_{2.7}$
	SAM	$42.4_{7.9}$	$65.3_{1.9}$	$184.0_{7.3}$	$17.2_{6.0}$	$47.3_{18.0}$	$50.8_{6.1}$	$26.9_{7.7}$	$23.9_{4.5}$
	SWA	$37.3_{9.9}$	$34.7_{1.2}$	$81.3_{22.3}$	$33.5_{7.5}$	$31.6_{6.6}$	$42.1_{12.3}$	$23.7_{10.3}$	$26.5_{9.5}$
1×10^{-3}	Adam	$6.6_{0.9}$	$5.8_{1.0}$	$20.2_{6.1}$	$5.9_{2.2}$	$8.6_{2.2}$	$16.4_{4.0}$	$5.8_{0.7}$	$6.2_{1.4}$
	SAM	$3.1_{0.4}$	$5.4_{0.6}$	$13.8_{1.2}$	$3.2_{1.1}$	$8.3_{1.6}$	$12.0_{3.8}$	$4.1_{0.8}$	$4.2_{0.3}$
	SWA	$10.7_{1.6}$	$5.2_{0.6}$	$5.4_{1.1}$	$8.9_{0.2}$	$12.4_{0.8}$	$11.8_{4.1}$	$5.2_{0.7}$	$7.9_{2.2}$
5×10^{-4}	Adam	$3.3_{0.2}$	$2.3_{0.3}$	$7.1_{2.5}$	$2.1_{0.6}$	$5.8_{0.5}$	$7.9_{1.1}$	$3.6_{0.9}$	$3.9_{0.6}$
	SAM	$1.1_{0.3}$	$2.0_{0.6}$	$5.1_{0.6}$	$1.2_{0.0}$	$3.6_{0.6}$	$5.9_{2.0}$	$1.9_{0.4}$	$2.3_{0.4}$
	SWA	$4.4_{0.5}$	$2.9_{0.0}$	$2.2_{0.4}$	$4.7_{0.1}$	$5.9_{0.7}$	$6.0_{2.0}$	$2.5_{0.5}$	$3.4_{0.2}$

Table 4: Evaluating sharpness metric for vanilla Adam, SAM and SWA optimized models at Chen et al. (2020)'s reference sparsities. In general we observe that SAM and SWA optimized models have lower sharpness values (lower corresponds to flatter minima) compared to vanilla Adam at various sparsity levels across different tasks (all results are averaged over 3 runs).

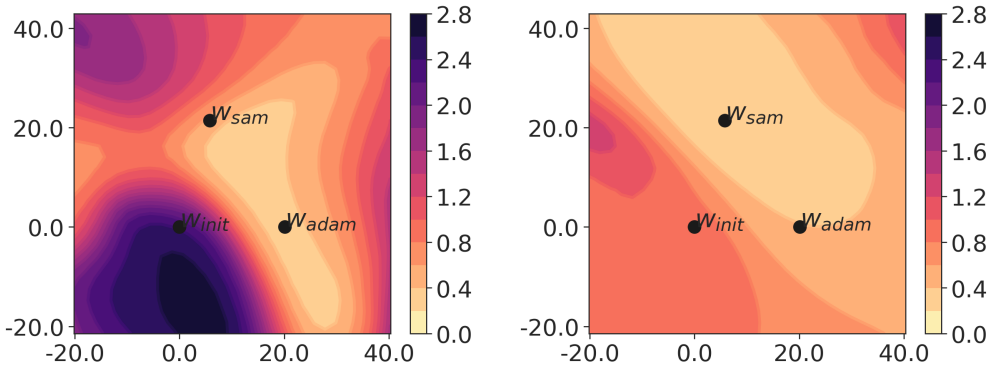


Figure 6: Visualization of loss contours for QQP on BERT_{base} models finetuned on the task using Adam and SAM optimizers (w_{adam} and w_{sam}), as well as the pre-trained BERT base initialization (w_{init}). On the left, all models are fitted with the linear classifier head originally trained with w_{adam}, while on the right side models are fitted with the linear classifier head originally trained with w_{sam}. The SAM-optimized model sits in a noticeably flatter, wider basin than the Adam-optimized model when both are fitted with their respective classifier heads. These results provide qualitative evidence that SAM indeed leads to flatter loss basins.

initial value to zero with no warmup period. For tasks with smaller datasets (RTE, MRPC, CoLA, STS-B), we fine-tune models for 10 epochs, evaluate them after every epoch and retain the checkpoint yielding best task-specific performance on the hold-out validation set (whereas Chen et al. (2020) finetune for only 3 for all tasks). For tasks with comparatively larger datasets (MNLI, QQP, QNLI, SST-2), we fine-tune models for 3 epochs. We set Adam's weight decay to $\epsilon = 0$ in order to remove the potential confound of regularization on models' amenability to magnitude pruning. This

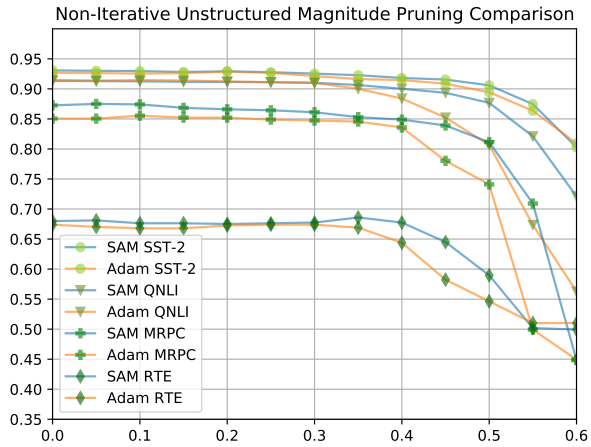


Figure 7: SAM- and Adam-optimized models evaluated directly after pruning a proportion of parameters from the full-sized model. Models fine-tuned with SAM hold up better to non-iterative magnitude pruning as well.

differs from Chen et al. (2020)'s $\epsilon = 1 \times 10^{-8}$, but we observed that our differences do not systematically affect the trends originally reported other than to improve full-size model performance and allow for a fairer comparison between SAM and Adam optimizers. In particular, training for only 3 epochs on the smaller tasks with our SAM optimizer does not allow models to converge.

A.5 Comparison with BERT Lottery Ticket Hypothesis Numbers

We present development set numbers on GLUE and SQuAD tasks in Table 5. For comparison, we include the reference (*Ref*) metrics reported by Chen et al. (2020) at their reported winning ticket sparsity levels.

A.6 Additional Individual Task Plots for BERT$_{base}$ IMP

In Figure 8, we present the SQuAD plot comparing SAM- and Adam-optimized BERT$_{base}$ models in the unstructured IMP setting in Figure 2, as well as versions of the GLUE plots in including SWA performance.

Plots for BERT$_{base}$ trained on GLUE and SQuAD tasks with unstructured *standard pruning* are shown in Figure 9.

A.7 Is SAM just implicitly doing ℓ_1 regularization?

No. It is clear that ℓ_1 regularization induces sparsity in a different way compared to SAM. Although in some cases ℓ_1 regularization can help a model reach higher accuracies at certain sparsity levels, we observe that simply optimizing with Adam throughout an iterative pruning process does not allow the model to reach SAM-optimized models' compression performance. Moreover, ℓ_1 regularization can actually hurt compression performance in some cases.

Further investigation is needed to understand the specific mechanisms allowing SAM to induce greater compresseibility in models.

A.8 Detailed Structure vs Optimization Results

Table 6 contains numbers presented in Figure 3 from §3.4.2.

Figure 12 presents the same information as Figure 3 in an alternative view, featuring colored bars representing ticket performance over different optimizers.

A.9 Ticket Transfer Experiments: Comparing Optimization Over Given Tickets

Figure 4 in §3.4.3 allows us to evaluate SAM vs. vanilla Adam *ticket* transferability across GLUE tasks. Figure 13, on the other hand, allows us to evaluate SAM vs. vanilla Adam *optimization* over given tickets for different GLUE tasks.

A.10 Structured Pruning Reproducibility and Hyperparameters

Following Xia et al. (2022), we train for 20 epochs each in the pruning and final fine-tuning stages. We use a sparsity epsilon value of 0.01, meaning that a model can be accepted if its actual sparsity level is within 1% of the target sparsity level of 95%.

We pick hyperparameters based on a grid search over Xia et al. (2022)'s baseline implementation (with their Adam-optimized teacher models), reported in Table 7.

For each task, we use the optimal λ and final fine-tuning learning rates found via grid search using Xia et al. (2022)'s implementation, including configurations for finetuning teacher models (which used normal Adam optimizers). There are small discrepancies between the reference metric values reported and the values we were able to reproduce, possibly due to variation across random seeds. However, the relative performance of hyperparameter settings seems to be fairly consistent across random seeds.

Although we do not conduct an additional full grid search for our comparison of compressed mod-

	Dataset Sparsity Metric	MNLI 70% Matched acc.	QQP 90% Acc.	STS-B 50% Pearson Cor.	QNLI 70% Acc.	MRPC 50% Acc.	RTE 60% Acc.	SST-2 60% Acc.	CoLA 50% Matthew's Cor.	SQuAD 40% F1
Full FT	*Ref*	$82.4_{0.5}$	$90.2_{0.5}$	$88.4_{0.3}$	$89.1_{1.0}$	$85.2_{0.1}$	$66.2_{3.6}$	$92.1_{0.1}$	$54.5_{0.4}$	$88.1_{0.6}$
	Adam	$84.7_{0.4}$	$91.0_{0.1}$	$89.0_{0.1}$	$91.5_{0.1}$	$85.5_{1.7}$	$67.6_{1.5}$	$92.7_{0.1}$	$58.1_{0.4}$	$88.5_{0.2}$
	SAM	$85.3_{0.2}$	$91.1_{0.1}$	$89.4_{0.2}$	$91.3_{0.6}$	$87.0_{0.7}$	$67.0_{0.8}$	$93.1_{0.6}$	$59.5_{0.4}$	$89.2_{0.1}$
IMP	*Ref*	$82.6_{0.2}$	$90.0_{0.2}$	$88.2_{0.2}$	$88.9_{0.4}$	$84.9_{0.4}$	$66.0_{2.4}$	$91.9_{0.5}$	$53.8_{0.9}$	$87.7_{0.5}$
	Adam	$82.6_{0.3}$	$85.0_{0.2}$	$88.3_{0.2}$	$88.6_{0.1}$	$84.2_{0.1}$	$63.7_{0.9}$	$91.7_{0.4}$	$56.4_{1.4}$	$86.9_{0.3}$
	SAM	$\mathbf{83.5_{0.1}}$	$84.9_{0.1}$	$\mathbf{88.9_{0.2}}$	$\mathbf{89.6_{0.1}}$	$\mathbf{85.8_{1.0}}$	$\mathbf{71.8_{1.7}}$	$\mathbf{93.2_{0.4}}$	$56.1_{0.9}$	$\mathbf{87.8_{0.2}}$
±10%	Adam	$82.1_{0.3}$	$87.2_{0.3}$	$88.3_{0.2}$	$88.0_{0.2}$	$83.7_{0.4}$	$64.2_{0.2}$	$91.5_{0.5}$	$55.0_{0.4}$	$86.8_{0.4}$
	SAM	$\mathbf{83.1_{0.1}}$	$87.4_{0.4}$	$\mathbf{88.8_{0.2}}$	$\mathbf{89.1_{0.2}}$	$\mathbf{85.5_{0.5}}$	$\mathbf{68.9_{0.4}}$	$\mathbf{92.9_{0.3}}$	$56.1_{0.2}$	$\mathbf{88.3_{0.5}}$
Std	*Ref*	82.1	90.0	88.5	89.9	85.8	63.0	90.0	52.0	87.1
	Adam	82.7	88.1	89.2	89.5	85.5	65.3	91.1	54.2	$86.3_{0.2}$
	SAM	**83.3**	**89.6**	**89.6**	**90.0**	85.3	**68.2**	**92.1**	**56.8**	$\mathbf{87.1_{0.2}}$

Table 5: We report task metrics on the development set at Chen et al. (2020)'s reference sparsities for Adam and SAM-optimized BERT-base models in their (1) Iterative Magnitude Pruning (**IMP**) and (2) **Std** pruning settings. We include Chen et al. (2020)'s reference (*Ref*) metrics in addition to our reported metrics (Adam, SAM). When applicable, we report mean and standard deviation calculated over 3 random seeds.

Dataset	Ticket	Optim.	Accuracy
RTE (60%)	Random	Adam	$54.9_{1.3}$
		SAM	$55.4_{1.6}$
	Adam	Adam	$63.7_{0.9}$
		SAM	$61.7_{2.4}$
	SAM	Adam	$70.2_{1.9}$
		SAM	$71.8_{1.7}$
MRPC (50%)	Random	Adam	$70.8_{0.8}$
		SAM	$70.1_{0.2}$
	Adam	Adam	$84.2_{0.1}$
		SAM	$85.3_{0.5}$
	SAM	Adam	$85.3_{1.3}$
		SAM	$85.7_{0.8}$
SST-2 (60%)	Random	Adam	$82.8_{0.2}$
		SAM	$83.3_{0.8}$
	Adam	Adam	$91.9_{0.3}$
		SAM	$92.7_{0.1}$
	SAM	Adam	$92.4_{0.1}$
		SAM	$92.9_{0.2}$
QNLI (70%)	Random	Adam	$61.7_{0.3}$
		SAM	$61.5_{0.1}$
	Adam	Adam	$89.0_{0.05}$
		SAM	$89.5_{0.04}$
	SAM	Adam	$89.1_{0.1}$
		SAM	$89.6_{0.2}$

Table 6: For RTE, MRPC, SST-2, and QNLI 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.

els using Adam and SAM-optimized teacher models, we do find that the optimal final fine-tuning learning rates, which are much less computationally expensive to test, transfer to our experimental settings.

A.11 Detailed Quantization Results

Table 8 contains the numbers used to generate Figure 5.

Dataset		λ	FT-LR	Teacher Acc.	Pruned Acc.
SST-2 (67k)	*Ref.*	-	-	93.1	90.6
	Reprod.	0.9	3e−5	93.6	90.6
QNLI (105k)	*Ref.*	-	-	91.5	86.1
	Reprod.	0.9	3e−5	91.9	86.5
QQP (364k)	*Ref.*	-	-	91.2	90.1
	Reprod.	0.7	3e−5	91.3	89.9
MNLI (393k)	*Ref.*	-	-	84.8	80.6
	Reprod.	0.7	3e−5	85.2	80.1

Table 7: We report reproduced (*Reprod.*) and reference (*Ref.*) evaluation metrics at 95% sparsity and optimal values for λ and fine-tuning learning rate on select tasks from Xia et al. (2022)'s structured pruning setting. We used the same hyperparameters as reported otherwise (distillation temperature $t = 2$, with 20 fine-tuning epochs after pruning, learning rate $= 2e-5$, and batch size$= 32$), and conducted our grid search over the same candidate values $\lambda \in \{0.1, 0.3, 0.5\}$ and FT-LR $\in \{1e-5, 2e-5, 3e-5\}$

A.12 Results for Other BERT Models

We investigate SAM's influence on amenability to sparsification in both BERT$_{large}$ and RoBERTa$_{base}$ models subject to iterative magnitude pruning (IMP). For consistency, we use the same hyperparameters as for the BERT$_{base}$ set of experiments ($\epsilon = 0$ weight decay; 10 (MRPC, RTE), 3 (SST-2, QNLI), or 2 (SQuAD) training epochs for each IMP iteration; linear learning rate decay schedules starting at $2e-5$ (GLUE) or $3e-5$ (SQuAD); batch size of 32 (GLUE) and 16 (SQuAD); maximum sequence length of 128 (GLUE) and 384 (SQuAD)). It is possible that a different set of hyperparameters would be optimal for these different models, but we also tried different numbers of training epochs for the less stable smaller GLUE tasks (5 and 3), as

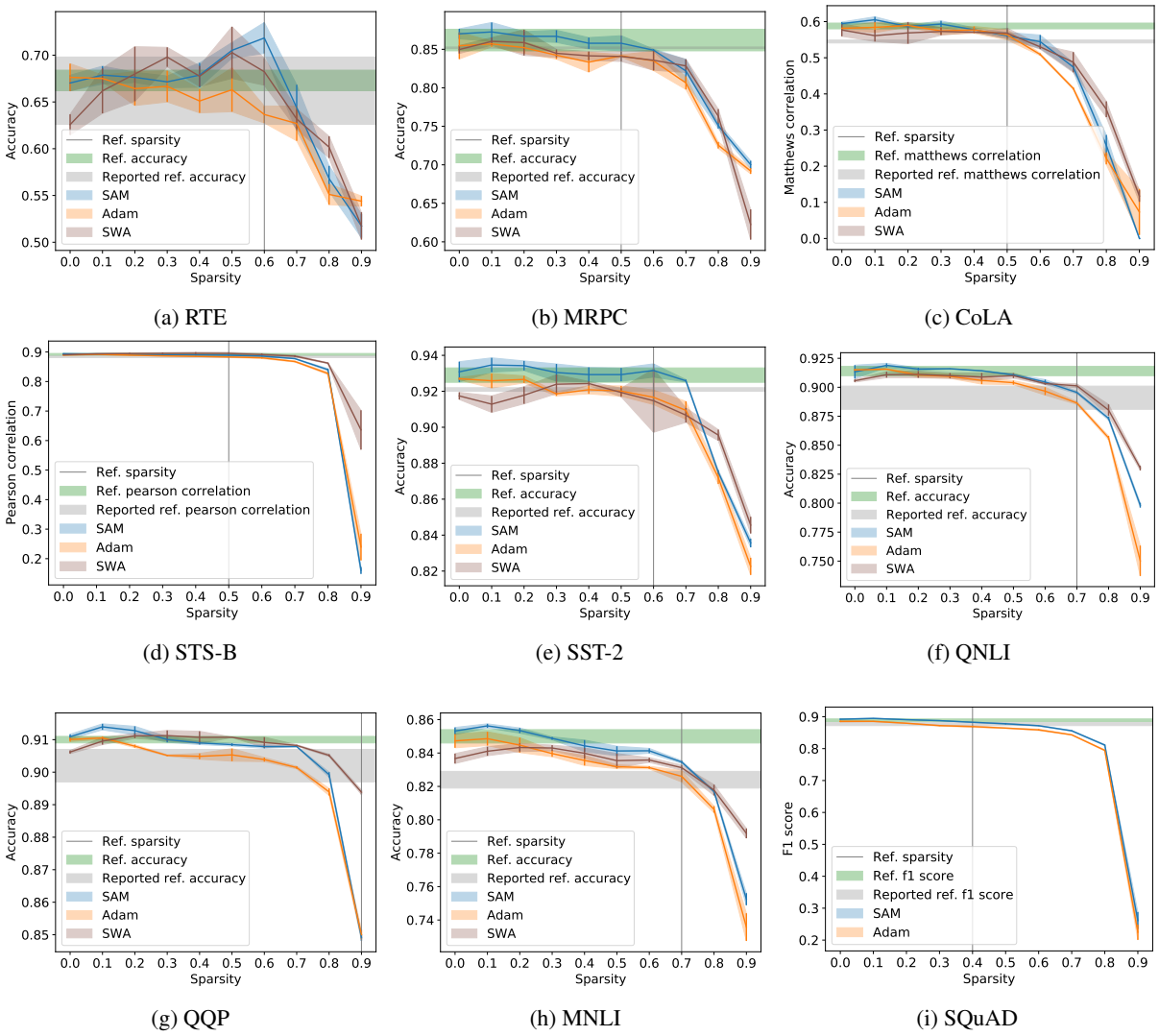
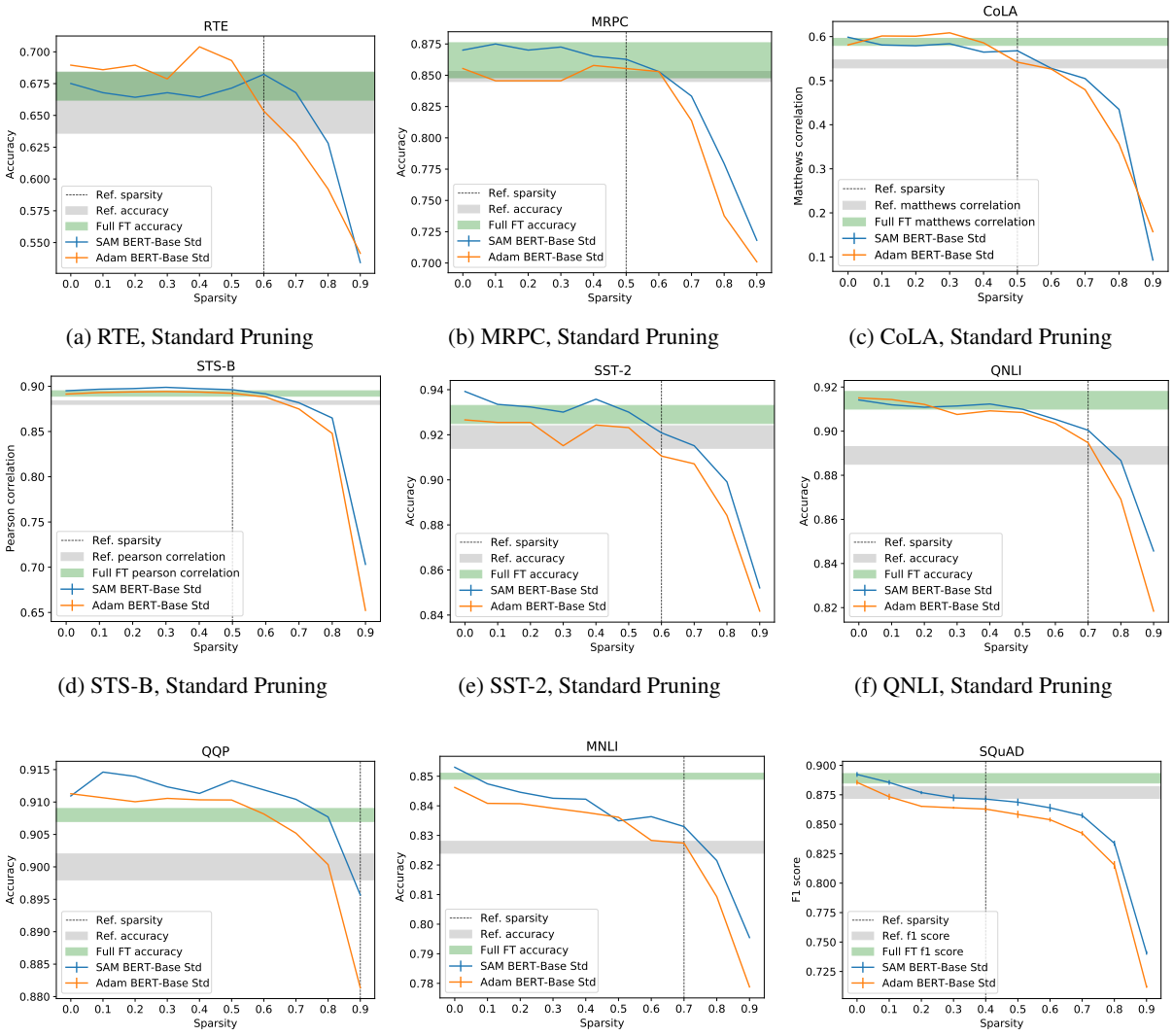


Figure 8: Individual plots showing sparsity vs. task metrics (validation set) for GLUE throughout IMP. The vertical lines and gray horizontal bands mark reference sparsity and "winning ticket" evaluation metric values that were obtained by Chen et al. (2020). The green horizontal bands mark the initial performance of our full fine-tuned (uncompressed) models.

well as Liu et al. (2019)'s learning rate of $1.5e-5$ for RoBERTa, and we generally found that simply matching hyperparameters from BERT$_{base}$ experiments worked well or better.

Figures 14 and 15 show plots for BERT$_{large}$ and RoBERTa$_{base}$ models compressed with iterative magnitude pruning (IMP). We include our BERT$_{base}$ model results with Chen et al. (2020)'s reference sparsity levels and accuracy ranges (which are likewise for BERT$_{base}$) in the same plots for comparison.

With the exception of RoBERTa$_{base}$ on MRPC, SAM-optimized models consistently fare better than Adam-optimized models in these other BERT variants as well. However, the comparison between BERT variants is more complex. While BERT$_{large}$ and RoBERTa$_{base}$ models generally achieve higher initial performance compared to BERT$_{base}$ and can maintain this higher performance at Chen et al. (2020)'s "winning ticket" sparsity levels (14b, 14d, 14e, 15b, 15e), the drop-off in performance does not always simply follow a parallel pattern. Initial higher performance tends to decrease more quickly with pruning than in BERT$_{base}$ models, such that BERT$_{large}$ and RoBERTa$_{base}$ performance sometimes falls to near (14c, 15c, 15d) or even *below* (14a, 15a) BERT$_{base}$ performance by the time they approach "winning ticket" sparsity levels (which in reality provide an inherent advantage to the larger models that are left with a greater absolute number



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

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

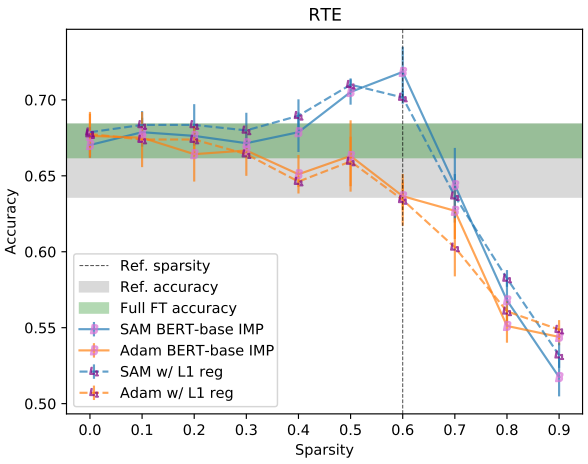
(g) QQP, Standard Pruning (h) MNLI, Standard Pruning (i) SQuAD, Standard Pruning

Figure 9: Individual plots showing sparsity vs. accuracy for GLUE tasks and SQuAD in BERT$_{base}$ 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.

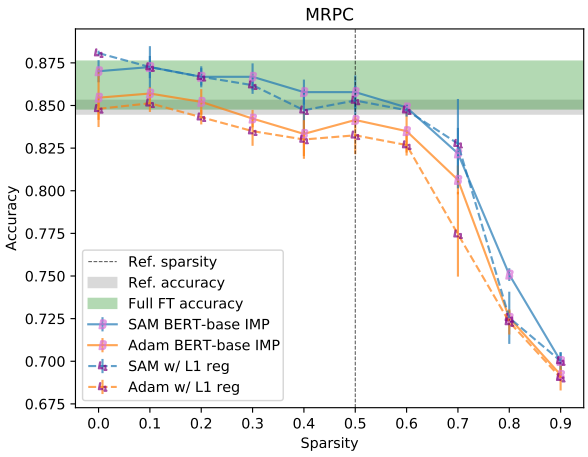
of parameters at the same sparsity levels).

A.13 Beyond task accuracy

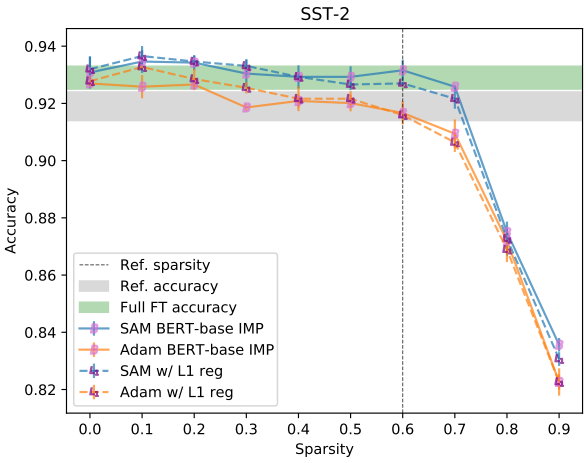
We evaluate full and pruned BERT$_{base}$ models optimized by vanilla Adam and SAM throughout IMP (with rewind) on Ribeiro et al. (2020)'s pre-curated test suites for sentiment analysis (SST-2), question paraphrase detection (QQP), and question answering (SQuAD). At this time, we do not explicitly make direct comparisons between SAM and vanilla Adam for unpruned and pruned models. A single test consists of multiple examples, and the x-axes of the histograms in Figure 16 refer to the proportions of examples passed within each test.



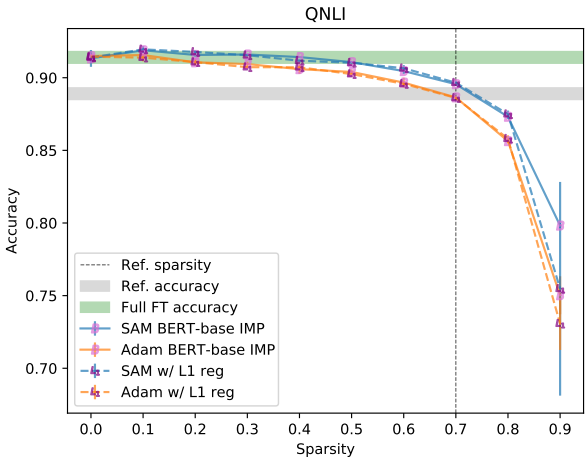
(a) RTE, IMP w/ ℓ_1 Regularization



(b) MRPC, IMP w/ ℓ_1 Regularization

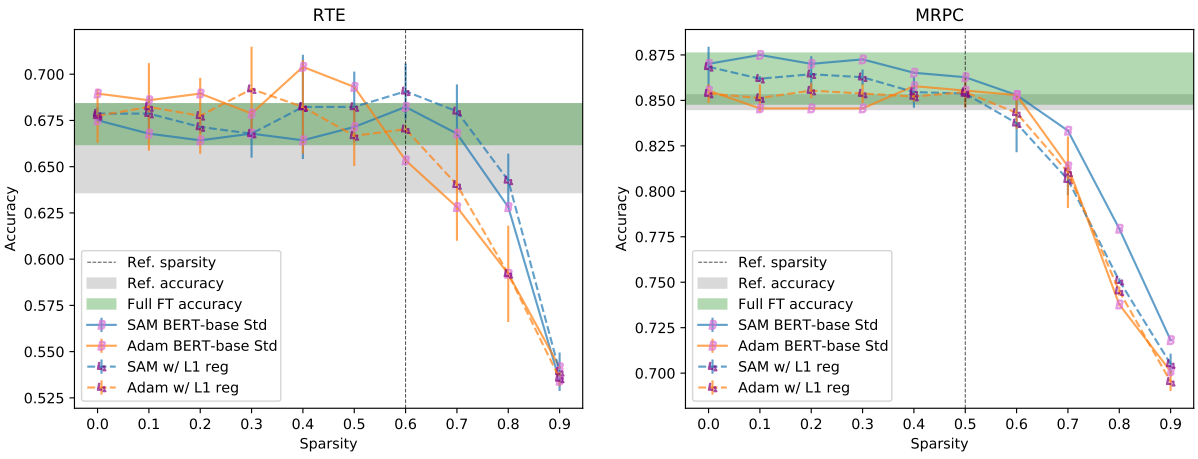


(c) SST-2, IMP w/ ℓ_1 Regularization



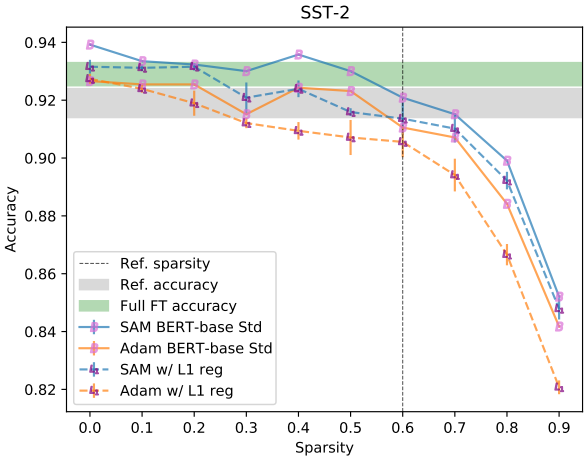
(d) QNLI, IMP w/ ℓ_1 Regularization

Figure 10: Individual plots showing sparsity vs. accuracy for GLUE tasks in ℓ_1-regularized BERT$_{base}$ models compressed with iterative magnitude pruning (IMP), with regular BERT$_{base}$ 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.

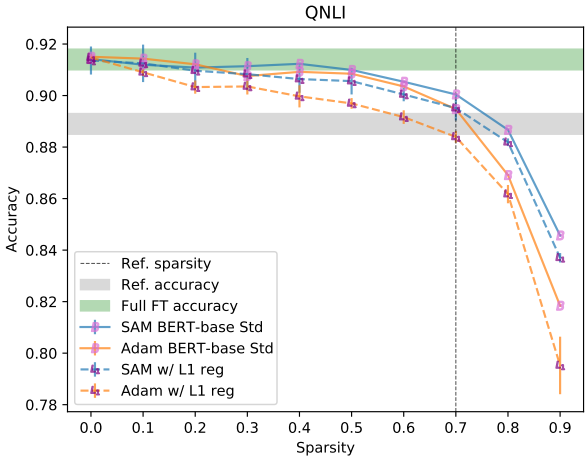


(a) RTE, Standard Pruning w/ ℓ_1 Regularization

(b) MRPC, Standard Pruning w/ ℓ_1 Regularization



(c) SST-2, Standard Pruning w/ ℓ_1 Regularization



(d) QNLI, Standard Pruning w/ ℓ_1 Regularization

Figure 11: Individual plots showing sparsity vs. accuracy for GLUE tasks and SQuAD in BERT$_{base}$ models trained with ℓ_1 regularization during iterative compression with standard pruning, with BERT$_{base}$ models trained without regularization during standard pruning 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.

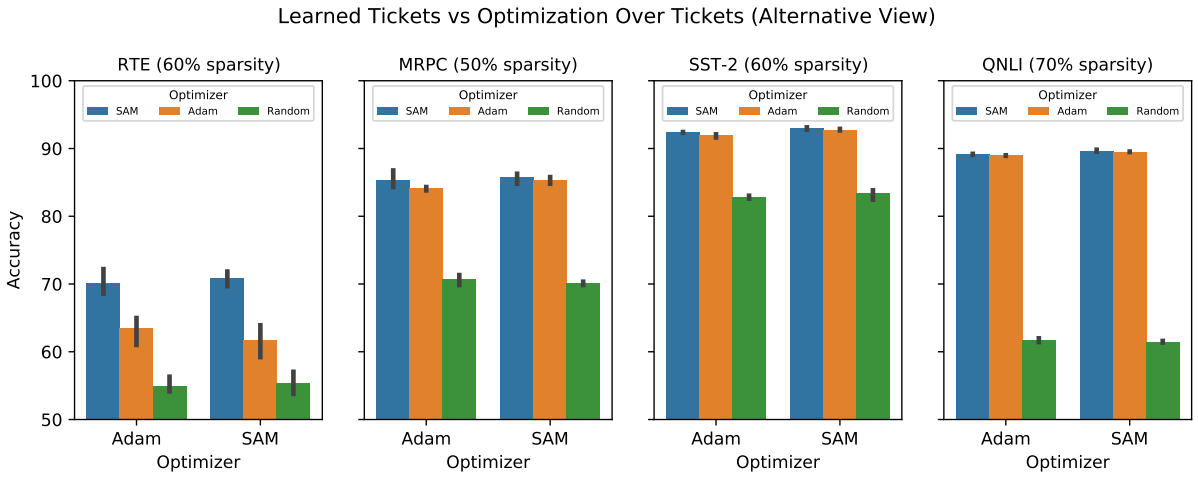


Figure 12: An alternative view of the same information contained in Figure 3. Note that, for example, in RTE, SAM tickets clearly outperform vanilla Adam tickets regardless of the optimizer used for the final fine-tuning.

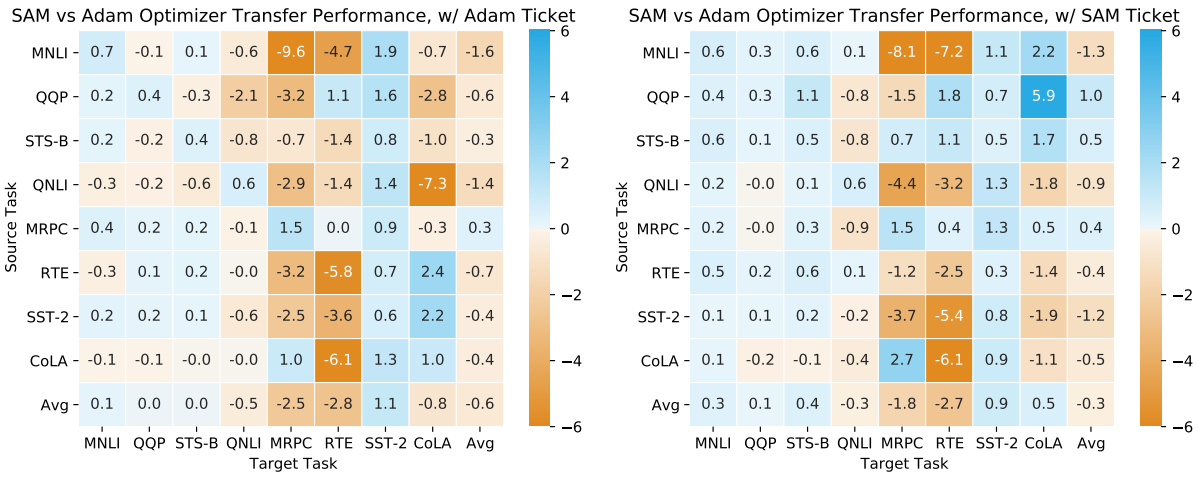
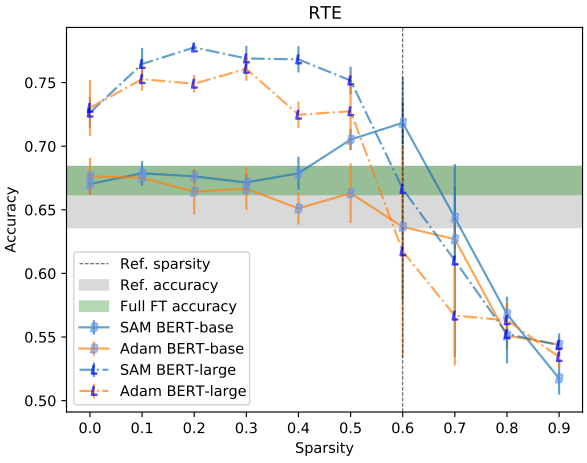
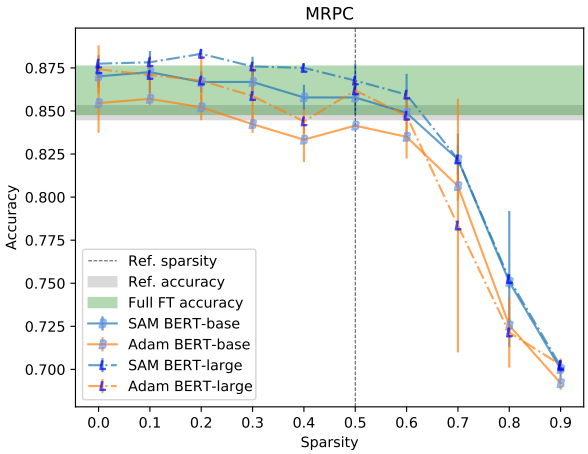


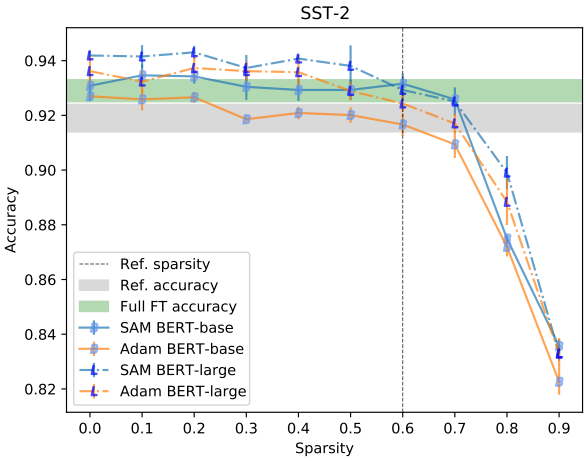
Figure 13: Heatmaps indicating the *difference* in target task performance between SAM and Adam optimizers during fine-tuning when transferring tickets across tasks. Values greater than 0 indicate the extent to which SAM optimizer worked better than Adam; Values less than 0 indicate where Adam optimizer worked better than SAM; Values close to 0 indicate little difference Note that positive values along the diagonal indicate superior SAM ticket performance in the single task setting, even with "transfer" between optimizers.



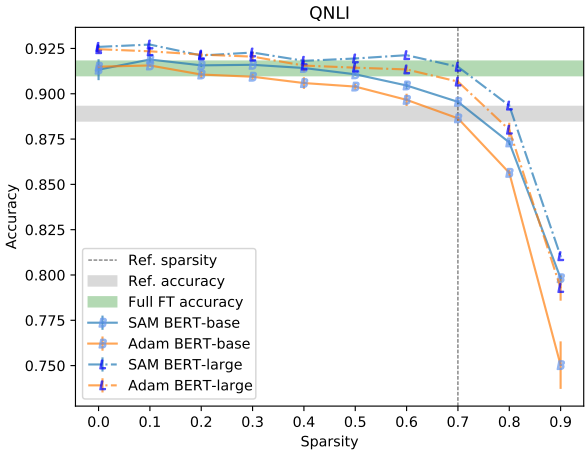
(a) RTE, IMP w/ BERT$_{large}$



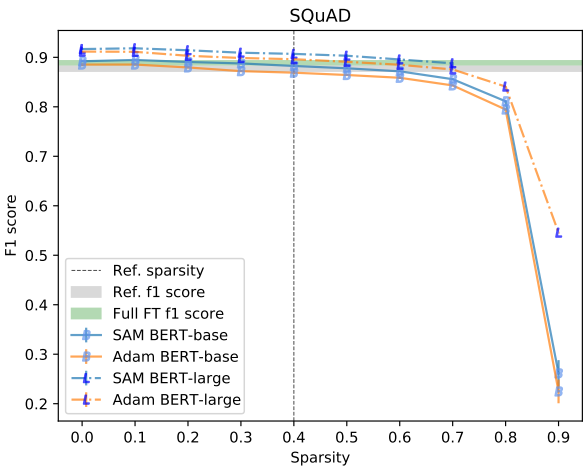
(b) MRPC, IMP w/ BERT$_{large}$



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

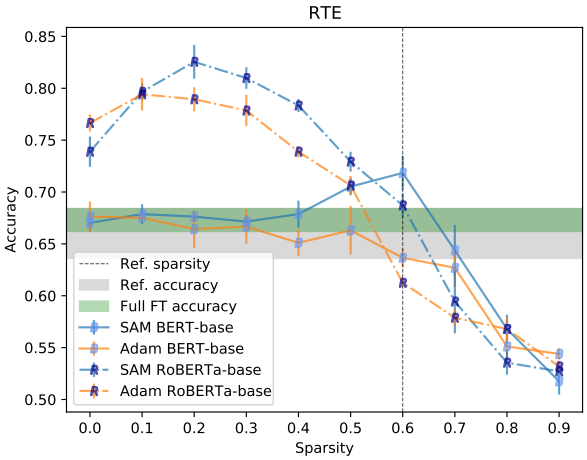


(d) QNLI, IMP w/ BERT$_{large}$

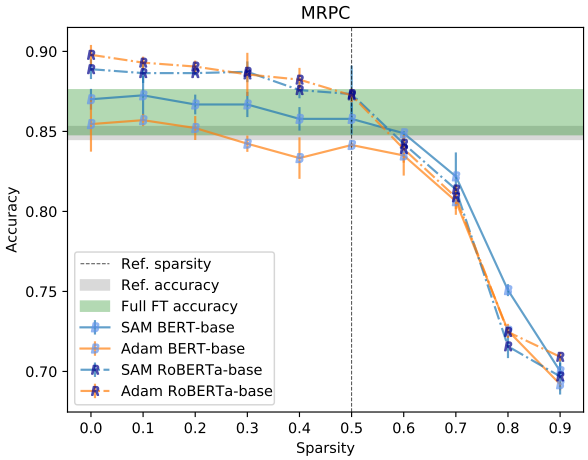


(e) SQuAD, IMP w/ BERT$_{large}$

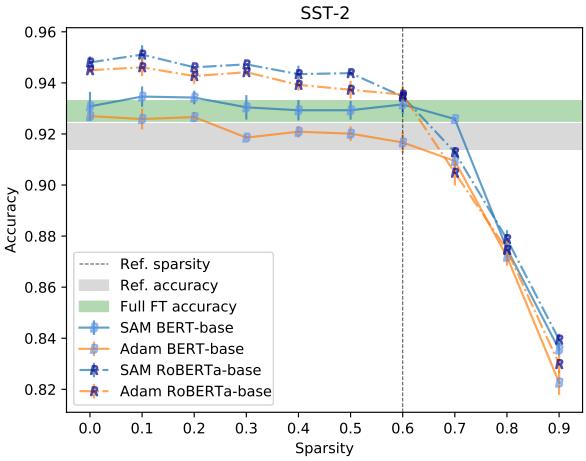
Figure 14: Individual plots showing sparsity vs. accuracy for GLUE tasks and SQuAD in BERT$_{large}$ models compressed with iterative magnitude pruning (IMP), with BERT$_{base}$ models for comparison. The vertical lines and gray horizontal bands mark reference sparsity and "winning ticket" evaluation metric values that were obtained by Chen et al. (2020). The green horizontal bands mark the initial performance of our full fine-tuned models.



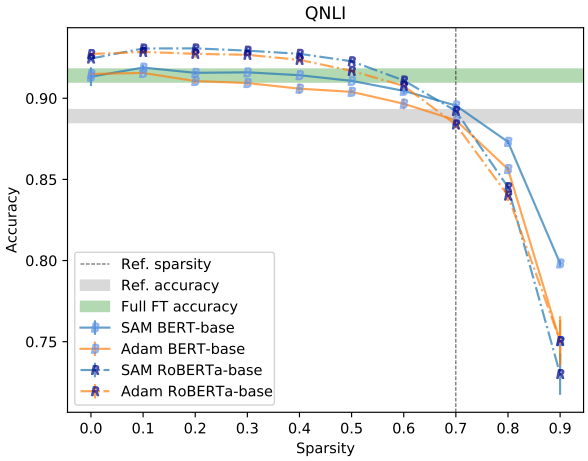
(a) RTE, IMP w/ RoBERTa$_{base}$



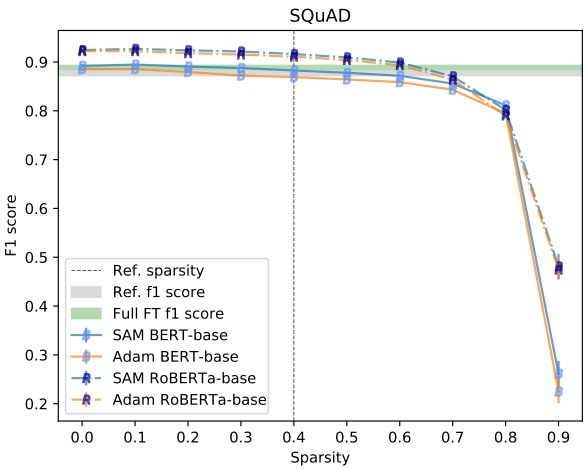
(b) MRPC, IMP w/ RoBERTa$_{base}$



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



(d) QNLI, IMP w/ RoBERTa$_{base}$



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

Figure 15: Individual plots showing sparsity vs. accuracy for GLUE tasks and SQuAD in RoBERTa$_{base}$ models compressed with iterative magnitude pruning (IMP), with BERT$_{base}$ 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.

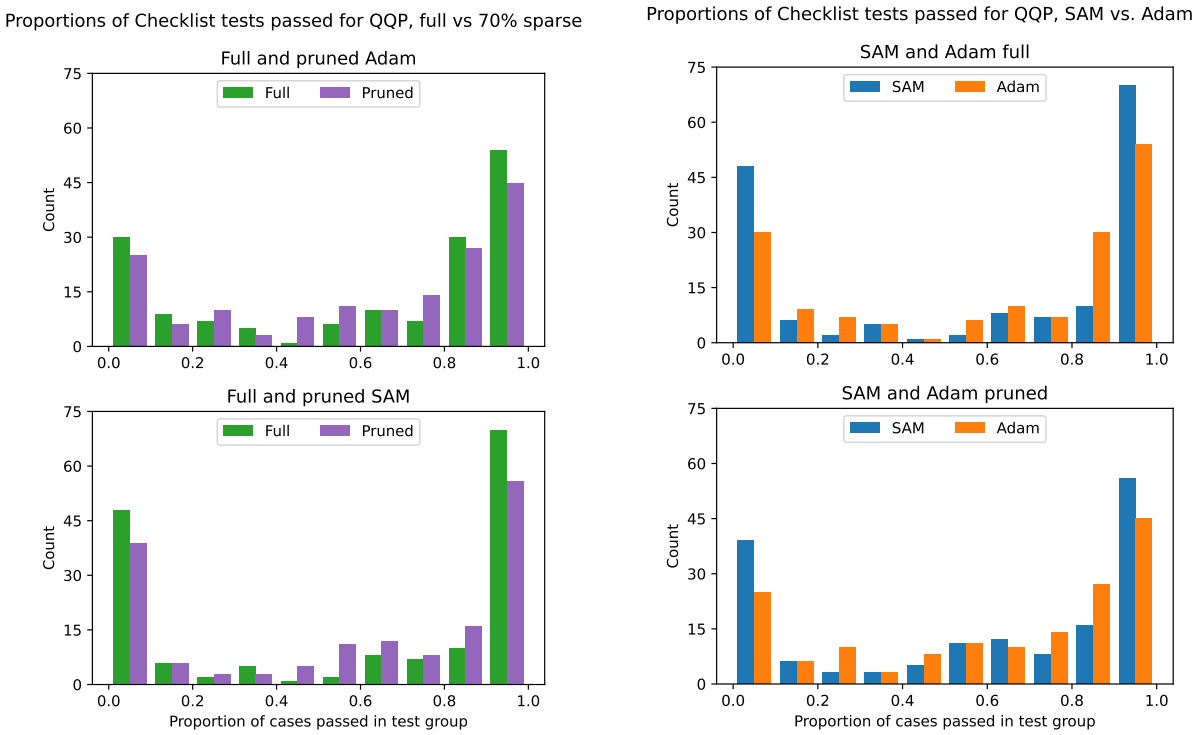


Figure 16: Aggregated Checklist (Ribeiro et al., 2020) results for QQP models. Tests target various capabilities of models such as robustness to typos and simple coference resolution. Note the concentration of frequencies near 0.0 and 1.0 for all models, as well as the shifts in frequencies when pruned for both vanilla Adam and SAM models; models can effectively *lose* or even *gain* specific capabilities throughout compression.

Dataset	QAT Ref.	Optim.	Full FT	Quantized
MNLI	N/A	Adam	$84.42_{0.37}$	$78.24_{4.00}$
Acc.		SAM	$84.68_{0.13}$	$83.47_{0.11}$
QQP	87.96	Adam	$87.98_{0.12}$	$85.35_{1.93}$
F1		SAM	$88.20_{0.08}$	$86.85_{0.28}$
STS-B	89.04	Adam	$89.06_{0.11}$	$86.54_{1.07}$
Pearson		SAM	$89.39_{0.07}$	$87.16_{0.76}$
QNLI	90.62	Adam	$91.49_{0.09}$	$89.05_{0.37}$
Acc.		SAM	$91.33_{0.58}$	$89.83_{0.54}$
MRPC	89.56	Adam	$89.57_{1.13}$	$86.81_{1.72}$
F1		SAM	$91.24_{0.20}$	$\mathbf{89.37_{0.83}}$
RTE	68.78	Adam	$67.63_{1.10}$	$56.80_{4.51}$
Acc.		SAM	$67.87_{0.36}$	$65.70_{1.65}$
SST-2	92.24	Adam	$92.70_{0.07}$	$91.17_{0.90}$
Acc.		SAM	$93.08_{0.57}$	$\mathbf{92.39_{0.40}}$
CoLA	58.48	Adam	$60.47_{0.55}$	$55.99_{2.86}$
Matt.		SAM	$59.09_{0.72}$	$54.88_{0.78}$
SQuAD	87.74	Adam	$89.20_{0.12}$	$80.13_{1.85}$
F1		SAM	$89.20_{0.12}$	$84.92_{0.55}$

Table 8: We compare full fine-tuned and quantized BERT$_{base}$ models optimized with SAM and Adam. Notably, applying a simpler post-training dynamic quantization technique on a SAM-optimized model can approach the reported (QAT ref) performance of a model quantized through quantization-aware training (Zafrir et al., 2019). These instances are **bolded**.