A. Additional Training Curves

A.1. Training Cost Using FLOPs

In Figure 10, we plot selected learning curves from the main text as a function of FLOPs rather than seconds. We compute FLOPs using the code provided by Clark et al. (2020).

A.2. The Impact of Batch Size

Figure 13 shows the learning curves associated with different batch sizes. Table 1 shows the learning rates associated with each batch size. We use the hyperparameters from Liu et al. (2019b) as a starting point and then lightly tune them.

Batch Size	Learning Rate
256	.0002
2048	.001
4096	.00125
8192	.0015
16384	.001875

Table 1. The learning rate for each batch size in Figure 13.

A.3. The Impact of Dataset Size

Figure 14 shows the learning curves for models trained using 5% and 1% of the training data.

B. Finetuning Models of Different Sizes

Table 2 shows that models with more parameters are not harder to finetune.

Model	Perplexity	MNLI	SST-2
12-layer, 768H	4.3	84.3	93.0
18-layer, 768H	4.1	85.4	92.6
24-layer, 768H	4.0	85.2	93.1
12-layer, 768H	4.3	84.3	93.0
12-layer, 1024H	3.9	85.5	93.2
12-layer, 1536H	4.3	85.1	93.8

Table 2. We train RoBERTa models of different sizes and stop them at roughly the same pretraining perplexity (the bigger models are trained for less wall-clock time). We then finetune each model on MNLI and SST-2. All models reach comparable accuracies (in fact, the big models often outperform small ones), which shows that larger models are not harder to finetune.

C. Negative Results: Layer Sharing

Sharing weights across transformer layers can provide a small or negligible degradation in final performance (Lan et al., 2020; Dehghani et al., 2019) while providing a reduction in memory consumption. In addition, models with shared layers are slightly faster to execute because they require less memory movement and reduced inter-device communication. Similar to Lan et al. (2020), we experiment with two types of layer sharing: sharing all layers and sharing only the attention layers.

Sharing layers reduces the maximum memory requirements, especially for small batch sizes. For example, sharing all the layers of a RoBERTa model with batch size 32 reduces total memory usage by 41%. However, both forms of sharing lead to slower training convergence and thus worse performance in the resource-constrained setting (Figure 11). Consequently, we do not recommend sharing layers for compute-efficient training or inference of transformers.

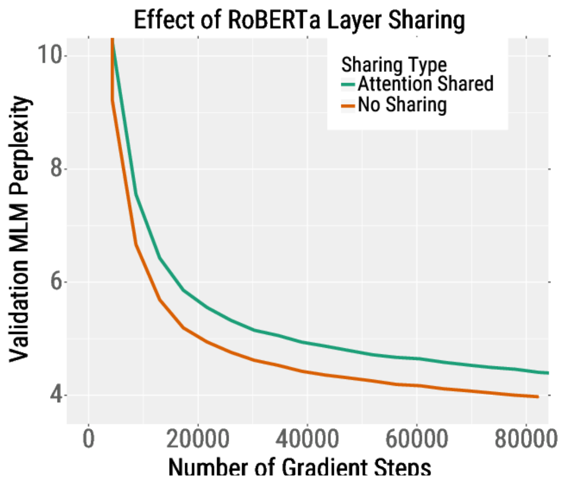

Figure 11. Sharing attention layers reduces the maximum memory consumption of RoBERTa but causes slower convergence and worse final accuracy.

D. Compression Results for SST-2

We follow Liu et al. (2019b) and report results on SST-2 (Socher et al., 2013) in addition to MNLI. Since the SST-2 dataset is smaller than MNLI it requires a more significant tuning of the finetuning hyperparameters. We tune the batch size in $\{16, 32, 64\}$, the learning rate in $\{5e{-}4, 3e{-}4, 1e{-}4\}$, the seed which controls the classifier initialization and training data shuffling in $\{100, 300, 500\}$, and the dropout in $\{0.1, 0.2, 0.3\}$. We choose the best value using the validation set for each model size. We then perform quantization, pruning, and quantization and pruning on all finetuned models. Similar to MNLI, the bigger models provide the highest accuracy for a given test budget (Figure 12).

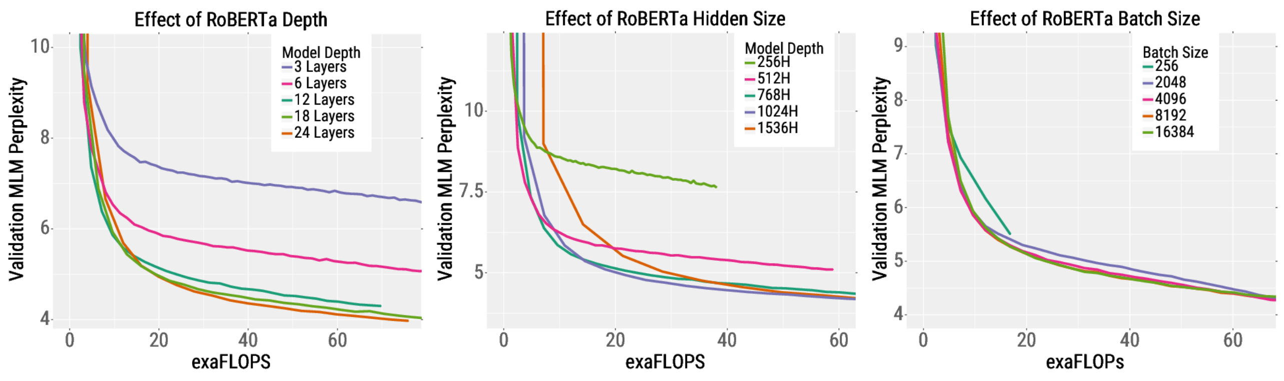

Figure 10. Floating Point Operations. We show Figures 2, 4, and 13 in terms of exaFLOPs instead of wall-clock time. Bigger models achieve better results than smaller models using the same number of floating point operations.

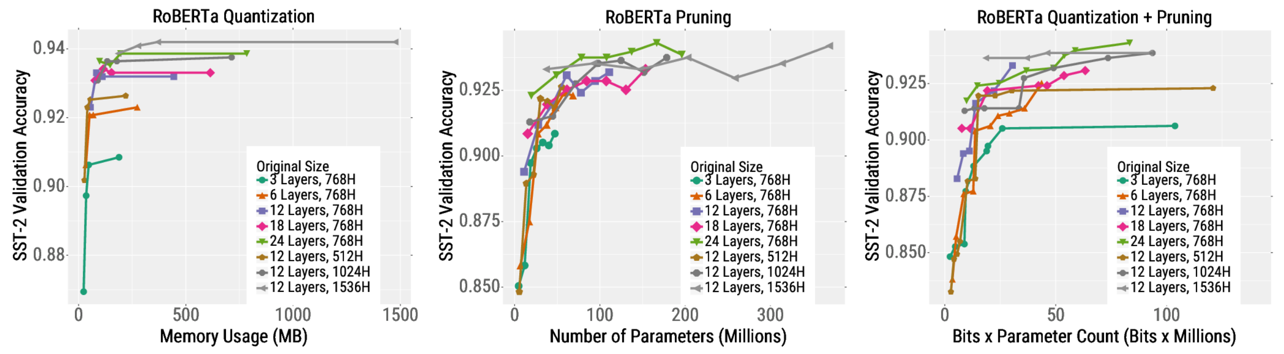

Figure 12. Compression for SST-2. For most budgets (x-axis), the highest accuracy SST-2 models are the ones which are trained large and then heavily compressed. We show results for quantization (left), pruning (center), and quantization and pruning (right).

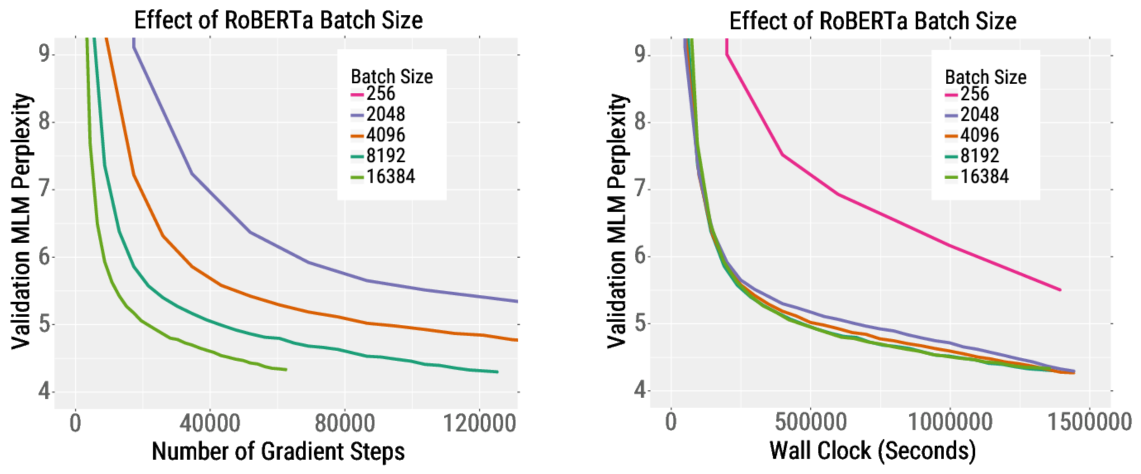

Figure 13. Increasing the batch size and the associated learning rate accelerates convergence in terms of gradient steps. However, increasing the batch size beyond 2048 provides only marginal improvements with respect to wall-clock time. Note that the wall-clock time includes the cost of accumulating gradients on a single machine (see Section 2.2). In other words, beyond a certain point increasing the batch size only provides speedups when additional hardware is available. The 256 batch size result is far to the right in the left plot.

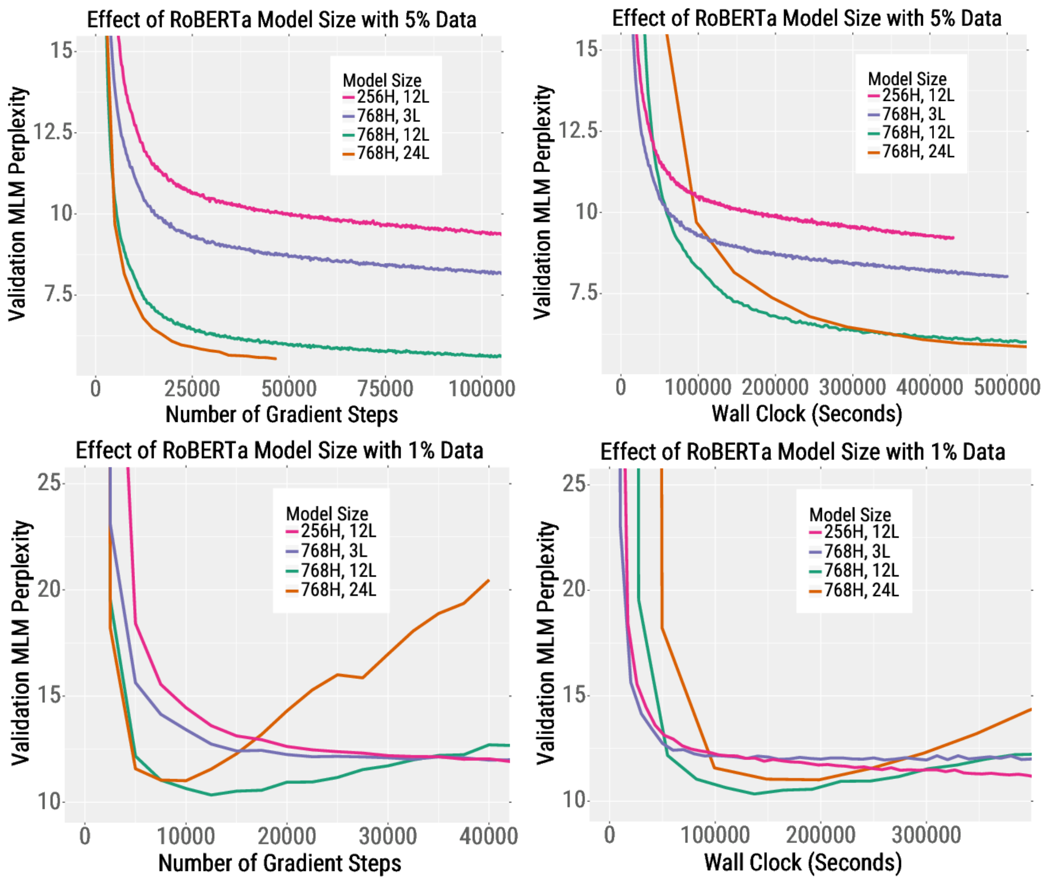

Figure 14. Effect of Smaller Datasets. In our experiments on the full dataset (see main text), the largest models we trained are always faster in terms of wall-clock time. However, when subsampling the data to 5% (top row), the biggest models do not improve on the speed of the smaller models (e.g., compare 24 Layer RoBERTa and 12 Layer RoBERTa). When the data is subsampled to 1% (bottom row), the bigger models are *worse* in terms of perplexity due to overfitting. This illustrates that the optimal model size depends on the dataset size.