A Compound Compression for Edge Deployment

Deploying large language models in edge environments requires running inference on low-power devices such as CPUs. Therefore, we follow the compound compression approach from [23] which bundles together structured, unstructured pruning, and quantization for efficient inference on CPUs. We start with ZipLM structurally pruned models, and apply on top the state-of-the-art oBERT unstructured pruning method [23] to 80% sparsity. After structured and unstructured pruning, we apply quantization-aware-training (QAT) [19] to quantize FP32 weights into INT8 representations. We benchmark these compound compressed models by running inference in the DeepSparse [36] engine, on a *single-core* of Intel Cascade Lake CPU. In this setting, we compare our results against the compound compression pipeline of [23] which applies layer dropping as a form of structured pruning. As can be seen from Figure 5, when we substitute layer dropping with a principled structured pruning via ZipLM, the resulting compound compressed models achieve very competitive latency-vs-accuracy performance in the edge-inference regime. At full accuracy recovery, ZipLM improves the speedup from 3x to 13x, while at the largest compression ratio ZipLM improves the speedup from 30x to 50x.

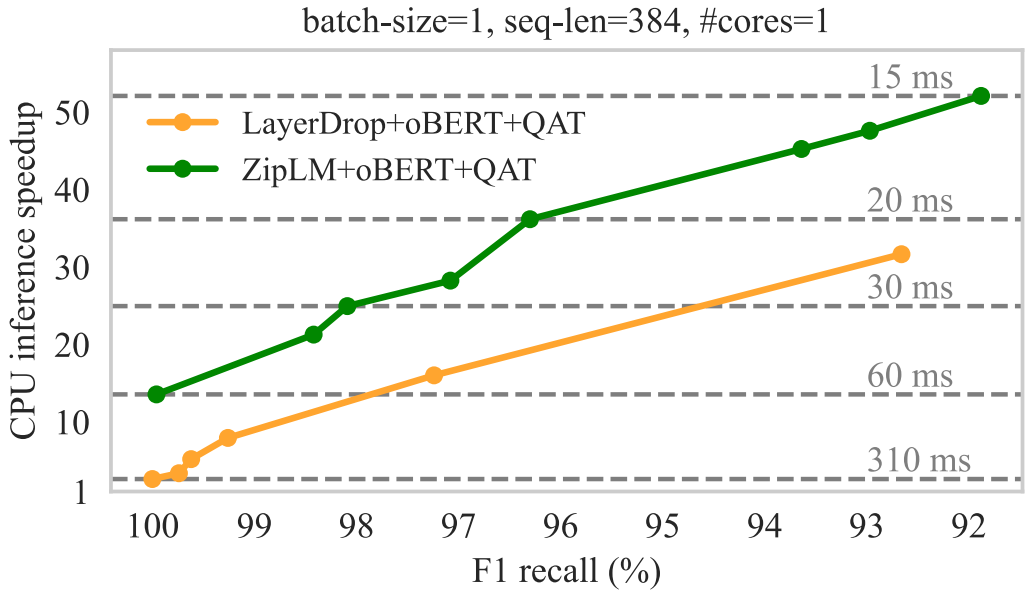

Figure 5: Improvements in CPU-inference speedups for compound compressed $\text{BERT}_{\text{base}}$ models on the SQuADv1.1 task when ZipLM is used for structured pruning. End-to-end latency indicated by the dashed line.

B Ablation Studies

In Table 3, we present ablation results for ZipLM and CoFi, with and without their respective layer-wise distillation techniques. ZipLM outperforms CoFi in all tasks when both methods use distillation, and in three out of four when distillation is not used. For example, ZipLM outperforms CoFi with a significant 3 point increase in F1 score on the SQuAD task in both setups. Furthermore, when comparing ZipLM results with and without layer-wise distillation, it can be observed that benefits are pronounced for low data tasks, where accuracy improvements reach up to 2 points.

C Additional Validation

Evaluating and comparing compressed models on the development set (dev-set) is standard practice, as it enables comparisons with off-the-shelf results from the literature. However, an implicit assumption behind such comparisons is that all methods tune their hyper-parameters only on a subset of the dev-set before evaluating and reporting results on all samples, which is not always the case.

Table 3: Comparison of ZipLM and CoFi dev-set results, with and without layer-wise distillation.

	SST-2 acc.	QNLI acc.	MNLI m-acc.	SQuAD F1
CoFi	90.4	86.1	80.6	82.6
ZipBERT$_{\text{base}}$	**91.7**	**88.6**	**81.7**	**85.7**
CoFi w/o $\mathcal{L}_{\text{layer}}$	**91.1**	85.1	79.7	82.5
ZipBERT$_{\text{base}}$ w/o $\mathcal{L}_{\text{token}}$	89.2	**86.5**	**81.2**	**85.7**

Table 4: Dev- and test-set comparison of ZipBERT$_{\text{base}}$ and CoFi models with comparable speedups.

	dev-set		test-set	
	CoFi	ZipBERT$_{\text{base}}$	CoFi	ZipBERT$_{\text{base}}$
QNLI, acc.	86.1	**88.6**	85.8	**88.4**
SST-2, acc.	90.4	**91.7**	88.2	**91.8**
MNLI, m-acc.	80.6	**81.7**	80.7	**81.9**
MNLI, mm-acc.	80.7	**82.0**	79.9	**80.6**
SQuAD, F1	82.6	**85.7**	N/A	N/A

Moreover, specifically-tuned hyper-parameters can lead to large performance differences, especially when compressing LLMs [22]. To ensure that there is no such "overfitting" on the dev-set, in Table 4 we compare ZipLM against the prior state-of-the-art CoFi approach on *unseen test-set*, obtained by submitting predictions to the official GLUE evaluation server. The results show consistent improvements over CoFi, on both dev- and test-sets.

D Speedup Evaluations

As shown in Figure 1, ZipLM is based on measuring runtimes of higher-level modules, such as attention heads and fully connected matrices, rather than low-level operators. This makes our approach independent of underlying optimizations in different inference engines and frameworks, which usually perform further optimizations such as operator-fusion. Our runtime lookup table contains information about the runtime of a Transformer layer with different numbers of attention heads, and various dimensions of the fully connected matrices. This implies that we measure runtimes of a layer with 12 heads, 11 heads, 10 heads, and so on, as well as the runtimes of fully connected matrices with hidden sizes ranging from 3072 to 0. We utilize this information to guide pruning decisions.

To fully validate the ability of ZipLM to compress the model while satisfying desired speedup constraints via the described approach, we provide the timing results in Table 5, comparing the desired (target) speedup and the achieved (measured) speedup for different models.

As can be seen from the Table 5, the deviation between the desired (target) and the achieved (measured) speedup is at most 5.28%. This confirms that our approach indeed provides reliable runtime information to guide the pruning decisions.

Table 5: Comparison of target (desired) inference speedups with achieved (on-device measured) speedups obtained with our ZipLM pruning approach.

BERT$_{\text{base}}$ on SQuADv1.1			BERT$_{\text{large}}$ on SQuADv1.1		
Target speedup	Achieved speedup	Deviation	Target speedup	Achieved speedup	Deviation
2	1.98	-1.00%	2	2.01	+0.50%
4	4.05	+1.25%	4	4.05	+1.25%
6	6.16	+2.67%	6	6.09	+1.50%
8	8.25	+3.12%	8	8.27	+3.37%
10	10.36	+3.60%	10	10.33	+3.30%
12	12.31	+2.58%	12	12.46	+3.83%
14	14.33	+2.35%	14	14.74	+5.28%

E Structure of Pruned Models

Through a comprehensive examination of ZipLM pruned BERT models across all datasets considered in Section 4, we aim to identify trends in the pruning of key components of the Transformer layer, namely attention heads and intermediate size, needed to achieve a specific speedup target. As illustrated in Figure 6, we observe that the intermediate size is pruned at a higher rate relative to attention heads, which aligns with the fact that the intermediate size dictates the dimensions of the two large linear layers in the feed-forward part of the Transformer block. For instance, to attain a 2x speedup, roughly 60% of the intermediate size and 40% of the attention heads need to be removed. Additionally, in Figure 7, we visualize the entire encoder size needed to reach a specific speedup target. Interestingly, we find that 15x faster models retain on average only 2% of intermediate size and 6% of attention heads which amounts to only 2.9M parameters overall, while at the same time recovering more than 95% of the uncompressed model's accuracy (see Figure 3).

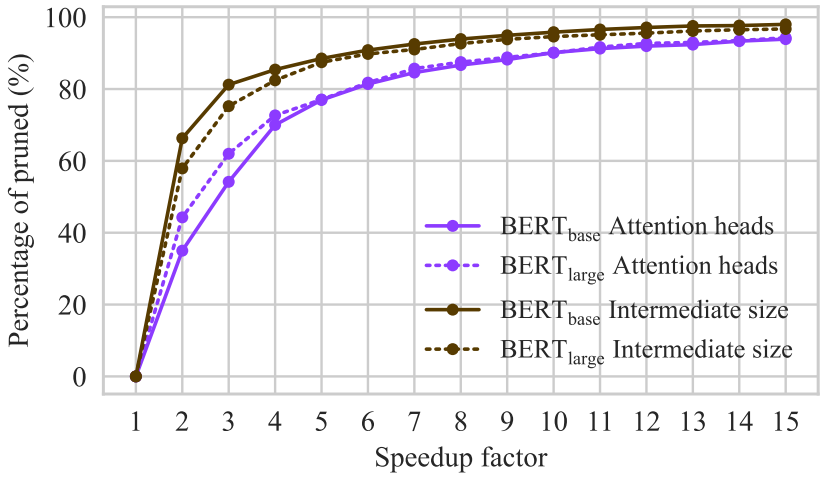

Figure 6: Percentage of pruned attention heads and intermediate size to reach a specific speedup target with ZipLM.

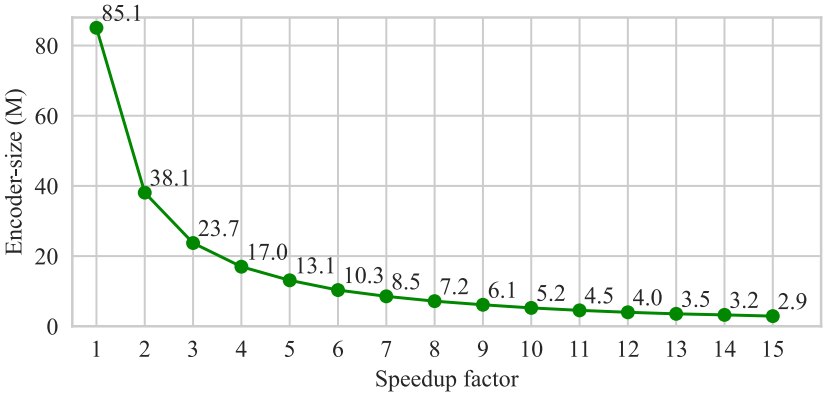

Figure 7: Encoder size vs. speedup factor of ZipLM pruned BERT$_{base}$ models, averaged over all considered datasets in Section 4.

Additionally, in Figures 8, 9, 10, 11 we visualize the number of remaining heads and intermediate size across all Transformer layers and various speedup targets on a subset of GLUE datasets.

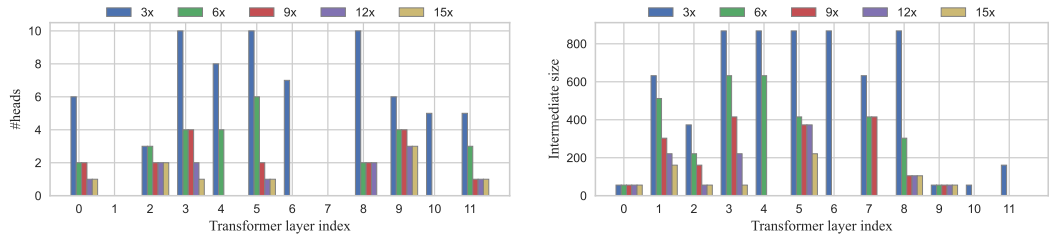

Figure 8: Remaining number of attention heads and intermediate size across all layers of the ZipLM compressed $\text{BERT}_{\text{base}}$ model at various speedups and MNLI dataset.

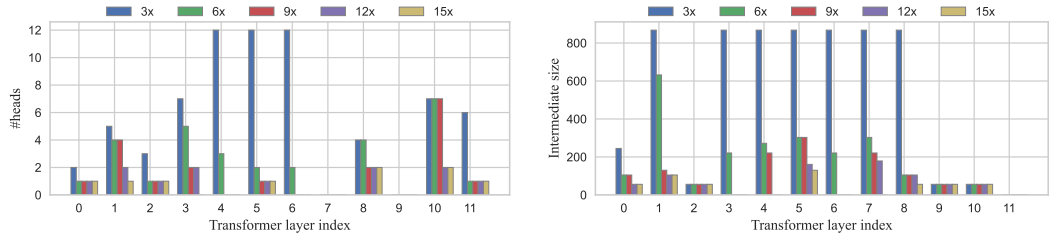

Figure 9: Remaining number of attention heads and intermediate size across all layers of the ZipLM compressed $\text{BERT}_{\text{base}}$ model at various speedups and QNLI dataset.

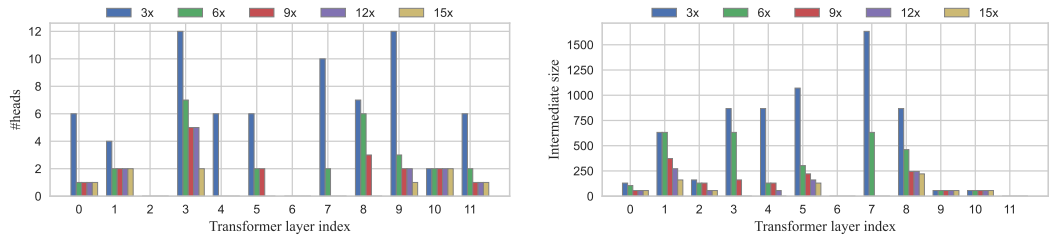

Figure 10: Remaining number of attention heads and intermediate size across all layers of the ZipLM compressed $\text{BERT}_{\text{base}}$ model at various speedups and QQP dataset.

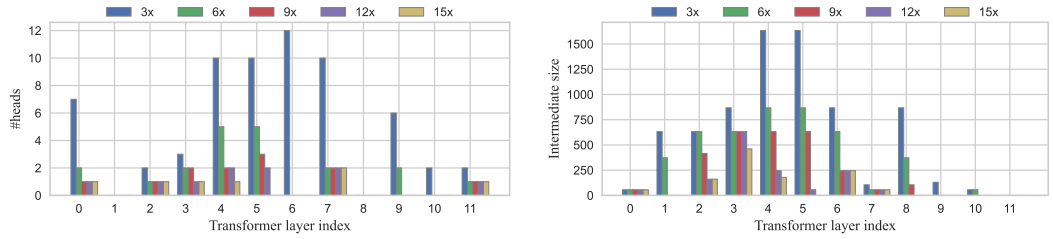

Figure 11: Remaining number of attention heads and intermediate size across all layers of the ZipLM compressed $\text{BERT}_{\text{base}}$ model at various speedups and SST-2 dataset.

F Experiments - Additional Results

In Table 6 we report accuracy and model size of ZipLM pruned models visualized in Section 4, in Figures 2 and 3.

G Hyper-parameters for Reproducibility

To facilitate reproducibility, we conduct experiments in the open-source Transformers library [54], and use publicly available datasets [27]. We plan to open-source our entire framework which supports

Table 6: Accuracy and model size for ZipLM pruned models in Section 4.

	$\text{BERT}_{\text{base}}$										$\text{BERT}_{\text{large}}$	
	QNLI		MNLI		SST2		QQP		SQuADv1		SQuADv1	
Speedup	Acc.	Encoder size (M)	Acc.	Encoder size (M)	Acc.	Encoder size (M)	Acc.	Encoder size (M)	F1	Encoder size (M)	F1	Encoder size (M)
2x	91.4	38.0	84.8	38.5	93.4	38.7	91.3	37.8	89.1	37.3	91.6	141.1
3x	91.1	23.8	84.8	23.5	93.4	24.1	91.3	23.8	88.6	23.4	91.4	88.3
4x	90.9	16.9	84.0	17.1	93.0	17.2	91.3	16.8	88.0	16.8	91.1	63.1
5x	90.8	12.5	84.0	13.5	93.0	13.5	91.1	13.0	87.5	13.0	90.8	48.5
6x	90.4	9.5	83.5	10.5	93.0	11.0	91.1	10.2	86.7	10.4	90.2	39.1
7x	89.8	8.0	83.2	8.8	93.0	9.0	90.9	8.1	86.1	8.7	89.9	32.7
8x	89.2	6.4	83.1	7.5	93.0	7.6	90.9	6.8	85.7	7.5	89.7	27.5
9x	89.1	5.7	82.8	6.3	93.0	6.7	90.8	5.8	85.3	6.2	89.3	23.8
10x	88.6	4.9	82.7	5.4	93.0	5.7	90.8	4.9	84.2	5.3	89.1	20.9
11x	88.6	4.0	82.5	4.7	92.7	4.9	90.7	4.3	83.8	4.7	88.8	18.4
12x	87.8	3.6	81.7	4.1	91.7	4.2	90.6	4.1	83.2	4.0	88.4	16.4
13x	87.6	3.2	81.3	3.5	91.7	3.8	90.6	3.7	82.5	3.4	87.9	14.9
14x	87.4	2.8	81.2	3.3	91.7	3.6	90.3	3.3	81.7	3.2	87.7	13.7
15x	87.2	2.6	80.8	2.9	90.7	3.2	90.3	2.9	81.4	2.9	87.6	12.5

Table 7: Hyper-parameters used for gradual ZipLM runs in Section 4.

	$\text{BERT}_{\text{base}}$	$\text{BERT}_{\text{large}}$	GPT2
batch-size	16 SQuADv1 32 GLUE		128
max-seq-length	384 SQuADv1 128 GLUE		1024
finetune before pruning	3 epochs		50k steps
finetune in-between pruning steps	8 epochs	10 epochs	2 epochs
LR schedule in-between pruning steps	linear decay		linear decay
initial LR	8e-5	5e-5	1e-3
#calibration samples	2048		512
speedup-targets	{2, 3, 4, 5, ..., 15}x		{1.5, 2, 2.5, 3}x
knowledge distillation λ_1	0		1.0
knowledge distillation λ_2	1.0 SQuADv1 0.5 GLUE		0
knowledge distillation λ_3	0.0 SQuADv1 0.5 GLUE		0
weight-decay	0.03	0.05	0

one-shot and gradual structured pruning via SparseML [24], making it very easy to experiment with other models and datasets. In addition to our code, we plan to open-source all of our compressed models via the popular HuggingFace Hub. In Table 7 we report hyper-parameters used to produce our ZipLM pruned models in Section 4. Because of the excessive memory overhead, we don't make use of any kind of knowledge distillation when pruning the GPT2 model. Following insights from DistilGTP2, we hypothesize that this can further improve our results. We follow [21] and disable dropout regularization while pre-training ZipGPT2 models at OpenWebTextCorpus dataset.

H Broader Impact and Limitations

Our results contribute to the line of work on efficient language models. Thus, it should help reduce the energy and monetary cost of inference over such models, and allow them to be used without

access to powerful hardware. While this is a mainly positive outcome, it also reduces the cost of employing these models for detrimental purposes, such as spam generation. Thus, this significant cost reduction for inference should also be seen as further motivation for methods to ensure safe usage of these models, such as watermarking or alignment.

As any academic study, our work is not without its limitations. All of our benchmarks are focused on English-language datasets and therefore our results do not provide insights into compression effects for low-data languages. Unfortunately, this limitation is inherent to all of the existing works on compression due to the lack of standardized benchmarks. Given that our structured pruning approach relies on a small sample of calibration data to perform pruning decisions, we hypothesize that our approach should be able to provide satisfying results in the low-data setup as well. At the moment we do not have data to support these claims, but we see it as an opportunity for future work.