GPT-Zip: Deep Compression of Finetuned Large Language Models

Berivan Isik [*1] **Hermann Kumbong** [*1] **Wanyi Ning** [*2] **Xiaozhe Yao** [*2] **Sanmi Koyejo** [1] **Ce Zhang** [2]

Abstract

Storage is increasingly a practical bottleneck to scaling large language model (LLM) systems with personalization, co-location, and other use cases that require storing the pretrained base model plus multiple finetuned models. To this end, we propose `GPT-Zip` for post-finetuning compression. `GPT-Zip` uses quantization and sparsification to efficiently compress finetuned models by exploiting their closeness to the pretrained base model. Specifically, we demonstrate that the *difference* between the finetuned models and the pretrained base model can efficiently be quantized into 2 bits and pruned with 95% sparsity together – providing up to 52 times overall size reduction. Thus, `GPT-Zip` avoids the linear growth in memory costs required for naive storage. We show that this compression can be achieved without performance degradation, as measured by evaluations on several tasks from the Natural Instructions dataset. Surprisingly, `GPT-Zip` sometimes improves accuracy over uncompressed models. We demonstrate the efficacy of `GPT-Zip` on four finetuned OPT-1.3B models and show that `GPT-Zip` reduces the storage cost by 16 times more than existing LLM compression techniques while attaining significantly better performance.

1. Introduction

In recent years, we have witnessed the remarkable success of Generative Pretrained Transformer models (known as GPT or OPT) (Brown et al., 2020; Radford et al., 2019; Vaswani et al., 2017; Zhang et al., 2022b) in various complex language applications. Since it is hard to access large task-specific data to train these models from scratch, it is common practice to first pretrain a base model on a large dataset for better initialization and then finetune the base model for task-specific and personalized experiences (Sanh et al., 2022). While finetuning yields state-of-the-art results, every time we finetune the base model for a new task, we end up with a new model (as large as the base model) with a new set of parameters for each task. This brings a massive obstacle in their storage and deployment since these models have billions of parameters.

One way of efficiently storing these finetuned models is to compress them individually by using the recent large language model (LLM) compression techniques such as `ZeroQuant` (Yao et al., 2022), `nuQmm` (Park et al., 2022), `LLM.int8()` (Dettmers et al., 2022), `GPT-Q` (Frantar et al., 2023), and `SparseGPT` (Frantar & Alistarh, 2023). However, these are typically efficient one-shot techniques for LLMs and cannot achieve ambitious compression gains without performance degradation. On the other hand, `AdaRound` (Nagel et al., 2020), `BitSplit` (Wang et al., 2020a), `AdaQuant` (Hubara et al., 2021), `BRECQ` (Li et al., 2021), `OBQ` (Frantar & Alistarh, 2022), and `OBS` (Hassibi et al., 1993b; Singh & Alistarh, 2020; Frantar et al., 2021) can compress the models more aggressively without a significant performance loss, but they are not scalable to LLMs since they require computationally expensive steps that scale with the number of parameters – such as computing the inverse Hessian or retraining.

In this work, we propose a compression framework called `GPT-Zip`[1], explicitly designed for compressing *finetuned* LLMs (i) in an efficient one-shot way without retraining, (ii) achieving ambitious compression gains such as 2-bit quantization and 95% sparsity simultaneously, (iii) while preserving the performance of the original finetuned model. Surprisingly, our experiments demonstrate that the finetuned models compressed with `GPT-Zip` sometimes outperform the original uncompressed finetuned models with respect to many metrics, including the accuracy on the finetuned task. More concretely, different from existing LLM compression techniques, `GPT-Zip` quantizes and prunes the difference $\mathbf{\Delta}$ between the finetuned model and the base model. Thus, whenever we finetune the base model for a new task, we only need to store the compressed $\mathbf{\Delta}$ instead of the full finetuned model. Our contributions can be summarized as:

(1) We propose `GPT-Zip` – the first compression technique

[*]Equal contribution [1]Stanford University [2]ETH Zurich. Correspondence to: Berivan Isik <berivan.isik@stanford.edu>.

ICML 2023 Workshop on Efficient Systems for Foundation Models. This workshop does not have official proceedings and this paper is non-archival.

[1]`GPT-Zip` is the name for the algorithm, not an acronym.

specifically designed for finetuned LLMs. We hope it will accelerate research on LLMs by allowing significantly more efficient storage of finetuned models (typically much more numerous and needed than the base models).

(2) We introduce two variants of GPT-Zip, one for quantization alone; and one for quantization and sparsification together.

(3) We show that having 2-bit quantization and 95% sparsity together yields up to 52 times compression of finetuned OPT-1.3B models without significant performance degradation with respect to accuracy on several tasks from the Natural Instructions dataset. This corresponds to a 16 times improvement over the prior LLM compression methods.

Related Work. We provide a summary of the literature on model compression, LLM compression, and efficient LLM finetuning in Appendix A.

2. Compressing Finetuned LLMs

We are interested in efficiently storing many finetuned models initialized from the same pretrained base model. We discover that compressing the difference $\mathbf{\Delta}$ between the finetuned and the base model provides significantly higher storage gains than compressing the finetuned model directly, even if we use state-of-the-art LLM compression techniques such as GPTQ (Frantar et al., 2023). To this end, we consider a base model with weights $\mathbf{W}_B$ pretrained to a local minimum in error and a target model with weights $\mathbf{W}_T$ finetuned from $\mathbf{W}_B$ for a downstream task. The finetuned model can be decomposed as $\mathbf{W}_T = \mathbf{W}_B + \mathbf{\Delta}$ with layer-wise summations. Our goal is to compress $\mathbf{\Delta}$ to $\hat{\mathbf{\Delta}}$ such that the new target model $\hat{\mathbf{W}}_T = \mathbf{W}_B + \hat{\mathbf{\Delta}}$ incurs a minimal increase in error. Following (Frantar & Alistarh, 2022; 2023; Frantar et al., 2023), we do layer-wise compression, i.e., we compress $\mathbf{\Delta}_l = \mathbf{W}_{T,l} - \mathbf{W}_{B,l}$ for each layer l separately and independently. In the rest of the paper, since we always work with layer l, we drop the subscript l for simplicity. Thus, we denote by $\mathbf{W}_B \in \mathbb{R}^{m\times n}$ the weights of the base model at layer l, $\mathbf{W}_T \in \mathbb{R}^{m\times n}$ the weights of the target finetuned model at layer l, $\mathbf{X} \in \mathbb{R}^{n\times k}$ the input to the layer l with k input samples, $\mathbf{\Delta} \in \mathbb{R}^{m\times n}$ the difference between the target finetuned model $\mathbf{W}_T$ and the base model $\mathbf{W}_B$, and $\hat{\mathbf{\Delta}} \in \mathbb{R}^{m\times n}$ its compressed version. Then, our goal is to solve the following optimization problem:

$$\begin{aligned}\min_{\hat{\mathbf{\Delta}}\in\mathbb{R}^{m\times n}} E(\hat{\mathbf{\Delta}}) &\equiv \min_{\hat{\mathbf{\Delta}}\in\mathbb{R}^{m\times n}} \|(\mathbf{W}_B + \hat{\mathbf{\Delta}})\mathbf{X} - (\mathbf{W}_B + \mathbf{\Delta})\mathbf{X}\|_2^2 \\ &\equiv \min_{\hat{\mathbf{\Delta}}\in\mathbb{R}^{m\times n}} \|\hat{\mathbf{\Delta}}\mathbf{X} - \mathbf{\Delta}\mathbf{X}\|_2^2. \end{aligned} \tag{1}$$

We first introduce GPT-Zip^Q in Section 2.1, which quantizes $\mathbf{\Delta}$ by minizing the error $E(\hat{\mathbf{\Delta}})$ in (1). Then in Section 2.2, we propose GPT-Zip^{Q,S} which combines GPT-Zip^Q with a simple sparsification step.

2.1. Quantization: GPT-Zip^Q

Similarly to (Frantar et al., 2023; Hassibi et al., 1993a), we take a greedy approach and quantize one parameter from $\mathbf{\Delta}$ at a time. To find which parameter to quantize next for the minimal increase in error and how to update the rest of the full-precision (not-yet-quantized) parameters to compensate for the error, we first simplify the objective in (1). We start by rewriting the ℓ_2 error $E(\hat{\mathbf{\Delta}})$ as a summation of the squared error of each row in $\mathbf{\Delta}$ as $E(\hat{\mathbf{\Delta}}) = \sum_{i=1}^m E_i(\hat{\mathbf{\Delta}}_{i,:})$, where $E_i(\hat{\mathbf{\Delta}}_{i,:}) = \|\hat{\mathbf{\Delta}}_{i,:}\mathbf{X} - \mathbf{\Delta}_{i,:}\mathbf{X}\|_2^2$ and $\mathbf{\Delta}_{i,:}$ is the i-th row of $\mathbf{\Delta}$. Then, the equivalent objective is

$$\min_{\hat{\mathbf{\Delta}}\in\mathbb{R}^{m\times n}} E(\hat{\mathbf{\Delta}}) \equiv \min_{\hat{\mathbf{\Delta}}\in\mathbb{R}^{m\times n}} \sum_{i=1}^m \|\hat{\mathbf{\Delta}}_{i,:}\mathbf{X} - \mathbf{\Delta}_{i,:}\mathbf{X}\|_2^2. \tag{2}$$

Now, we note three key observations which simplify our layer-wise $\mathbf{\Delta}$ compression problem in (2) to the problem setup in the state-of-the-art layer-wise LLM quantization works (Frantar & Alistarh, 2022; Frantar et al., 2023):

(1) We can assume that $\frac{\partial E(\hat{\mathbf{\Delta}})}{\partial \mathbf{\Delta}_{i,:}} \approx 0$ since both the base and the finetuned models are well-trained.

(2) Removing a parameter from one row does not affect the output of another row, i.e., there is no Hessian interaction between different rows. This allows us to work only with the smaller $n \times n$ Hessian corresponding to individual rows.

(3) The Hessian of the quadratic objective $E_i(\hat{\mathbf{\Delta}}_{i,:}) = \|\hat{\mathbf{\Delta}}_{i,:}\mathbf{X} - \mathbf{\Delta}_{i,:}\mathbf{X}\|_2^2$ is $\mathbf{H} = 2\mathbf{X}\mathbf{X}^T$ and same for all the rows since $\mathbf{\Delta}_{i,:}\mathbf{X}$ is fixed (not affected by compression).

Although their problem is different than ours since they care about quantizing the weights themselves (and not $\mathbf{\Delta}$), both OBQ (Frantar & Alistarh, 2022) and GPTQ (Frantar et al., 2023) propose greedy solutions to the same simplified layer-wise compression problem under the same observations above. The greedy approach in OBQ quantizes each row independently by quantizing one parameter at a time while updating the full-precision (not-yet-quantized) parameters to compensate for the error due to quantizing that last parameter. Denoting by F the set of remaining full-precision parameters in $\mathbf{\Delta}$ and by $\mathbf{H}_F$ the Hessian of the parameters from this set; OBQ suggests the following rules (by adapting it to our problem of compressing $\mathbf{\Delta}$) to determine which parameter δ_q from $\mathbf{\Delta}$ to quantize next and the corresponding optimal update $\mathbf{u}_F$ for all the parameters in F:

$$\delta_q = \arg\min_{\delta_q} \frac{(\text{quant}(\delta_q) - \delta_q)^2}{[\mathbf{H}_F^{-1}]_{qq}}, \quad \mathbf{u}_F = -\frac{\delta_q - \text{quant}(\delta_q)}{[\mathbf{H}_F^{-1}]_{qq}} \cdot (\mathbf{H}_F^{-1})_{:,q}, \tag{3}$$

where quant$(\cdot)$ is the rounding operation that rounds the input to the nearest value on the quantization grid, $[\mathbf{H}_F^{-1}]_{qq}$ denotes the q-th diagonal entry of the inverse Hessian, and $(\mathbf{H}_F^{-1})_{:,q}$ is its q-th column. OBQ algorithm with these two rules runs until all the parameters are quantized. However, the inverse Hessian must be recomputed every step after removing δ_q from F. This requires, for each row, inverting an $n \times n$ matrix at each of the $O(n)$ steps with a computational complexity $\Theta(n^3)$ – giving an overall runtime of $O(n^4)$ to finish quantizing the whole row. Unfortunately, this complexity is too high to scale this approach to even moderate-size layers with $\sim$ 10K parameters. As an alternative to this costly inverse operation, Frantar & Alistarh (2022) proposes using one-step Gaussian elimination by essentially removing the q-th row and column of the inverse Hessian directly, which has $\Theta(n^2)$ time complexity:

$$(\mathbf{H}_F^{-1})_{-q} = \left(\mathbf{H}_F^{-1} - \frac{1}{[\mathbf{H}_F^{-1}]_{qq}}(\mathbf{H}_F^{-1})_{:,q}(\mathbf{H}_F^{-1})_{:,q}\right)_{-q}, \tag{4}$$

where $(\mathbf{H}_F^{-1})_{-q}$ is the inverse of $\mathbf{H}_F$ with row and column q removed. This eliminates the full recomputation of the Hessian every step, speeds up OBQ significantly to the overall runtime of $O(n^3)$ for quantizing the whole row, and makes it applicable for neural networks as large as ResNet-50 (He et al., 2016) and BERT (Devlin et al., 2019). However, $O(m \cdot n^3)$ runtime per layer is still infeasibly slow for LLMs from the GPT family. Therefore, we follow a similar approach as GPTQ and quantize the parameters of $\mathbf{\Delta}$ in an arbitrary but fixed order instead of finding the optimal parameter to quantize at every step. In other words, for each row, instead of finding which δ_q to quantize next, we quantize the parameters in an arbitrary but same order for each row. This way, we can generate a list of $(\mathbf{H}_F^{-1})_{-q}$'s in (4) only once for one row and use the same inverse Hessian for all the other rows since $\mathbf{H} = 2\mathbf{X}\mathbf{X}^T$ is the same for all the rows in the same layer. Our experimental results verify the observation of Frantar et al. (2023) that fixing the order of quantizing parameters beforehand does not significantly increase error. This way, the total runtime for the whole layer reduces to $O(\max\{m \cdot n^2, n^3\})$ – providing enough speed up to run it on LLMs in a few GPU hours.

Similar to (Frantar et al., 2023), we apply GPT-Zip^Q to $B = 128$ columns at a time for a more memory-efficient computation by requiring the storage of fewer inverse Hessians. More specifically, we only need to recompute the new inverse Hessian after quantizing all the parameters in the set P of B columns as follows:

$$\mathbf{u}_F = -(\delta_P - \text{quant}(\delta_P))([\mathbf{H}_F^{-1}]_{PP})^{-1}(\mathbf{H}_F^{-1})_{:,P}, \tag{5}$$

$$(\mathbf{H}_F^{-1})_{-P} = \left(\mathbf{H}_F^{-1} - (\mathbf{H}_F^{-1})_{:,P}([\mathbf{H}_F^{-1}]_{PP})^{-1}(\mathbf{H}_F^{-1})_{P,:}\right)_{-P}. \tag{6}$$

With these modifications, we reach enough runtime and memory efficiency to run GPT-Zip^Q on OPT-1.3B in a few GPU hours without a memory problem.

2.2. Quantization Followed by Pruning: GPT-Zip^{Q,S}

We observe that the quantized $\hat{\mathbf{\Delta}}$ after running GPT-Zip^Q is still quite compressible. Even with hard thresholding based on the magnitude (which has been shown to work surprisingly well in smaller-scale models for various tasks (Han et al., 2016; 2015; Isik, 2021)) of the quantized parameters, we show that we can remove up to 95% of the parameters from quantized $\hat{\mathbf{\Delta}}$ and still maintain the same performance. We refer to this combination of GPT-Zip^Q and hard thresholding as GPT-Zip^{Q,S}.

3. Experiments

Overview: We demonstrate the efficacy of GPT-Zip on several tasks from the Nautural Instructions (NI) dataset (Mishra et al., 2022; Wang et al., 2022) and compare it against the state-of-the-art LLM compression framework, GPTQ (Frantar et al., 2023). More specifically, we pick four tasks from the NI dataset, namely *Answer Verification* (AV), *Irony Detection* (ID), *Toxic Language Detection* (TLD), and *Word Semantics* (WS); and finetune OPT-1.3B (Zhang et al., 2022a) on each of them separately. We provide the details of finetuning in Appendix B. Then, we compare the accuracy of the base OPT-1.3B model, uncompressed finetuned model, and compressed finetuned models on all four tasks. Here, our main goal is to preserve the accuracy of the original uncompressed finetuned model on the main task it was finetuned for, while not suffering from significant performance degradation on the other three tasks as well.

During the quantization step in GPT-Zip, to compute the Hessian matrix $\mathbf{H} = \mathbf{X}\mathbf{X}^T$, we need to use calibration data. For each finetuned model, we use 128 random 2048 token segments from the task it was finetuned for. For instance, to compress the OPT-1.3B model finetuned for the AV task, we use token segments from the AV task during calibration. This means that when evaluating a task other than AV, such as ID, TLD, or WS, GPT-Zip is actually not calibrated on the evaluation task – which provides us the zero-shot performance of GPT-Zip. Similar to (Dettmers et al., 2022; Frantar et al., 2023), we perform standard uniform per-row asymmetric quantization on the min-max grid.

Baselines: We compare GPT-Zip against the state-of-the-

art LLM compression framework, GPTQ (Frantar et al., 2023), which has the same runtime as GPT-Zip. Since we care about storing a pretrained base model and multiple finetuned models; we compare the required size for storing both the base and finetuned models. Our difference from GPTQ is in how we compress and store the finetuned models, whereas we can treat the base model in the same way, i.e., we can either compress it in the same way or leave it uncompressed. While GPTQ directly compresses the weights of the finetuned model $\mathbf{W}_T$, GPT-Zip compresses the difference $\mathbf{\Delta} = \mathbf{W}_T - \mathbf{W}_B$ between the target finetuned model $\mathbf{W}_T$ and the base model $\mathbf{W}_B$. The new finetuned model is then reconstructed as $\hat{\mathbf{W}}_T = \mathbf{W}_B + \hat{\mathbf{\Delta}}$. The reported size of the finetuned model for GPT-Zip is then the size of the compressed $\hat{\mathbf{\Delta}}$, instead of $\mathbf{W}_T$.

Results: We provide the results for OPT-1.3B finetuned for TLD in Table 1. Due to the page limit, we present the results for the other finetuned models tuned for AV, ID, and WS in Appendix C. We evaluate the finetuned models compressed with GPT-Zip (with different quantization and sparsity combinations) in terms of accuracy on all four tasks AV, ID, TLD, and WS and compare them with the finetuned models compressed with GPTQ, as well as the uncompressed finetuned model and the uncompressed base model. Additionally, we provide a comparison between GPT-Zip and GPTQ in Figure 1 on OPT-1.3B models finetuned for TLD and AV tasks. In the figure, we only plot the accuracy on the main task which is TLD for **(left)** and AV for **(right)** by choosing the best number of bits and sparsity combinations from Tables 1 and 2. As can be seen from the tables and the figure, GPTQ degrades the accuracy on all the tasks down to 0% for 2-bit quantization; whereas GPT-Zip is able to compress more than 5 times of this rate while still preserving the accuracy.

In our experiments, after compressing the finetuned model $\hat{\mathbf{W}}_\mathbf{T}$ with GPTQ or the difference $\hat{\mathbf{\Delta}}$ with GPT-Zip; we pack, zip, and store the compressed parameters following the implementation in https://github.com/PanQiWei/AutoGPTQ. At evaluation time, we read the finetuned model $\hat{\mathbf{W}}_\mathbf{T}$ from the disk and test it or read $\hat{\mathbf{\Delta}}$, reconstruct $\hat{\mathbf{W}}_\mathbf{T} = \mathbf{W_B} + \hat{\mathbf{\Delta}}$, and test it. Notice from Table 1 that even with the same quantization level, finetuned model $\hat{\mathbf{W}}_\mathbf{T}$ compressed with GPTQ takes significantly more space than $\hat{\mathbf{\Delta}}$ compressed with GPT-Zip – indicating that the entropy of the quantized $\hat{\mathbf{\Delta}}$ is much smaller than that of the quantized finetuned model $\hat{\mathbf{W}}_\mathbf{T}$. This supports our earlier claim that the difference $\mathbf{\Delta}$ is significantly more compressible than the finetuned model $\mathbf{W}_T$ itself.

4. Conclusion

In this work, we take the first step towards a paradigm shift for efficient compression of finetuned models by discovering that the difference between the models is much more compressible than the models themselves. We believe our work will motivate further research in exploiting this phenomenon as the interest and need for numerous finetuned models increase. Clearly, storing so many finetuned models will be a bottleneck in making them accessible. Our framework, GPT-Zip, closes this gap by compressing the difference between the finetuned models and the base model; and provides up to 16 times improvement in the compression rate over the state-of-the-art without performance degradation.

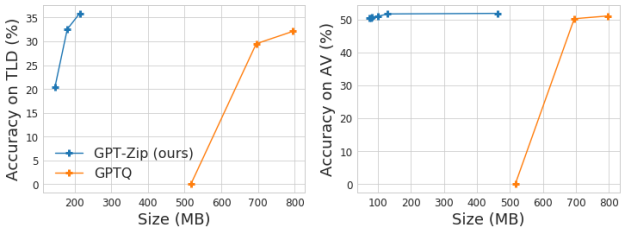

Figure 1: GPT-Zip vs GPTQ on compressing finetuned OPT-1.3B models tuned for **(left)** TLD and **(right)** AV tasks. For GPT-Zip, best quantization-sparsity combinations have been selected from Tables 1 and 2.

Table 1: Comparison of GPT-Zip with GPTQ (Frantar et al., 2023). Base model is OPT-1.3B. Finetuned model is OPT-1.3B tuned on TLD from NI dataset. Sparsity is the fraction of parameters that are set to zero. "# Bits" refers to the number of bits the parameters are quantized into. The size of the base model is 2.6 GB for all baselines. The main evaluation task is TLD since the model is finetuned for TLD.

Method	Final Size (Finetuned)	# Bits	Sparsity	TLD (main task)	AV	ID	WS
base model directly	-	-	-	14.88	50.41	58.14	19.23
uncomp. finetuned model	2.6 GB	16	0	29.73	49.26	50.61	14.00
GPTQ (Frantar et al., 2023)	795.63 MB	4	0	32.11	49.40	45.88	15.70
GPT-Zip^{Q} (ours)	522.07 MB	4	0	33.10	49.39	59.37	15.37
GPT-Zip^{Q,S} (ours)	**253.53 MB**	4	0.90	**33.61**	**49.19**	**59.72**	**17.93**
GPT-Zip^{Q,S} (ours)	**203.28 MB**	4	0.95	**31.65**	**50.54**	**59.02**	**18.06**
GPT-Zip^{Q,S} (ours)	**154.20 MB**	4	0.99	19.26	**50.44**	**59.19**	**18.79**
GPTQ (Frantar et al., 2023)	695.10 MB	3	0	29.49	47.88	51.66	8.27
GPT-Zip^{Q} (ours)	426.97 MB	3	0	33.07	49.42	59.72	15.52
GPT-Zip^{Q,S} (ours)	**241.25 MB**	3	0.90	**34.04**	**49.49**	**59.72**	**17.72**
GPT-Zip^{Q,S} (ours)	**196.63 MB**	3	0.95	**31.77**	**50.48**	**59.02**	**18.04**
GPT-Zip^{Q,S} (ours)	**151.83 MB**	3	0.99	18.84	**51.04**	**59.37**	**18.71**
GPTQ (Frantar et al., 2023)	517.04 MB	2	0	0	0	0	0
GPT-Zip^{Q} (ours)	280.84 MB	2	0	35.75	49.39	59.89	15.91
GPT-Zip^{Q,S} (ours)	**213.89 MB**	2	0.90	**35.87**	**49.16**	**60.25**	**17.44**
GPT-Zip^{Q,S} (ours)	**180.06 MB**	2	0.95	**32.55**	**49.88**	**59.02**	**18.29**
GPT-Zip^{Q,S} (ours)	**146.34 MB**	2	0.99	20.32	**50.54**	**59.54**	**18.74**

Limitations and Broader Impact. In the evaluation of GPT-Zip, similar to other LLM compression works, we only considered task accuracy and compression rate. However, compression might impact other properties of the finetuned model as well, such as fairness (Hooker et al., 2020), data leakage, and over-personalization. We believe these considerations deserve more attention from the community.

References

Aghajanyan, A., Gupta, S., and Zettlemoyer, L. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. 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)*, pp. 7319–7328, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.568. URL https://aclanthology.org/2021.acl-long.568.

Banner, R., Hubara, I., Hoffer, E., and Soudry, D. Scalable methods for 8-bit training of neural networks. In *Advances in neural information processing systems*, pp. 5145–5153, 2018.

Bapna, A. and Firat, O. Simple, scalable adaptation for neural machine translation. 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)*, pp. 1538–1548, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1165. URL https://aclanthology.org/D19-1165.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Chen, B., Dao, T., Winsor, E., Song, Z., Rudra, A., and Ré, C. Scatterbrain: Unifying sparse and low-rank attention. *Advances in Neural Information Processing Systems*, 34: 17413–17426, 2021a.

Chen, T., Cheng, Y., Gan, Z., Yuan, L., Zhang, L., and Wang, Z. Chasing sparsity in vision transformers: An end-to-end exploration. *Advances in Neural Information Processing Systems*, 34:19974–19988, 2021b.

Choi, Y., El-Khamy, M., and Lee, J. Universal deep neural network compression. *IEEE Journal of Selected Topics in Signal Processing*, 2020.

Dai, B., Zhu, C., Guo, B., and Wipf, D. Compressing neural networks using the variational information bottleneck. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1135–1144. PMLR, 10–15 Jul 2018. URL http://proceedings.mlr.press/v80/dai18d.html.

Dettmers, T., Lewis, M., Belkada, Y., and Zettlemoyer, L. Llm. int8 (): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*, 2022.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. 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)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

Ding, N., Qin, Y., Yang, G., Wei, F., Yang, Z., Su, Y., Hu, S., Chen, Y., Chan, C.-M., Chen, W., et al. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models. *arXiv preprint arXiv:2203.06904*, 2022.

Elsen, E., Dukhan, M., Gale, T., and Simonyan, K. Fast sparse convnets. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 14629–14638, 2020.

Fan, A., Grave, E., and Joulin, A. Reducing transformer depth on demand with structured dropout. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SylO2yStDr.

Federici, M., Ullrich, K., and Welling, M. Improved bayesian compression. *arXiv preprint arXiv:1711.06494*, 2017.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *International Conference on Learning Representations (ICLR)*, 2019.

Frantar, E. and Alistarh, D. Optimal brain compression: A framework for accurate post-training quantization and pruning. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=ksVGCOlOEba.

Frantar, E. and Alistarh, D. SparseGPT: Massive language models can be accurately pruned in one-shot. *arXiv preprint arXiv:2301.00774*, 2023.

Frantar, E., Kurtic, E., and Alistarh, D. M-fac: Efficient matrix-free approximations of second-order information. *Advances in Neural Information Processing Systems*, 34: 14873–14886, 2021.

Frantar, E., Ashkboos, S., Hoefler, T., and Alistarh, D. OPTQ: Accurate quantization for generative pre-trained

transformers. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=tcbBPnfwxS.

Guo, D., Rush, A., and Kim, Y. 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)*, pp. 4884–4896, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.378. URL https://aclanthology.org/2021.acl-long.378.

Han, S., Pool, J., Tran, J., and Dally, W. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pp. 1135–1143, 2015.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *International Conference on Learning Representations (ICLR)*, 2016.

Hassibi, B., Stork, D. G., Wolff, G., and Watanabe, T. Optimal brain surgeon: Extensions and performance comparisons. In *Proceedings of the 6th International Conference on Neural Information Processing Systems*, NIPS'93, pp. 263–270, San Francisco, CA, USA, 1993a.

Hassibi, B., Stork, D. G., and Wolff, G. J. Optimal brain surgeon and general network pruning. In *IEEE international conference on neural networks*, pp. 293–299. IEEE, 1993b.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015. URL http://arxiv.org/abs/1503.02531.

Hooker, S., Moorosi, N., Clark, G., Bengio, S., and Denton, E. Characterising bias in compressed models. *arXiv preprint arXiv:2010.03058*, 2020.

Hou, L., Huang, Z., Shang, L., Jiang, X., Chen, X., and Liu, Q. Dynabert: Dynamic bert with adaptive width and depth. *Advances in Neural Information Processing Systems*, 33:9782–9793, 2020.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pp. 2790–2799. PMLR, 2019.

Hu, E. J., yelong shen, Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Hubara, I., Nahshan, Y., Hanani, Y., Banner, R., and Soudry, D. Accurate post training quantization with small calibration sets. In *International Conference on Machine Learning*, pp. 4466–4475. PMLR, 2021.

Idelbayev, Y. and Carreira-Perpinan, M. A. Low-rank compression of neural nets: Learning the rank of each layer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Idelbayev, Y., Molchanov, P., Shen, M., Yin, H., Carreira-Perpinan, M. A., and Alvarez, J. M. Optimal quantization using scaled codebook. In *Proc. of the 2021 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'21), Virtual*, 2021.

Ioannou, Y., Robertson, D., Shotton, J., Cipolla, R., and Criminisi, A. Training cnns with low-rank filters for efficient image classification. *arXiv preprint arXiv:1511.06744*, 2015.

Isik, B. Neural 3d scene compression via model compression. *arXiv preprint arXiv:2105.03120*, 2021.

Isik, B., Weissman, T., and No, A. An information-theoretic justification for model pruning. In *International Conference on Artificial Intelligence and Statistics*, pp. 3821–3846. PMLR, 2022.

Isik, B., Choi, K., Zheng, X., Weissman, T., Ermon, S., Wong, H.-S. P., and Alaghi, A. Neural network compression for noisy storage devices. *ACM Trans. Embed. Comput. Syst.*, mar 2023. ISSN 1539-9087. doi: 10.1145/3588436. URL https://doi.org/10.1145/3588436.

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., and Kalenichenko, D. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2704–2713, 2018.

Jiao, X., Yin, Y., Shang, L., Jiang, X., Chen, X., Li, L., Wang, F., and Liu, Q. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 4163–4174, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.372. URL https://aclanthology.org/2020.findings-emnlp.372.

Jung, S., Son, C., Lee, S., Son, J., Han, J.-J., Kwak, Y., Hwang, S. J., and Choi, C. Learning to quantize deep networks by optimizing quantization intervals with task loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4350–4359, 2019.

Khetan, A. and Karnin, Z. schuBERT: Optimizing elements of BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 2807–2818, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.250. URL https://aclanthology.org/2020.acl-main.250.

Kwon, W., Kim, S., Mahoney, M. W., Hassoun, J., Keutzer, K., and Gholami, A. A fast post-training pruning framework for transformers. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=0GRBKLBjJE.

Lagunas, F., Charlaix, E., Sanh, V., and Rush, A. Block pruning for faster transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 10619–10629, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.829. URL https://aclanthology.org/2021.emnlp-main.829.

LeCun, Y., Denker, J., and Solla, S. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

Li, B., Kong, Z., Zhang, T., Li, J., Li, Z., Liu, H., and Ding, C. Efficient transformer-based large scale language representations using hardware-friendly block structured pruning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 3187–3199, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.286. URL https://aclanthology.org/2020.findings-emnlp.286.

Li, F., Zhang, B., and Liu, B. Ternary weight networks. *arXiv preprint arXiv:1605.04711*, 2016.

Li, Y., Gong, R., Tan, X., Yang, Y., Hu, P., Zhang, Q., Yu, F., Wang, W., and Gu, S. {BRECQ}: Pushing the limit of post-training quantization by block reconstruction. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=POWv6hDd9XH.

Lin, Z., Liu, J., Yang, Z., Hua, N., and Roth, D. Pruning redundant mappings in transformer models via spectral-normalized identity prior. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 719–730, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.64. URL https://aclanthology.org/2020.findings-emnlp.64.

Liu, H., Tam, D., Mohammed, M., Mohta, J., Huang, T., Bansal, M., and Raffel, C. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=rBCvMG-JsPd.

Liu, Y., Lin, Z., and Yuan, F. Rosita: Refined bert compression with integrated techniques. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 8715–8722, 2021a.

Liu, Z., Li, F., Li, G., and Cheng, J. Ebert: Efficient bert inference with dynamic structured pruning. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 4814–4823, 2021b.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Louizos, C., Ullrich, K., and Welling, M. Bayesian compression for deep learning. *arXiv preprint arXiv:1705.08665*, 2017a.

Louizos, C., Welling, M., and Kingma, D. P. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*, 2017b.

Mishra, S., Khashabi, D., Baral, C., and Hajishirzi, H. Cross-task generalization via natural language crowdsourcing instructions. In *ACL*, 2022.

Molchanov, D., Ashukha, A., and Vetrov, D. Variational dropout sparsifies deep neural networks. In *International Conference on Machine Learning*, pp. 2498–2507. PMLR, 2017.

Nagel, M., Amjad, R. A., Van Baalen, M., Louizos, C., and Blankevoort, T. Up or down? adaptive rounding for post-training quantization. In *International Conference on Machine Learning*, pp. 7197–7206. PMLR, 2020.

Park, G., Park, B., Kwon, S. J., Kim, B., Lee, Y., and Lee, D. nuqmm: Quantized matmul for efficient inference of large-scale generative language models. *arXiv preprint arXiv:2206.09557*, 2022.

Park, S., Lee, J., Mo, S., and Shin, J. Lookahead: A far-sighted alternative of magnitude-based pruning. *International Conference on Learning Representations (ICLR)*, 2020.

Polino, A., Pascanu, R., and Alistarh, D. Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668*, 2018.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Rebuffi, S.-A., Bilen, H., and Vedaldi, A. Learning multiple visual domains with residual adapters. *Advances in neural information processing systems*, 30, 2017.

Renda, A., Frankle, J., and Carbin, M. Comparing fine-tuning and rewinding in neural network pruning. In *International Conference on Learning Representations*, 2020.

Sainath, T. N., Kingsbury, B., Sindhwani, V., Arisoy, E., and Ramabhadran, B. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 6655–6659. IEEE, 2013.

Sajjad, H., Dalvi, F., Durrani, N., and Nakov, P. On the effect of dropping layers of pre-trained transformer models. *Computer Speech & Language*, 77:101429, 2023.

Sanh, V., Webson, A., Raffel, C., Bach, S., Sutawika, L., Alyafeai, Z., Chaffin, A., Stiegler, A., Raja, A., Dey, M., Bari, M. S., Xu, C., Thakker, U., Sharma, S. S., Szczechla, E., Kim, T., Chhablani, G., Nayak, N., Datta, D., Chang, J., Jiang, M. T.-J., Wang, H., Manica, M., Shen, S., Yong, Z. X., Pandey, H., Bawden, R., Wang, T., Neeraj, T., Rozen, J., Sharma, A., Santilli, A., Fevry, T., Fries, J. A., Teehan, R., Scao, T. L., Biderman, S., Gao, L., Wolf, T., and Rush, A. M. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=9Vrb9D0WI4.

Singh, S. P. and Alistarh, D. Woodfisher: Efficient second-order approximation for neural network compression. *Advances in Neural Information Processing Systems*, 33: 18098–18109, 2020.

Sun, S., Cheng, Y., Gan, Z., and Liu, J. 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)*, pp. 4323–4332, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1441. URL https://aclanthology.org/D19-1441.

Sung, Y.-L., Nair, V., and Raffel, C. A. Training neural networks with fixed sparse masks. *Advances in Neural Information Processing Systems*, 34:24193–24205, 2021.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Voita, E., Talbot, D., Moiseev, F., Sennrich, R., and Titov, I. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5797–5808, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1580. URL https://aclanthology.org/P19-1580.

Wang, J., Bao, W., Sun, L., Zhu, X., Cao, B., and Philip, S. Y. Private model compression via knowledge distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 1190–1197, 2019a.

Wang, K., Liu, Z., Lin, Y., Lin, J., and Han, S. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8612–8620, 2019b.

Wang, P., Chen, Q., He, X., and Cheng, J. Towards accurate post-training network quantization via bit-split and stitching. In *International Conference on Machine Learning*, pp. 9847–9856. PMLR, 2020a.

Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., and Zhou, M. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in Neural Information Processing Systems*, 33: 5776–5788, 2020b.

Wang, Y., Mishra, S., Alipoormolabashi, P., Kordi, Y., Mirzaei, A., Arunkumar, A., Ashok, A., Dhanasekaran, A. S., Naik, A., Stap, D., et al. Super-naturalinstructions:generalization via declarative instructions on 1600+ tasks. In *EMNLP*, 2022.

Xin, J., Tang, R., Lee, J., Yu, Y., and Lin, J. DeeBERT: Dynamic early exiting for accelerating BERT inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 2246–2251, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.204. URL https://aclanthology.org/2020.acl-main.204.

Yao, Z., Ma, L., Shen, S., Keutzer, K., and Mahoney, M. W. Mlpruning: A multilevel structured pruning framework for transformer-based models. *arXiv preprint arXiv:2105.14636*, 2021.

Yao, Z., Yazdani Aminabadi, R., Zhang, M., Wu, X., Li, C., and He, Y. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in Neural Information Processing Systems*, 35: 27168–27183, 2022.

Young, S. I., Zhe, W., Taubman, D., and Girod, B. Transform quantization for cnn compression. *arXiv preprint arXiv:2009.01174*, 2020.

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V., et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022a.

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V., et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022b.

Zhou, W., Xu, C., Ge, T., McAuley, J., Xu, K., and Wei, F. Bert loses patience: Fast and robust inference with early exit. *Advances in Neural Information Processing Systems*, 33:18330–18341, 2020.

A. Related Work

In this section, we first briefly summarize the model compression literature by focusing on the prior work that is most relevant to us, then go over the recent LLM compression techniques, and finally discuss prior work on efficient finetuning of LLMs.

Model Compression. The recent progress in machine learning applications has come with the challenge of storing and deploying overparameterized models on resource-constrained devices. This has motivated significant efforts in various model compression approaches such as pruning (Elsen et al., 2020; Frankle & Carbin, 2019; Han et al., 2016; Hassibi et al., 1993b; Isik, 2021; Isik et al., 2023; 2022; LeCun et al., 1989; Park et al., 2020; Renda et al., 2020), quantization (Banner et al., 2018; Choi et al., 2020; Idelbayev et al., 2021; Jacob et al., 2018; Jung et al., 2019; Li et al., 2016; Wang et al., 2019b; Young et al., 2020), low-rank factorization (Idelbayev & Carreira-Perpinan, 2020; Ioannou et al., 2015; Sainath et al., 2013), Bayesian compression (Dai et al., 2018; Federici et al., 2017; Louizos et al., 2017a;b; Molchanov et al., 2017), and knowledge distillation (Hinton et al., 2015; Polino et al., 2018; Wang et al., 2019a). Among them, some specifically have focused on transformer models and proposed quantization (Frantar & Alistarh, 2022; Hubara et al., 2021; Li et al., 2021; Nagel et al., 2020; Wang et al., 2020a), pruning (Chen et al., 2021b; Fan et al., 2020; Hou et al., 2020; Khetan & Karnin, 2020; Kwon et al., 2022; Lagunas et al., 2021; Li et al., 2020; Lin et al., 2020; Liu et al., 2021b; Sajjad et al., 2023; Voita et al., 2019; Xin et al., 2020; Yao et al., 2021; Zhou et al., 2020), low-rank factorization (Chen et al., 2021a; Liu et al., 2021a), and knowledge distillation (Jiao et al., 2020; Liu et al., 2021a; Sun et al., 2019; Wang et al., 2020b) methods for transformers. While these methods achieve a good tradeoff between the compression ratio and final performance, they require computationally expensive operations such as retraining and inverting large matrices (e.g., Hessian). As a result, even for models up to ≈ 100 million parameters, compression takes a few GPU hours. This makes it infeasible to scale them to orders of magnitude larger models, such as LLMs from the GPT family. In this work, we aim to achieve a significantly better compression-performance tradeoff without suffering from high computational costs and compress finetuned LLMs in less than a few GPU hours.

Large Language Model Compression. The transformer compression methods mentioned in the previous paragraph, although achieving appealing compression-performance tradeoffs, are not scalable to LLMs. Therefore, there has been a recent interest in developing alternative techniques specifically for larger models at the cost of some performance drop. For instance, `SparseGPT` (Frantar & Alistarh, 2023) is significantly faster than the prior transformer pruning methods (Frantar & Alistarh, 2022; Hubara et al., 2021) – making it applicable to LLMs. Similarly, `GPTQ` (Frantar et al., 2023), `ZeroQuant` (Yao et al., 2022), `nuQmm` (Park et al., 2022), `LLM.int8()` (Dettmers et al., 2022) avoid costly operations present in (Frantar & Alistarh, 2022; Li et al., 2021; Nagel et al., 2020) to scale up transformer quantization methods to LLMs. However, eliminating these costly operations comes with a price. The existing LLM compression techniques do not enjoy the same compression-performance gains as the smaller-scale model compression methods since they compromise optimality for computational efficiency. In this work, while compressing the finetuned models, we aim to enjoy both computational efficiency and a much better compression-performance tradeoff by exploiting the closeness between the base and finetuned models.

Efficient Finetuning of Large Language Models. Another line of work that is relevant to us are the recent efforts in the efficient finetuning of transformer models. Due to the high computational cost of finetuning the whole base model, more efficient ways for finetuning have attracted attention from the community (Liu et al., 2022). Some earlier efforts include training only adapters (Bapna & Firat, 2019; Houlsby et al., 2019; Rebuffi et al., 2017) (trainable layers inserted between the frozen layers of the base model), finetuning only a sparse subnet of the base model while keeping the rest frozen (Guo et al., 2021; Sung et al., 2021), and finetuning in a lower-dimensional subspace (Aghajanyan et al., 2021). More recently, low-rank adapters for LLMs called `LoRA`, have been introduced to freeze some layers in the base model and train low-rank adapters on top of them (Hu et al., 2022). `LoRA` has successfully finetuned LLMs with billions of parameters in a computationally efficient way while achieving reasonable performance. However, we note that the low-rank adapters in `LoRA` are only added to the self-attention layers to avoid drastic performance drops. Hence the other layers are still being fully finetuned as usual, which limits the potential storage efficiency `LoRA` can provide. Our work is tangential to efficient finetuning efforts since we are interested in compressing fully finetuned models for two reasons: (1) we need immediate solutions to the high storage cost of increasingly many fully finetuned models to accelerate research, and (2) efficiently finetuned models do not reach the same performance as the fully finetuned models which are trained without computational concerns (Ding et al., 2022).

B. Additional Experimental Details

We finetune the OPT-1.3B model on four different classification tasks from the natural instructions dataset, namely, *Answer Verification* (AV), *Irony Detection* (ID), *Toxic Language Detection* (TLD), and *Word Semantics* (WS). For each task, we use 80% of the data for finetuning and the remaining 20% for evaluation. We apply AdamW (Loshchilov & Hutter, 2017) with a learning rate $1e-5$, $\beta = (0.9, 0.999)$, and weight decay 0.01. We finetune for 1 epochs with a batch size of 16 and a maximum sequence length of 2048. We use the same hyperparameters for all tasks. We finetune all models on 8 NVIDIA RTX A6000 with a data parallel degree of 4 and a pipeline parallel degree of 2.

After finetuning, we compress the finetuned models using `GPT-Zip` and `GPT-Q`. For all methods, our calibration data consists of 128 random sequences from the training data.

C. Additional Experimental Results

We now provide additional results we had to skip in the main body due to the page limit. Figure 2 shows the comparison between `GPT-Zip` and `GPTQ` on the OPT-1.3B models finetuned for ID and WS tasks from the NI dataset. For the `GPT-Zip` curve, the best combinations of the number of quantization and sparsity are selected. More detailed results on all four tasks are provided in Tables 2, 3, and 4.

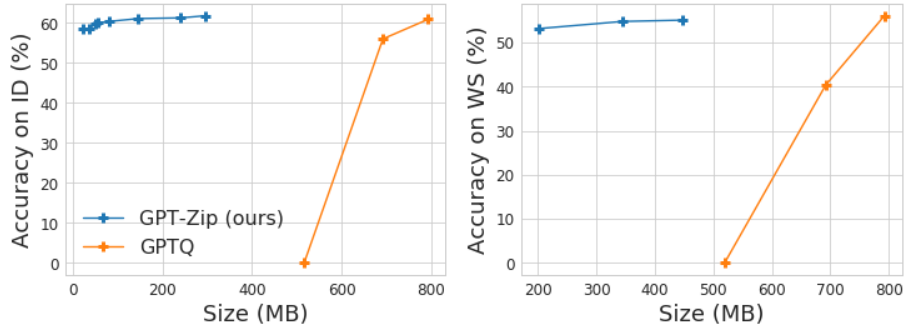

Figure 2: `GPT-Zip` vs `GPTQ` on compressing finetuned OPT-1.3B models tuned for **(left)** ID and **(right)** WS tasks. For `GPT-Zip`, best quantization-sparsity combinations have been selected from Tables 3 and 4.

Table 2: Comparison of GPT-Zip with GPTQ (Frantar et al., 2023). Base model is OPT-1.3B. Finetuned model is OPT-1.3B tuned on ID from NI dataset. Sparsity is the fraction of parameters that are set to zero. "# Bits" refers to the number of bits the parameters are quantized into. The size of the base model is 2.6 GB for all baselines. The main evaluation task is AV since the model is finetuned for AV.

Method	Final Size (Finetuned)	# Bits	Sparsity	AV (main task)	ID	TLD	WS
base model directly	-	-	-	50.41	58.14	14.88	19.23
uncomp. finetuned model	2.6 GB	16	0	51.46	41.68	25.96	13.94
GPTQ (Frantar et al., 2023)	795.31 MB	4	0	51.04	38.53	30.38	13.35
GPT-Zip^{Q} (ours)	**461.47 MB**	4	0	**51.83**	**57.39**	**27.47**	15.47
GPT-Zip^{Q,S} (ours)	**172.60 MB**	4	0.90	**51.17**	**59.37**	**23.49**	**17.72**
GPT-Zip^{Q,S} (ours)	**125.55 MB**	4	0.95	**50.90**	**59.37**	20.21	**18.48**
GPT-Zip^{Q,S} (ours)	**80.94 MB**	4	0.99	50.67	**58.84**	16.53	**18.99**
GPTQ (Frantar et al., 2023)	694.53 MB	3	0	50.21	27.85	36.78	13.32
GPT-Zip^{Q} (ours)	362.35 MB	3	0	50.84	56.39	**27.44**	15.57
GPT-Zip^{Q,S} (ours)	**162.42 MB**	3	0.90	**51.30**	**59.37**	**23.54**	**17.83**
GPT-Zip^{Q,S} (ours)	**119.24 MB**	3	0.95	**50.84**	**59.54**	20.11	**18.40**
GPT-Zip^{Q,S} (ours)	**79.01 MB**	3	0.99	**50.51**	**58.67**	16.55	**19.12**
GPTQ (Frantar et al., 2023)	515.87 MB	2	0	0	0	0	0
GPT-Zip^{Q} (ours)	224.57 MB	2	0	50.58	55.69	26.19	15.57
GPT-Zip^{Q,S} (ours)	**128.70 MB**	2	0.90	**51.66**	**59.37**	**20.97**	**17.96**
GPT-Zip^{Q,S} (ours)	**100.09 MB**	2	0.95	**50.94**	**59.54**	**18.15**	**18.48**
GPT-Zip^{Q,S} (ours)	**73.93 MB**	2	0.99	**50.44**	**58.67**	**15.83**	**18.87**

Table 3: Comparison of GPT-Zip with GPTQ (Frantar et al., 2023). Base model is OPT-1.3B. Finetuned model is OPT-1.3B tuned on ID from NI dataset. Sparsity is the fraction of parameters that are set to zero. "# Bits" refers to the number of bits the parameters are quantized into. The size of the base model is 2.6 GB for all baselines. The main evaluation task is ID since the model is finetuned for ID.

Method	Final Size (Finetuned)	# Bits	Sparsity	ID (main task)	AV	TLD	WS
base model directly	-	-	-	58.14	50.41	14.88	19.23
uncomp. finetuned model	2.6 GB	16	0	62.35	51.00	33.31	17.03
GPTQ (Frantar et al., 2023)	793.81 MB	4	0	60.95	50.67	28.63	18.11
GPT-Zip^{Q} (ours)	**294.57 MB**	4	0	**61.82**	**51.43**	**22.49**	**18.29**
GPT-Zip^{Q,S} (ours)	**81.44 MB**	4	0.90	**60.42**	50.18	16.27	**19.07**
GPT-Zip^{Q,S} (ours)	**51.18 MB**	4	0.95	**59.72**	**50.54**	**15.56**	**19.15**
GPT-Zip^{Q,S} (ours)	**24.60 MB**	4	0.99	**58.14**	**50.54**	**15.03**	**19.23**
GPTQ (Frantar et al., 2023)	692.66 MB	3	0	56.04	21.55	18.82	13.24
GPT-Zip^{Q} (ours)	**240.76 MB**	3	0	**61.30**	**50.67**	**22.31**	**18.48**
GPT-Zip^{Q,S} (ours)	**55.32 MB**	3	0.90	**60.07**	**50.44**	15.80	**18.92**
GPT-Zip^{Q,S} (ours)	**35.84 MB**	3	0.95	**58.67**	**50.58**	15.31	**19.25**
GPT-Zip^{Q,S} (ours)	**22.31 MB**	3	0.99	**58.32**	**50.61**	14.99	**19.25**
GPTQ (Frantar et al., 2023)	516.67 MB	2	0	0	0	0	0
GPT-Zip^{Q} (ours)	**146.23 MB**	2	0	**61.12**	**50.48**	**21.50**	**18.71**
GPT-Zip^{Q,S} (ours)	**41.68 MB**	2	0.90	**58.67**	**50.54**	15.39	**19.25**
GPT-Zip^{Q,S} (ours)	**32.99 MB**	2	0.95	**58.49**	**50.44**	15.29	**19.23**
GPT-Zip^{Q,S} (ours)	**21.81 MB**	2	0.99	**58.49**	**50.47**	15.09	**19.36**

Table 4: Comparison of GPT-Zip with GPTQ (Frantar et al., 2023). Base model is OPT-1.3B. Finetuned model is OPT-1.3B tuned on ID from NI dataset. Sparsity is the fraction of parameters that are set to zero. "# Bits" refers to the number of bits the parameters are quantized into. The size of the base model is 2.6 GB for all baselines. The main evaluation task is WS since the model is finetuned for WS.

Method	Final Size (Finetuned)	# Bits	Sparsity	WS (main task)	AV	ID	TLD
base model directly	-	-	-	19.23	50.41	58.14	14.88
uncomp. finetuned model	2.6 GB	16	0	61.80	50.71	46.94	30.28
GPTQ (Frantar et al., 2023)	792.47 MB	4	0	56.05	50.71	40.81	22.20
GPT-Zip^{Q} (ours)	**447.49 MB**	4	0	**55.07**	**50.71**	**56.04**	**32.27**
GPT-Zip^{Q,S} (ours)	**132.61 MB**	4	0.90	**30.60**	50.15	**58.84**	**23.05**
GPT-Zip^{Q,S} (ours)	**88.24 MB**	4	0.95	25.42	50.64	59.02	**18.69**
GPT-Zip^{Q,S} (ours)	**47.10 MB**	4	0.99	21.17	**50.90**	58.67	15.80
GPTQ (Frantar et al., 2023)	692.14 MB	3	0	40.32	47.35	1.40	18.73
GPT-Zip^{Q} (ours)	**344.15 MB**	3	0	**54.76**	**50.71**	**56.39**	**32.22**
GPT-Zip^{Q,S} (ours)	**121.86 MB**	3	0.90	**30.66**	**50.18**	**59.02**	**23.22**
GPT-Zip^{Q,S} (ours)	**83.01 MB**	3	0.95	**25.84**	**50.77**	**59.02**	**18.66**
GPT-Zip^{Q,S} (ours)	**44.70 MB**	3	0.99	21.09	**50.67**	**59.02**	15.55
GPTQ (Frantar et al., 2023)	519.40 MB	2	0	0	0	0	0
GPT-Zip^{Q} (ours)	**201.86 MB**	2	0	**53.15**	**50.71**	**56.04**	**32.13**
GPT-Zip^{Q,S} (ours)	**91.86 MB**	2	0.90	**27.73**	**50.28**	**59.19**	**18.62**
GPT-Zip^{Q,S} (ours)	**65.93 MB**	2	0.95	**23.94**	**50.41**	**59.54**	**16.79**
GPT-Zip^{Q,S} (ours)	**40.38 MB**	2	0.99	20.68	**50.48**	**58.84**	15.39