Quadapter: Adapter for GPT-2 Quantization

Minseop Park, Jaesong You, Markus Nagel, Simyung Chang
Qualcomm AI Research*
{minspark, jaseong, markusn, simychan}@qti.qualcomm.com

Abstract

Transformer language models such as GPT-2 are difficult to quantize because of outliers in activations leading to a large quantization error. To adapt to the error, one must use quantization-aware training, which entails a fine-tuning process based on the dataset and the training pipeline identical to those for the original model. Pretrained language models, however, often do not grant access to their datasets and training pipelines, forcing us to rely on arbitrary ones for fine-tuning. In that case, it is observed that quantization-aware training overfits the model to the fine-tuning data. For quantization without overfitting, we introduce a quantization adapter (Quadapter), a small set of parameters that are learned to make activations quantization-friendly by scaling them channel-wise. It keeps the model parameters unchanged. By applying our method to the challenging task of quantizing GPT-2, we demonstrate that it effectively prevents the overfitting and improves the quantization performance.

1 Introduction

Quantizing a transformer model is not a simple matter due to numerous channel-dependent outliers in activations (Bondarenko et al., 2021). They lead to a large quantization error (Zhao et al., 2019), and we observe that the problem is worse in the decoder-only transformers like GPT-2. One solution to the difficulty is quantization-aware training (QAT), an approach that fine-tunes the model parameters in response to the numerical error arising from quantization. Post-training quantization (PTQ) – a counterpart of QAT that performs quantization without modifying model parameters – is not powerful enough to cope with the outliers.

While QAT is effective, it requires the dataset and the training pipeline, and the problem is that they are often inaccessible when dealing with the original pretrained model without any downstream task. One then cannot but use arbitrary fine-tuning data for QAT.

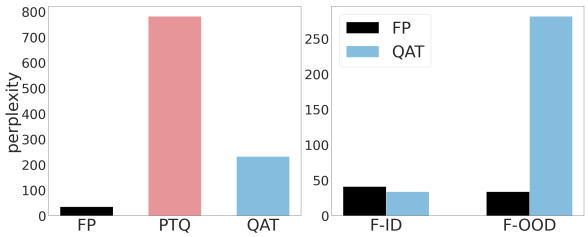

Figure 1: Average perplexity (PPL) of the full-precision (FP) model and the models quantized with PTQ and QAT on 5 datasets (left). We use the PTB dataset as the fine-tuning data (F-ID) for QAT. The FP model and the QAT model are evaluated on the F-ID and the other 4 datasets (F-OOD) (right).

However, the fine-tuning returns worse accuracies for distributions unseen during training (out-of-distribution with regard to fine-tuning; F-OOD) despite improving for the training distribution (in-distribution with regard to fine-tuning; F-ID) (Kumar et al., 2022). This is consistent with our observation that QAT overfits the model to the fine-tuning data as in Figure 1. The resulting quantized model therefore has its generality impaired. This violates the premise of a general-purpose language model, which must operate well across various texts of the target language.

Our hypothesis is that QAT incurs the overfitting because it changes all the parameters of the model. This difficulty is much like the research topic of continual learning, where it is important that a model should not forget its past capability when transferring to a new task (Zhang et al., 2021). Adapter is a strategy to adapt to a new distribution by training only a small number of parameters. It is a popular method to lessen the catastrophic forgetting. We borrow this concept to propose Quadapter, a lightweight module to adapt to the quantization error on behalf of the intact original model.

The contribution of this work is that we suc-

*Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

Findings of the Association for Computational Linguistics: EMNLP 2022, pages 2510–2517

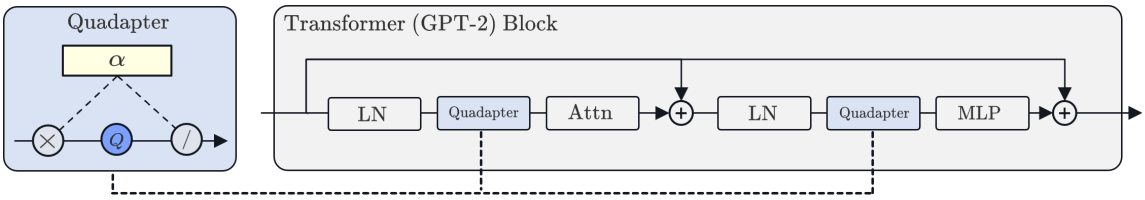

Figure 2: Quadapter performs a linear scaling and its inversion before and after Q, the quantizer for the target activation (left). In the transformer block of GPT-2, Quadapters can be installed in two different locations (right).

cessfully quantize GPT-2, overcoming the large inter-channel variance and the QAT overfitting issue with Quadapter. To the best of our knowledge, this is the first work to quantize both weights and activations of GPT-2 without the complete training pipeline.

2 Related Works

Adapters Extensive researches have been conducted on how to steer a large pretrained model with few adapter parameters. The concept of adapter has proven its usefulness in language models for transfer learning (Houlsby et al., 2019), multi-task learning (Stickland and Murray, 2019), and domain adaptation (Zhang et al., 2021). Several works apply adapters to the visual domain as well (Li and Hoiem, 2016; Perez et al., 2018).

Transformer Quantization In comparison to GPT-2, BERT is easier to quantize. It can be quantized with PTQ under a limited performance drop (Shen et al., 2020). QAT on BERT for a given downstream task recovers full-precision (FP) performance even with ultra-low precision (Zafrir et al., 2019; Bondarenko et al., 2021), or with integer-only operations for non-linear layers (Kim et al., 2021). On the other hand, quantization studies on autoregressive transformers are relatively limited in their scope, using weight-only quantization (Chung et al., 2020) or requiring full-fledged training (Prato et al., 2020; Tao et al., 2022). Please note that these works focus on quantizing GPT-2 that is finetuned on a downstream task whereas ours quantizes the original pretrained GPT-2.

Quantization techniques Directly relevant to our work are cross-layer-equalization (CLE) (Nagel et al., 2019) and adaptive rounding (AdaRound) (Nagel et al., 2020). Similarly to CLE, Qudapter rescales associated model weights to lessen the quantization burden. AdaRound and our proposed method are alike in training foldable helper parameters to minimize the block-wise quantization error. In addition, learned step size (LSQ) (Esser et al., 2020) and its extension (LSQ+) (Bhalgat et al., 2020) train the quantization-related parameters during QAT, to which Quadapter bears similarity.

3 Methods

Quadapter is simply a set of learnable parameters. On the other hand, the Quadapter block represents the actual working mechanism of Quadapter, involving two consecutive layers of linear relations, their quantizers, and their associated Quadapter instance. The effectiveness of Quadapter comes from the interaction amongst the involved components, and from the two-phase training procedure.

3.1 Quadapter Design

Quadatper linearly scales the input channels and reverts after quantization. This ensures the identity relation if not for quantizers, making it possible to keep the model parameters intact (Figure 2 left).

The scaling and the inverse-scaling of an activation are, in practice, folded to the weight and the bias of the preceding layer and to the weight of the following layer. For example, given a forward pass of two linear layers:

$$\mathbf{y} = \mathbf{W}_2(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2, \tag{1}$$

the Quadapter block output $\hat{\mathbf{y}}$ is as follows:

$$\begin{aligned}\hat{\mathbf{y}} &= Q_{\boldsymbol{\theta}_2}(\mathbf{W}_2\mathbf{A}^{-1})Q_{\boldsymbol{\theta}_a}(Q_{\boldsymbol{\theta}_1}(\mathbf{A}\mathbf{W}_1)\mathbf{x} \\ &\qquad + \mathbf{A}\mathbf{b}_1) + \mathbf{b}_2 \quad (2)\\ &= Q_{\boldsymbol{\theta}_2}(\mathbf{W}_2')Q_{\boldsymbol{\theta}_a}(Q_{\boldsymbol{\theta}_1}(\mathbf{W}_1')\mathbf{x} + \mathbf{b}_1') + \mathbf{b}_2. \quad (3)\end{aligned}$$

Here, $\mathbf{A} = \mathrm{diag}(\alpha)$ is a diagonal matrix with $\mathbf{A}_{ii} = \alpha_i$, where $\alpha \in \mathbb{R}^d$ is the learnable Quadapter parameter with the intermediate activation dimension d. $Q_{\boldsymbol{\theta}_1}$ and $Q_{\boldsymbol{\theta}_2}$ are the weight quantizers, and $Q_{\boldsymbol{\theta}_a}$ is the activation quantizer. Each quantizer $Q_{\boldsymbol{\theta}}$ quantizes its input values based on the quantization parameter $\boldsymbol{\theta} = (\theta_{\min}, \theta_{\max})$ (Krishnamoorthi, 2018). Quadapter α is trained during training and fused at the inference time (Equation 3).

Data	Method	GPT-2 Wikitext2	PTB	LAMBADA	CBT_CN	CBT_NE	DistilGPT-2 Wikitext2	PTB	LAMBADA	CBT_CN	CBT_NE
-	FP32	29.27	41.31	48.39	27.29	30.53	44.36	59.73	74.94	42.54	47.09
Calib. data	PTQ	915.58	751.23	827.06	655.31	759.83	87.52	114.42	205.35	93.16	104.94
	AdaRound	507.07	478.29	685.98	319.74	309.11	84.94	104.94	164.98	107.89	92.59
	CLE	40.28	59.33	74.61	38.92	43.69	69.81	86.66	144.06	68.78	76.80
	Quadapter BC	**34.53**	**50.65**	**63.51**	**32.47**	**36.46**	**52.79**	**70.43**	**102.75**	**51.97**	**57.81**
Wikitext2	QAT	32.51	100.75	125.40	54.94	63.94	35.04	109.40	129.19	67.03	76.55
	Quadapter BC+QAT	**21.61**	57.06	63.65	33.80	38.40	**28.50**	80.52	86.57	50.64	57.05
	Quadapter (ours)	29.34	**47.30**	**57.28**	**30.37**	**34.05**	43.05	**66.28**	**85.42**	**47.66**	**52.49**
PTB	QAT	331.61	33.94	330.10	212.12	252.03	347.25	37.44	308.22	214.14	257.44
	Quadapter BC+QAT	79.74	**24.10**	106.32	59.90	69.79	121.62	**29.65**	146.48	91.73	106.31
	Quadapter (ours)	**33.69**	39.46	**55.68**	**31.45**	**35.16**	**50.73**	56.63	**87.02**	**49.43**	**54.35**

Table 1: Performance evaluation of the quantized GPT-2 and DistilGPT-2 on various datasets. The metric is PPL (lower is better). In the case of Quadapter BC+QAT, QAT initiates after the block-wise calibration of Quadapter. For Quadapter (ours), both the training phases are completed. Underline indicates the results on F-ID

As in Equation 2, the forward scaling and the inverse scaling correspond across three nested quantizers that are strongly nonlinear operations. Therefore α should be learned rather than set analytically as in (Nagel et al., 2019); a single analytical solution is not sufficient to balance the quantization burden between the two layers.

3.2 Quadapter Training

The learning of Quadapter is comprised of two phases: the block-wise calibration and the end-to-end fine-tuning.

Phase 1: Block-wise Calibration Each of the Quadapter instances is initialized to $\vec{1}$ and trained with the calibration data, independently per Quadpter block. The local objective for each block is L2 loss:

$$\arg\min_{\alpha} ||\mathbf{y} - \hat{\mathbf{y}}||_2^2, \tag{4}$$

which (Nagel et al., 2020) shows to be effectively complementary to the task loss.

$\hat{\mathbf{y}}$ is computed in the dynamic quantization mode (Zafrir et al., 2019), where the statistics are obtained per batch. Quadapter resulting from the calibration phase is a PTQ method that is independent of the fine-tuning process. We therefore denote such Quadapter by *Quadapter BC*.

Phase 2: End-to-end Fine-tuning The subsequent fine-tuning starts with more accommodating quantization parameters (i.e. the min/max statistics) since they have moved to moderate values from extreme outliers during the first phase. The fine-tuning therefore converges much more quickly.

In the second phase, the statistics for quantization are computed in the fashion of static quantization (Zafrir et al., 2019), based on the same calibration data as in the first phase. Quadapter is then trained to minimize the end-to-end task loss. During the course, the quantization parameters are jointly learned as in (Bhalgat et al., 2020) while the model parameters stay fixed. Algorithm 1 details the full flow of the Quadatper training.

Algorithm 1: Quadapter training

input : pretrained model M, Quadapter blocks, calibration data D_1, fine-tuning data D_2, learning rates η_1, η_2.
output : Learned Quadapter parameters $\{\alpha_1, \alpha_2, ...\}$ and quantization parameters $\boldsymbol{\theta}^* = \{\boldsymbol{\theta}^1, \boldsymbol{\theta}^2, ...\}$.

```
/* Phase 1                                                    */
foreach i-th Quadapter block do
    Initialize α_i = 1⃗
    From M and D_1, gather block input x_i and
      output y_i
    while not converged do
        ŷ_i ← Eq. 2
        α_i ← α_i − η_1 ∇_{α_i} ||ŷ_i − y_i||_2^2
    end
end
/* Phase 2                                                    */
Apply learned Quadapters to M
Initialize θ* with D_1 to make quantized model M_Q
while not converged do
    for x, y ∈ D_2 do
        compute L_task(M_Q(x), y)
        foreach i-th Quadapter block do
            α_i ← α_i − η_2 ∇_{α_i} L_task
        end
        update θ* with LSQ+
    end
end
```

4 Experiments

Models We quantize GPT-2 (Radford et al., 2019) and DistilGPT-2 (Sanh et al., 2019) based on their huggingface pretrained models[1]. Our quantization configuration follows (Siddegowda et al., 2022), doing uniform asymmetric 8-bit quantization for both activations and weights. All the weights and

[1] huggingface.co/gpt2, huggingface.co/distilgpt2

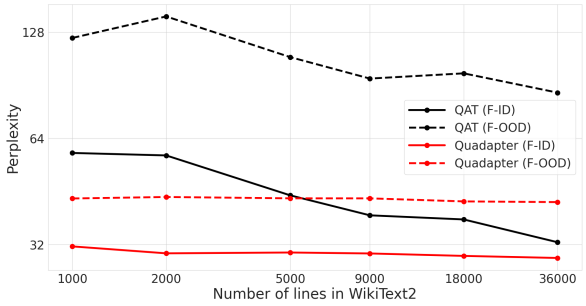

Figure 3: GPT-2 quantization performance when fine-tuned on F-ID of varying sizes. Both axes are logarithmic.

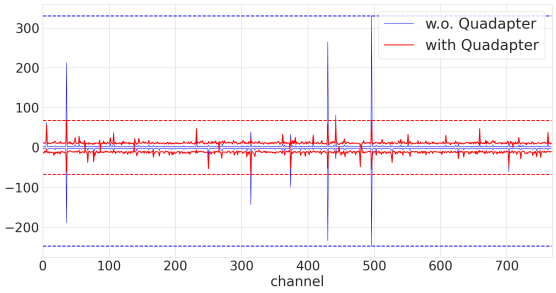

Figure 4: Visualization of the per-channel (x-axis) min/max (y-axis) values of the final layer norm output activation in GPT-2. The solid/dotted lines represent per-channel/total min and max.

activations are quantized, except for biases, non-linear opererations, and additions (Zafrir et al., 2019; Kim et al., 2021). For every transformer block, the Quadapter instances are installed in between the first layer norm and the linear projection for key/query/value as well as between the second layer norm and the first feed-forward network (Figure 2 right). One additional instance is applied to between the final layer norm and the logit projection.

Baseline methods Our implementation of LSQ+ follows the original proposition (Bhalgat et al., 2020), except for updating the min/max parameters for stability of training (Siddegowda et al., 2022). It is applied for all the QAT experiments. We use AI Model Efficiency Toolkit[2] to obtain AdaRound performance. The CLE metrics are computed with an untrained Quadapter, initialized analytically as in (Nagel et al., 2019).

Datasets We employ WikiText-2 (Merity et al., 2016), the English Penn Treebank (PTB) corpus (Marcus et al., 1993), the LAMBADA dataset (Paperno et al., 2016), and the named-entity subset (CBT_NE) as well as the common-noun subset (CBT_CN) of Children's Book Test (Hill et al., 2016). We follow the datasets' default divisions as to training/validation/test splits.

Experiment design To test the overfitting resiliency, GPT-2 and DistilGPT-2 are quantized with various PTQ and QAT methods on one of the five datasets. The resulting quantized model is evaluated on its F-ID and on the other four datasets (F-OOD). In addition, we expose the models to varying amounts of fine-tuning data during quantization to compare the changing behaviors of QAT and Quadapter.

Results In Table 1, Quadapter outperforms the baseline methods on the F-OOD in both GPT-2 and DistilGPT-2. This observation evinces the general capability of Quadapter to reduce overfitting across different models. The comparison between Quadapter (ours) and Quadapter BC+QAT is the ablation of the end-to-end finetuning, and the reusult proves its importance.

Noteworthy is that Quadapter is a powerful stand-alone PTQ technique. Even without QAT fine-tuning, the F-OOD metrics are better than those of the QAT baselines. In addition, the effectiveness of the calibration phase is shown by the comparison between CLE and Quadapter BC.

Another advantage of Quadapter is that it is a viable quantization option in data-scarce situations. As shown in Figure 3, Quadapter outperforms QAT throughout different amounts of fine-tuning data, and the gap is most evident when only a small amount of data is available.

Aside from the convincing metrics reported above, we further explore if Quadapter does the intended job of transforming an activation into a more uniform distribution. Figure 4 describes the per-channel statistics before and after the Quadapter training. Values in most activation dimensions except for few have small magnitudes around 0, and such dimensions lose precision when quantized because of the large magnitudes of total min/max before applying Quadpater. The illustration verifies that the effect of Quadapter indeed aligns with our expectation, reducing the ranges of outlier-ridden channels while enlarging the ranges of the others.

5 Limitations

One limitation of Quadapter is that it requires two consecutive layers of linear relations. In other words, it can be a mediator only for convolution layers, linear layers, or normalization layers (when

[2]https://github.com/quic/aimet

followed by a linear or convolution layer), but not if residual connections or nonlinear activation functions intervene.

6 Conclusions

We identify two challenges in quantizing autoregressive transformer language models: the overfitting issue of QAT and the inter-channel variance in activations. Through experiments, we demonstrate that Quadapter not only mitigates the two problems but also serves as an effective PTQ technique.

References

Yash Bhalgat, Jinwon Lee, Markus Nagel, Tijmen Blankevoort, and Nojun Kwak. 2020. Lsq+: Improving low-bit quantization through learnable offsets and better initialization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*.

Yelysei Bondarenko, Markus Nagel, and Tijmen Blankevoort. 2021. Understanding and overcoming the challenges of efficient transformer quantization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, (EMNLP)*.

Insoo Chung, Byeongwook Kim, Yoonjung Choi, Se Jung Kwon, Yongkweon Jeon, Baeseong Park, Sangha Kim, and Dongsoo Lee. 2020. Extremely low bit transformer quantization for on-device neural machine translation. In *Findings of the Association for Computational Linguistics (EMNLP)*, volume abs/2009.07453.

Steven K. Esser, Jeffrey L. McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S. Modha. 2020. Learned step size quantization. In *8th International Conference on Learning Representations (ICLR)*.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The goldilocks principle: Reading children's books with explicit memory representations.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*.

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, (ICML)*.

Raghuraman Krishnamoorthi. 2018. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint*, abs/1806.08342.

Ananya Kumar, Aditi Raghunathan, Robbie Jones, Tengyu Ma, and Percy Liang. 2022. Fine-tuning can distort pretrained features and underperform out-of-distribution. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhizhong Li and Derek Hoiem. 2016. Learning without forgetting. In *European Conference on Computer Vision (ECCV)*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Comput. Linguistics*, 19(2):313–330.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models.

Markus Nagel, Rana Ali Amjad, Mart van Baalen, Christos Louizos, and Tijmen Blankevoort. 2020. Up or down? adaptive rounding for post-training quantization. In *Proceedings of the 37th International Conference on Machine Learning, (ICML)*.

Markus Nagel, Mart Van Baalen, Tijmen Blankevoort, and Max Welling. 2019. Data-free quantization through weight equalization and bias correction. In *IEEE/CVF International Conference on Computer Vision (ICCV)*.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL*.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. 2018. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

Gabriele Prato, Ella Charlaix, and Mehdi Rezagholizadeh. 2020. Fully quantized transformer for machine translation. In *Findings of the Association for Computational Linguistics (EMNLP)*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. In *NeurIPS EMC2 Workshop*.

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. In *The Thirty-Fourth Conference on Artificial Intelligence (AAAI)*.

Sangeetha Siddegowda, Marios Fournarakis, Markus Nagel, Tijmen Blankevoort, Chirag Patel, and Abhijit Khobare. 2022. Neural network quantization with AI model efficiency toolkit (AIMET). *arXiv*, abs/2201.08442.

Asa Cooper Stickland and Iain Murray. 2019. BERT and pals: Projected attention layers for efficient adaptation in multi-task learning. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*.

Chaofan Tao, Lu Hou, Wei Zhang, Lifeng Shang, Xin Jiang, Qun Liu, Ping Luo, and Ngai Wong. 2022. Compression of generative pre-trained language models via quantization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics, ACL*.

Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. 2019. Q8BERT: quantized 8bit BERT. In *Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing (EMC2@NeurIPS)*.

Rongsheng Zhang, Yinhe Zheng, Xiaoxi Mao, and Minlie Huang. 2021. Unsupervised domain adaptation with adapter. In *35th Conference on Neural Information Processing Systems (NeurIPS),*.

Ritchie Zhao, Yuwei Hu, Jordan Dotzel, Christopher De Sa, and Zhiru Zhang. 2019. Improving neural network quantization without retraining using outlier channel splitting. In *Proceedings of the 36th International Conference on Machine Learning, (ICML)*.

Appendix

A Additional Experiments

F-ID Expansion In Table 1, we limit the F-ID to one amongst the five available datasets. Here, we perform an additional experiment by expanding the F-ID to include four of them, limiting the F-OOD to the one remaining dataset. The results are in Table 2. Comparing the metrics on WikiText2 when the QAT model is fine-tuned on PTB (Table 1) and when fine-tuned on all but WikiText2 (Table 2), we can observe the improvement of Quadapter's F-OOD performance. On the other hand, QAT still suffers from overfitting despite the expanded fine-tuning data.

Ablation of LSQ+ In (Bhalgat et al., 2020), LSQ+ is a composite method that includes initialization of weight quantizer, model parameter training, and quantization parameter ($\boldsymbol{\theta}$) training. However, in our work, we isolate the quantization parameter training and denote it by LSQ+. In Table 1, QAT is accompanied by LSQ+, thus training both the model parameters and the quantization parameters. We ablate LSQ+ in Table 3. The results show that LSQ+ tends to improve the quantization performance in general, particularly in conjunction with our proposed method.

Quadapter BC effectiveness on QAT As discussed in the main text, QAT makes the model overfit to F-ID and perform poorly on F-OOD. However, when employed with Quadapter BC (i.e. Quadapter BC+QAT), the QAT training process is stabilized, and so the quantized model reaches near the upper bound of the fine-tuned FP model (Figure 5). This shows that Quadapter fosters QAT.

B Hyperparameter Details

Block-wise Calibration We use 10 calibration data with the max time length (block size) of 512, yielding 5120 text tokens in total. The same calibration data is used for all the PTQ and QAT experiments. The initial learning rate is 0.1, and it decays at the rate of 0.2 every 100 steps. The training takes place for 500 steps with the Adam optimizer. Figure 6 shows the convergence of training in two of the GPT-2 Quadapter blocks: the very first layer norm and the final layer norm. The total block-wise calibration phase takes approximately 2 minutes with RTX 2080 Ti, when training all the Quadatper blocks in GPT-2 sequentially from bottom to top.

End-to-end Fine-tuning We use the initial learning rates 1e-5, 1e-3, and 1e-3 for model parameters, Quadapter α, and quantization parameters $\boldsymbol{\theta}$, respectively. The learning rate linearly decreases to 0 over 10,000 training steps. The batch size is set to 4 with the max time length of 512. All the QAT methods follow this training scheme. After the completion of training, PPL is measured with the max time length of 1024. All of the PPL metrics reported in this work share this configuration.

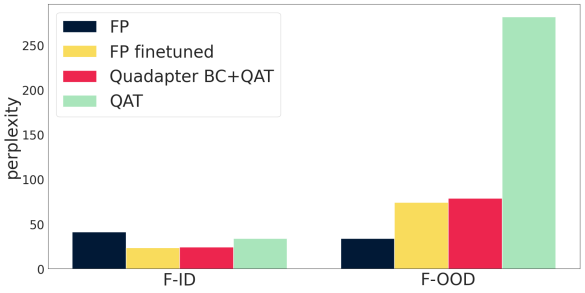

Figure 5: Comparison of the fine-tuned FP model (FP finetuned) with other methods. PTB is the F-ID, and the other 4 datasets are the F-OOD.

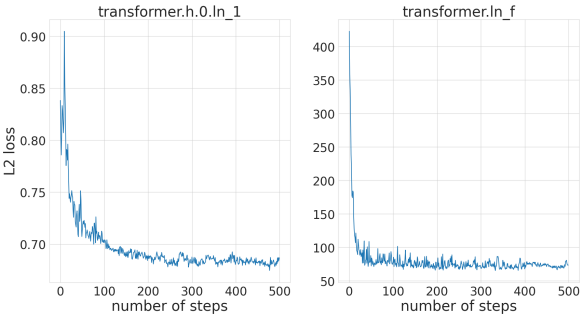

Figure 6: Block-wise calibration learning curves of the two selected Quadapter blocks in GPT-2.

C Implementation Details

Quantization Implementation The quantizer function $Q_{\boldsymbol{\theta}}$ is defined as follows:

$$Q_{\boldsymbol{\theta}}(x) = s \cdot (clip(\left\lfloor \frac{x}{s} + o \right\rceil, 0, 2^b - 1) - o), \quad (5)$$

$$s = \frac{\theta_{max} - \theta_{min}}{2^b - 1}, \quad o = \left\lfloor -\frac{\theta_{min}}{s} \right\rceil, \quad (6)$$

where s and o are the scale and offset, and b is the target bit depth (8-bit in our case). $\lfloor \cdot \rceil$ is a rounding function, and $clip(\mathbf{x}, n, p)$ clamps all the values between n and p from the input $\mathbf{x}$.

In the first calibration phase, θ_{min} and θ_{max} are obtained from the batch statistics at each inference step (i.e. dynamic quantization). In the second fine-tuning phase, the parameters are initially set based

	Wikitext2	PTB	LAMBADA	CBT_CN	CBT_NE	F-ID	F-OOD
FP	29.28	41.31	48.39	27.29	30.53	36.88	29.28
QAT	61.57	31.95	39.60	22.67	24.83	29.76	61.57
Quadapter BC+QAT	41.13	25.37	34.05	19.79	21.56	25.19	41.13
Quadapter (ours)	31.71	44.83	45.79	27.23	30.07	36.98	31.71

Table 2: PPL measurements of GPT-2 for expanded F-ID. Wikitext2 is the F-OOD, and the other 4 datasets are the F-ID. The average PPL is reported in the columns, F-ID and F-OOD.

Data	Method	Wikitext2	PTB	LAMBADA	CBT_CN	CBT_NE
Wikitext2	QAT	36.28 (3.77)	101.81 (1.07)	134.80 (9.40)	58.77 (3.83)	68.67 (4.73)
	Quadapter BC+QAT	21.63 (0.02)	57.40 (0.34)	61.72 (-1.93)	33.48 (-0.32)	38.02 (-0.38)
	Qudapter (ours)	32.69 (3.35)	49.01 (1.71)	59.59 (2.31)	31.44 (1.07)	35.27 (1.22)
PTB	QAT	275.80 (-55.81)	37.87 (3.93)	284.61 (-45.49)	165.75 (-46.37)	194.71 (-57.32)
	Quadapter BC+QAT	80.00 (0.26)	23.93 (-0.17)	101.61 (-4.71)	59.95 (0.05)	69.78 (-0.01)
	Quadapter (ours)	33.79 (0.10)	46.98 (7.52)	59.46 (3.78)	31.54 (0.09)	35.37 (0.21)

Table 3: PPL measurements of GPT-2 trained without LSQ+. The differences from the counterparts with LSQ+ in Table 1 are noted in the parenthesis (a positive value indicates LSQ+'s performance gain).

on the calibration data (i.e. static quantization), and trained afterwards. When $\boldsymbol{\theta}$ is trained, the gradients are passed through the rounding function via straight-through-estimation.

Quadapter Implementation The details of the actual application of Quadapter to GPT-2 is slightly different from the general form of Quadapter block in Equation 2. In the transformer block of GPT-2, $\mathbf{W}_1$ denotes only the affine transformation, but not the preceding normalization of the layer norm operation. For example, we can define the layer norm operation as follows:

$$\mathbf{y}_l = \frac{\mathbf{x}_l - \mu(\mathbf{x}_l)}{\sigma(\mathbf{x}_l)} \odot \gamma + \beta, \tag{7}$$

then $\mathbf{W}_1 = \gamma, \mathbf{b}_1 = \beta$, and the input of Quadapter block is $(\mathbf{x}_l - \mu(\mathbf{x}_l))/\sigma(\mathbf{x}_l)$. The fused weight is computed as $\mathbf{W}_1' = \alpha \odot \gamma$ with element-wise multiplication $\odot$.

D Constraint of Two Linear Layers

While stating that Quadapter is applicable only to two consecutive layers of linear nature in Section 5, the main text omits the discussion of piece-wise linear activation function for brevity. For an operation f that meets the following scaling-invariant condition:

$$f(sx) = sf(x), \tag{8}$$

the identity relation between the scaling and the inverse-scaling of Quadapter still holds. Therefore, it is possible to install Quadapter through piece-wise linear functions (e.g. ReLU, leaky ReLU, PReLU, etc.) as well.

Our future goal is to further expand the Quadapter applicability. We thus plan to investigate if Quadapter would be applicable even with an intervening nonlinear activation function (e.g. GeLU, tanh, etc.) by enhancing expressivity (i.e. setting up additional learnable scalar variables for the inverse scaling). In addition, by scaling all the tensors involved in a residual connection, we expect to apply Quadapter even in the presence of a residual connection in between the two target layers.