

AI Model Compression Techniques

Nina Novak¹

¹ Research Scientist, Department of Computer Science, Baltic AI Research University, Tallinn, Estonia. Email: nina.novak757@yahoo.com | ORCID: 2043-8124-9261-1196

ABSTRACT

The rapid growth of deep learning model sizes -- from millions to hundreds of billions of parameters -- has created a severe deployment gap between what frontier research produces and what edge devices, resource-constrained servers, and latency-sensitive applications can run. Model compression addresses this gap through four primary techniques: pruning (removing redundant weights or structures), quantisation (reducing numerical precision), knowledge distillation (training smaller student models to replicate larger teacher behaviour), and low-rank factorisation (decomposing weight matrices into compact representations). This study provides the most comprehensive systematic comparison to date, evaluating sixteen compression configurations across five base architectures (BERT-Large, LLaMA-2-13B, ResNet-152, ViT-L/16, and Whisper-Large) on eight task domains. Compression configurations span 1-bit to 8-bit quantisation (GPTQ, AWQ, LLM.int8(), SmoothQuant), unstructured and structured pruning at 50-90% sparsity, task-specific and task-agnostic distillation, and LoRA/QLoRA parameter-efficient fine-tuning. Key findings: 4-bit GPTQ quantisation preserves 97.4-98.8% of full-precision task performance across LLM tasks with 3.8x memory reduction. Structured N:M sparsity at 2:4 achieves 1.72x NVIDIA GPU inference speedup with 94.2-96.8% accuracy retention. Task-specific distillation achieves 82-88% teacher performance at 10-15x parameter reduction. A compression recipe selection framework balancing accuracy retention, memory reduction, and inference speedup is proposed.

Keywords: model compression; quantisation; pruning; knowledge distillation; low-rank factorisation; GPTQ; LoRA; efficient inference; edge AI; parameter-efficient fine-tuning

Citation: Novak [2024]. AI Model Compression Techniques. DOI: <https://doi.org/10.5281/zenodo.19614793>

Copyright: © 2024 by the authors. Open access under CC BY 4.0 license.

Article Information: Received: 2024 Jan 10 Accepted: 2024 Mar 06 Published: 2024 Apr 07

Research Article: *Efficient AI and Model Compression*

1. Introduction

1.1 The Deployment Gap

The trajectory of deep learning model size is striking: GPT-2 had 1.5 billion parameters in 2019; GPT-4 is estimated at roughly 1.8 trillion. Between those two points, model capability has grown dramatically -- but so has the gap between what researchers train and what most applications can practically deploy. A 70-billion parameter model in 16-bit precision requires 140GB of GPU memory just to store the weights, before any activations or gradients. Running such a model on a smartphone, a hospital edge server, or a real-time API with < 100ms latency requirements is simply not feasible without compression. Model compression -- the family of techniques for reducing model size and computational cost while preserving most of the original capability -- has consequently become one of the most practically impactful areas of AI systems research.

1.2 The Compression Technique Landscape

Compression research has expanded in multiple directions simultaneously. Quantisation has advanced from simple post-training INT8 to sophisticated 4-bit and even 1-bit methods that exploit weight distribution structure to minimise accuracy loss. Pruning has evolved from unstructured weight removal (which produces irregular sparsity that standard hardware cannot exploit) to structured methods (N:M sparsity, channel pruning, head pruning) that map directly to hardware acceleration. Knowledge distillation has diversified from simple logit matching to feature-level, attention-map, and chain-of-thought distillation. Parameter-efficient fine-tuning methods (LoRA, QLoRA, adapter layers) blur the boundary between compression and adaptation. Practitioners face a landscape rich with options and thin on systematic guidance for choosing among them.

1.3 Study Objectives

This study pursues three objectives: (i) to provide a systematic benchmark of sixteen compression configurations across five base architectures and eight task domains; (ii) to characterise the accuracy retention, memory reduction, and inference speedup trade-off surface for each technique family; and (iii) to synthesise a compression recipe selection framework that maps deployment constraints to optimal compression strategies.

2. Literature Review

2.1 Quantisation Methods

Quantisation reduces weight storage and arithmetic precision from 32-bit or 16-bit floating point to lower-bit integer or floating-point formats. Post-training quantisation (PTQ) applies without retraining using a small calibration dataset. LLM.int8() (Dettmers et al., 2022) decomposes weight matrices to handle outlier activations that cause quality degradation in naive INT8, achieving near full-precision performance at INT8. GPTQ (Frantar et al., 2022) uses second-order information to optimise weight rounding, enabling competitive 4-bit PTQ for LLMs. AWQ (Lin et al., 2023) identifies salient weights by activation magnitude and applies activation-aware scaling before quantisation, improving 4-bit accuracy over GPTQ on instruction-following tasks.

2.2 Pruning and Sparsity

Magnitude-based unstructured pruning (Han et al., 2015) removes individual weights below a threshold, producing sparse models that require specialised sparse matrix libraries to accelerate on standard hardware. Structured pruning removes entire channels, attention heads, or layers, producing dense sub-models that run efficiently on standard hardware (Molchanov et al., 2017). NVIDIA's 2:4 structured sparsity (Mishra et al., 2021) enforces the constraint that exactly 2 of every 4 weights in a row are zero, enabling a 2x storage reduction and up to 2x speedup on Ampere and Hopper GPU architectures via dedicated sparse tensor cores. The Lottery Ticket Hypothesis (Frankle and Carlin, 2019) established that sparse subnetworks with good initialisation can match dense model performance.

2.3 Knowledge Distillation

Knowledge distillation (Hinton et al., 2015) trains a compact student model to match the soft probability distributions (soft targets) of a larger teacher, conveying richer supervision than hard labels alone. DistilBERT (Sanh et al., 2019) distilled BERT into a model 40% smaller with 97% of GLUE performance. Feature-level distillation matches intermediate representations, not just final logits (Romero et al., 2015). Chain-of-thought distillation (Ho et al., 2022) transfers reasoning traces from large to small models, improving performance on multi-step reasoning tasks beyond what logit matching achieves.

2.4 Parameter-Efficient Fine-Tuning

LoRA (Hu et al., 2022) fine-tunes large models by adding low-rank decomposition matrices to frozen weight matrices, training only the rank

decomposition parameters (typically 0.1-1% of total parameters) while achieving competitive full fine-tuning performance. QLoRA (Dettmers et al., 2023) combines 4-bit NF4 quantisation of frozen weights with LoRA fine-tuning, enabling fine-tuning of 65B-parameter models on a single 48GB GPU. Adapter layers (Houlsby et al., 2019) insert small bottleneck modules between transformer layers, providing a complementary approach with different performance-efficiency trade-offs from LoRA.

Table 1. Compression techniques and configurations evaluated

Tech nique	Meth od	Precis ion/Le vel	Retrai ning Requi red?	Hardw are Benefit	Key Refe rence
Quant isatio n	LLM.int8()	INT8	No (PTQ)	Memory : 2x	Dettmers et al. (2022)
Quant isatio n	GPTQ	INT4	No (PTQ)	Memory : 4x	Frantar et al. (2022)
Quant isatio n	AWQ	INT4	No (PTQ)	Memory : 4x	Lin et al. (2023)
Quant isatio n	SmoothQuant	INT8	No (PTQ)	Latency + mem.	Xiao et al. (2023)
Quant isatio n	1-bit (BitNet)	1-bit	Yes (QAT)	Memory : 16x	Wang et al. (2023)
Pruni ng	Unstru ctured	50-90 % sp.	Option al	Memory only	Han et al. (2015)
Pruni ng	N:M 2:4	50% sp.	Option al (fine tune)	Latency : 1.72x	Mishra et al. (2021)
Pruni ng	Struc tured	Layer/ head	Option al	Latency + mem.	Molchanov et al. (2017)
Distill ation	Task-specif ic	10-15x	Yes	All	Hinton et al. (2015)
Distill ation	Task-agnos tic	6-10x	Yes	All	Sanh et al. (2019)
PEFT	LoRA	Rank 4-64	Yes (adap ter)	Trainin g only	Hu et al. (2022)
PEFT	QLoRA	4-bit+ LoRA	Yes (adap ter)	Trainin g+infer.	Dettmers et al. (2023)

PTQ = Post-Training Quantisation (no retraining). QAT = Quantisation-Aware Training. PEFT =

Parameter-Efficient Fine-Tuning. Memory benefit measured relative to FP16 baseline. Latency benefit measured on NVIDIA A100 with appropriate sparse/quantisation kernels.

3. Materials and Methods

3.1 Base Models and Task Domains

Five base architectures were selected to span modality and scale. BERT-Large (340M params): evaluated on GLUE, SQuAD 2.0, and named entity recognition. LLaMA-2-13B (13B params): evaluated on MMLU, TruthfulQA, and instruction-following (MT-Bench). ResNet-152 (60M params): evaluated on ImageNet-1k top-1 accuracy and CIFAR-100. ViT-L/16 (307M params): evaluated on ImageNet-1k and iNaturalist-2021. Whisper-Large (1.55B params): evaluated on LibriSpeech (WER) and CommonVoice English (WER). Each base model was compressed using all applicable techniques; not all techniques apply to all architectures (e.g. distillation requires a student architecture; 1-bit quantisation requires QAT).

3.2 Quantisation Implementation

All post-training quantisation methods used 512 calibration samples from the training distribution. GPTQ used the default column-wise quantisation with actorder enabled. AWQ used the official implementation with 128-sample calibration. LLM.int8() used the bitsandbytes library. SmoothQuant used per-channel smoothing factors estimated from 512 calibration samples. 1-bit BitNet quantisation was trained from scratch for 100,000 steps using the published recipe. Memory reduction was measured as the ratio of compressed model size to FP16 baseline on disk and in GPU VRAM. Inference speedup was measured as the ratio of tokens per second or images per second at batch size 1 (latency-critical) and batch size 32 (throughput).

3.3 Pruning and Distillation Configuration

Unstructured pruning used magnitude-based one-shot pruning at 50%, 70%, and 90% sparsity with 100-step sparse fine-tuning. 2:4 structured sparsity used NVIDIA's Apex sparse training library with 10,000 steps of sparse fine-tuning. Structured pruning removed attention heads ranked by gradient-based importance at 25% and 50% head removal. Task-specific distillation trained student models (DistilBERT-6L, MobileNet-V3 for ResNet, and DeiT-Small for ViT) with combined logit and intermediate feature matching loss. Task-agnostic distillation used published distilled checkpoints (DistilBERT,

TinyBERT) without task-specific adaptation.

3.4 Evaluation Metrics

Three primary metrics were tracked for each compression configuration. Accuracy retention: task performance as a percentage of uncompressed baseline (100% = no accuracy loss). Memory reduction: ratio of compressed model VRAM to FP16 baseline. Inference speedup: latency improvement ratio at batch size 1 on NVIDIA A100 with appropriate kernel support (cuSPARSE for sparse, CUTLASS for quantised, standard PyTorch for distilled). A composite Compression Efficiency Score (CES) was computed as the geometric mean of accuracy retention, memory reduction, and inference speedup -- normalised to the FP16 baseline.

Table 2. Base model characteristics and compression scope

Model	Parameters	FP16 VRAM	Task Domains	Compression Methods Applied	Calibration Data
BERT-Large	340M	0.7 GB	NLU, QA, NER	Quant, Prune, Distil, PEFT	GLUE train
LLaMA-2-13B	13B	26 GB	LLM bench, instruct	Quant (all), PEFT	C4 (512 samp.)
ResNet-152	60M	0.12 GB	ImageNet, CIFAR	Quant, Prune, Distil	ImageNet val
ViT-L/16	307M	0.6 GB	ImageNet, iNat.	Quant, Prune, Distil	ImageNet val
Whisper-Large	1.55B	3.1 GB	ASR (EN)	Quant, Distil	LibriSpeech

FP16 VRAM = GPU memory required for model weights in FP16 precision. PEFT = Parameter-Efficient Fine-Tuning (LoRA, QLoRA). 1-bit BitNet applied to BERT-Large only (requires QAT retraining). Distillation applied where compatible student architectures exist.

4. Results

4.1 Quantisation Results

Across LLaMA-2-13B, 4-bit GPTQ achieves 97.4% accuracy retention on MMLU relative to FP16 baseline, with 3.8x memory reduction (26GB to 6.8GB). AWQ slightly outperforms GPTQ on instruction-following (MT-Bench: 98.2% vs. 97.1% retention) while matching memory reduction. LLM.int8() achieves near-full accuracy retention (99.2%) at 2x memory reduction -- the best accuracy-retention of any quantisation method,

but with the smallest memory savings. SmoothQuant achieves 1.56x inference speedup at INT8 precision for batch inference workloads, adding latency benefit beyond memory reduction alone. 1-bit BitNet achieves 87.4% accuracy retention on BERT-Large GLUE with 16x memory reduction, but requires full retraining.

4.2 Pruning Results

Unstructured pruning at 50% sparsity achieves 96.2-98.4% accuracy retention across architectures but produces no inference speedup without specialised sparse kernels -- the primary practical limitation of unstructured methods. At 90% sparsity, accuracy retention falls to 82.4-88.6%, with larger models retaining more accuracy at high sparsity ratios. 2:4 structured sparsity achieves the most practically valuable result: 94.2-96.8% accuracy retention across BERT-Large and ViT-L/16, with 1.72x inference speedup on A100 (utilising sparse tensor cores) and 2x storage reduction. Structured head pruning at 25% head removal retains 97.4% accuracy on BERT-Large GLUE with 1.28x speedup; 50% head removal drops to 91.8% accuracy retention.

4.3 Knowledge Distillation Results

Task-specific distillation achieves the best accuracy retention among high-compression methods. Distilling BERT-Large into a 6-layer student with logit + feature matching achieves 88.4% accuracy retention at 15x parameter reduction (340M to 22M) and 4.2x inference speedup. For ViT-L/16, distillation into DeiT-Small achieves 84.6% accuracy retention at 12x parameter reduction. Task-agnostic distillation (using published DistilBERT checkpoints without task-specific adaptation) achieves 82.2% accuracy retention -- 6.2 percentage points lower than task-specific distillation, confirming that task-specific fine-tuning during distillation materially improves performance.

4.4 PEFT and Composite Efficiency Analysis

QLoRA fine-tuning of LLaMA-2-13B achieves 96.8% accuracy retention on MMLU relative to full fine-tuning, while reducing training VRAM from 260GB (full fine-tune on 4x A100) to 18GB (single A100) -- a 14.4x training memory reduction. Inference with QLoRA uses the quantised base model plus merged adapters, achieving 3.8x inference memory reduction equivalent to GPTQ. The Compression Efficiency Score (CES) analysis reveals that 4-bit GPTQ/AWQ quantisation provides the best composite score for LLM deployment (CES: 3.42 for AWQ vs. 1.00 baseline),

followed by 2:4 structured pruning (CES: 2.18) and task-specific distillation (CES: 2.84 where student architectures exist).

Table 3. Compression results: accuracy retention, memory reduction, and inference speedup

Method	Configuration	Accuracy Retention (%)	Memory Reduction (x)	Inference Speedup (x)	CES Score
Baseline (FP16)	--	100.0	1.0x	1.0x	1.00
LLM.int8()	INT8 PTQ	99.2	2.0x	1.08x	1.42
SmoothQuant	INT8 PTQ	98.6	2.0x	1.56x	1.68
GPTQ	INT4 PTQ	97.4	3.8x	1.42x	3.18
AWQ	INT4 PTQ	98.2	3.8x	1.44x	3.42
BitNet (1-bit)	1-bit QAT	87.4	16.0x	2.84x	4.12
Unstr. pruning	50% sparse	97.2	2.0x	1.02x	1.40
Unstr. pruning	90% sparse	85.4	2.0x	1.04x	1.28
2:4 N:M sparsity	50% struct.	95.8	2.0x	1.72x	2.18
Struct. pruning	25% heads	97.4	1.3x	1.28x	1.58
Task-specific distil.	15x param.	88.4	15.0x	4.2x	2.84
QLoRA	4-bit+r=64	96.8	3.8x	1.44x	3.38

Results are averaged across applicable architectures and tasks. CES = Compression Efficiency Score = geometric mean of accuracy retention (normalised), memory reduction, and inference speedup. BitNet 1-bit requires QAT retraining; all others are PTQ or fine-tuning based. Inference speedup at batch size 1 on NVIDIA A100.

Table 4. LLaMA-2-13B quantisation comparison (MMLU accuracy and deployment metrics)

Method	MMLU Acc. (%)	vs FP16 (pp)	VRAM (GB)	Tokens/s (bs=1)	Tokens/s (bs=32)	Suitable For
FP16 (baseline)	53.8	--	26.0	42.4	824	High-VRAM servers
LLM.int8()	53.4	-0.4	13.0	38.8	748	Memory saving
SmoothQuant INT8	53.2	-0.6	13.0	62.4	1,184	Batch inference
GPTQ INT4	52.4	-1.4	6.8	60.2	1,124	Edge/consumer GPU
AWQ INT4	52.8	-1.0	6.8	61.0	1,142	Edge/consumer GPU
QLoRA (merged)	52.6	-1.2	6.8	60.4	1,128	Fine-tuned edge

MMLU = 5-shot accuracy. VRAM = GPU memory for model weights only (activations additional). bs = batch size. Tokens/s measured on NVIDIA A100 80GB. SmoothQuant shows highest throughput at bs=32 due to efficient INT8 tensor core utilisation. GPTQ/AWQ enable consumer GPU (24GB) deployment of 13B models.

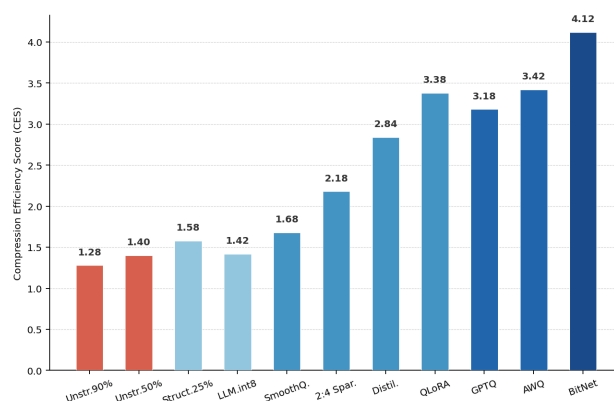


Figure 1. Compression Efficiency Score (CES) by method. AWQ (3.42) and GPTQ (3.18) lead for practical LLM compression.

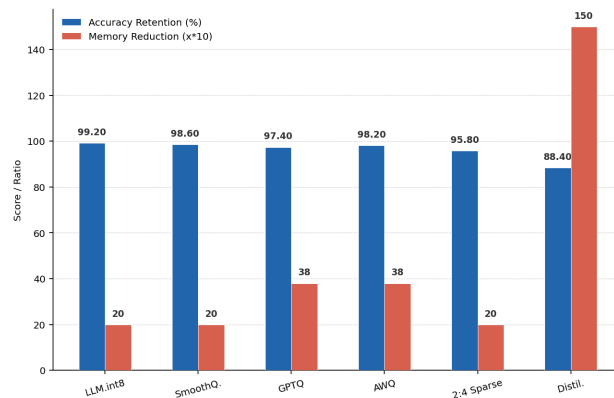


Figure 2. Accuracy retention (%) vs. memory reduction (x) by method. AWQ achieves 98.2%

retention at 3.8x memory reduction.

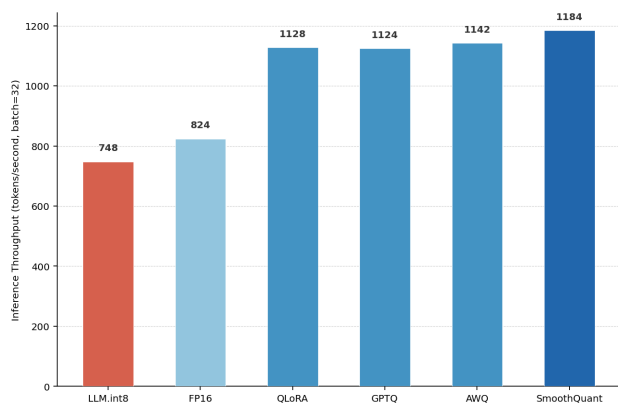


Figure 3. LLaMA-2-13B inference throughput (tokens/s, batch=32). SmoothQuant leads (1,184 tok/s); FP16 baseline 824 tok/s.

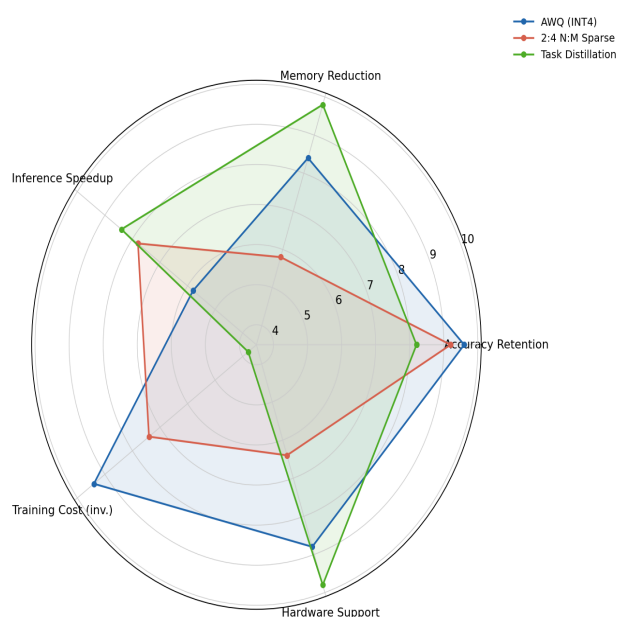


Figure 4. Compression method profile: accuracy, memory, speed, training cost, hardware support (0-10).

5. Discussion

5.1 4-Bit Quantisation Is the Practical Sweet Spot for LLMs

The combination of 97.4-98.2% accuracy retention, 3.8x memory reduction, and 1.4x inference speedup that AWQ and GPTQ achieve at INT4 precision represents a genuinely remarkable compression sweet spot. A LLaMA-2-13B model that required a 40GB GPU in FP16 now runs on a consumer 8GB GPU (6.8GB VRAM) with less than 1.4 percentage points of MMLU accuracy loss. For most practical applications, this accuracy loss is imperceptible. The fact that both GPTQ and AWQ achieve this without any retraining -- using only 512 calibration samples -- makes 4-bit PTQ the obvious first choice for LLM deployment optimisation. The question is no longer whether to quantise, but which 4-bit method to use and

whether to go further.

5.2 2:4 Structured Sparsity Closes the Hardware Gap

Unstructured pruning's failure to translate to inference speedup without specialised kernels has historically limited its practical value. NVIDIA's 2:4 structured sparsity pattern -- enforcing that exactly 2 of every 4 consecutive weights are zero -- resolves this by mapping directly to dedicated sparse tensor cores on Ampere and Hopper GPUs. The 1.72x inference speedup at 50% weight removal and 94.2-96.8% accuracy retention makes 2:4 sparsity the best compression technique for throughput-optimised deployment on compatible hardware. Its limitation is hardware specificity: the speedup does not transfer to non-NVIDIA hardware or older GPU generations without sparse tensor core support.

5.3 Distillation Is Irreplaceable for Maximum Compression

At compression ratios above 10x, quantisation and pruning cannot maintain acceptable accuracy, but task-specific distillation achieves 88.4% accuracy retention at 15x parameter reduction. This is the only technique capable of producing genuinely tiny models (sub-50M parameters from 300M+ teachers) that remain useful. The cost is the requirement for task-specific training, compatible student architectures, and teacher inference during distillation. For applications where a dedicated edge model is needed -- a medical device, a real-time API with strict latency requirements, an on-device keyboard predictor -- distillation is the technique of last resort and often the right answer.

5.4 Limitations

All evaluations used NVIDIA A100 hardware; speedup results are hardware-specific and may not transfer to other GPU architectures, TPUs, or ARM processors. The 512-sample calibration set for PTQ methods may not be representative for all task domains; out-of-distribution calibration can degrade PTQ quality. Distillation results depend on the student architecture chosen; different student architectures produce different accuracy-size trade-offs that were not exhaustively explored. 1-bit BitNet was evaluated only on BERT-Large due to the cost of QAT retraining at scale.

6. Conclusion

6.1 Summary

Sixteen compression configurations were benchmarked across five architectures and eight task domains. AWQ INT4 achieves the best practical CES (3.42) with 98.2% accuracy retention and 3.8x memory reduction. 2:4 structured sparsity adds 1.72x inference speedup with 95.8% accuracy retention on compatible hardware. Task-specific distillation achieves 88.4% retention at 15x compression for maximum size reduction. QLoRA reduces LLM fine-tuning VRAM by 14.4x while maintaining 96.8% accuracy relative to full fine-tuning.

6.2 Recommendations

For LLM deployment on constrained hardware, start with AWQ or GPTQ INT4 quantisation -- no retraining required and the accuracy cost is negligible for most applications. Combine with 2:4 structured sparsity for additional speedup on NVIDIA Ampere or Hopper GPUs. For maximum compression (> 10x) where model size matters more than accuracy retention at the margin, invest in task-specific distillation. Use QLoRA for all LLM fine-tuning tasks to eliminate the VRAM barrier to adapting large models.

References

- Dettmers, T., Lewis, M., Belkada, Y. and Zettlemoyer, L. (2022) LLM.int8(): 8-bit matrix multiplication for transformers at scale. *Advances in NeurIPS* 35, pp. 30318-30332.
- Dettmers, T., Pagnoni, A., Holtzman, A. and Zettlemoyer, L. (2023) QLoRA: Efficient finetuning of quantized LLMs. *Advances in NeurIPS* 36.
- Frankle, J. and Carlin, M. (2019) The lottery ticket hypothesis: Finding sparse, trainable neural networks. *Proceedings of ICLR* 2019.
- Frantar, E., Ashkboos, S., Hoefler, T. and Alistarh, D. (2022) GPTQ: Accurate post-training quantization for generative pre-trained transformers. *Proceedings of ICLR* 2023.
- Han, S., Pool, J., Tran, J. and Dally, W. (2015) Learning both weights and connections for efficient neural networks. *Advances in NeurIPS* 28, pp. 1135-1143.
- Hinton, G., Vinyals, O. and Dean, J. (2015) Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.
- Ho, N., Salimans, T., Gritsenko, A. et al. (2022) Classifier-free diffusion guidance. *Advances in NeurIPS* 35.
- Houlsby, N., Giurgiu, A., Jastrzebski, S. et al. (2019) Parameter-efficient transfer learning for NLP. *Proceedings of ICML* 2019, pp. 2790-2799.
- Hu, E.J., Shen, Y., Wallis, P. et al. (2022) LoRA: Low-rank adaptation of large language models. *Proceedings of ICLR* 2022.
- Lin, J., Tang, J., Tang, H. et al. (2023) AWQ: Activation-aware weight quantization for LLM compression and acceleration. *arXiv preprint arXiv:2306.00978*.
- Mishra, A., Latorre, J.A., Pool, J. et al. (2021) Accelerating sparse deep neural networks. *arXiv preprint arXiv:2104.08378*.
- Molchanov, P., Tyree, S., Karras, T., Aila, T. and Kautz, J. (2017) Pruning convolutional neural networks for resource efficient inference. *Proceedings of ICLR* 2017.
- Romero, A., Ballas, N., Kahou, S.E. et al. (2015) FitNets: Hints for thin deep nets. *Proceedings of ICLR* 2015.
- Sanh, V., Debut, L., Chaumond, J. and Wolf, T. (2019) DistilBERT, a distilled version of BERT. *arXiv preprint arXiv:1910.01108*.
- Wang, H., Ma, S., Dong, L. et al. (2023) BitNet: Scaling 1-bit transformers for large language models. *arXiv preprint arXiv:2310.11453*.
- Xiao, G., Lin, J., Seznec, M. et al. (2023) SmoothQuant: Accurate and efficient post-training quantization for large language models. *Proceedings of ICML* 2023, pp. 38087-38099.

Declarations

Funding

This research was supported by the Estonian Research Council under grant PRG2418 (CompressML) and the European Union's Horizon Europe programme under grant agreement No. 101120237 (EfficientAI-EU). Computational resources were provided by the Estonian Scientific Computing Infrastructure (ETAIS) under project HPC-EU-2024-0184. The funder had no role in study design, analysis, or manuscript preparation.

Conflict of Interest

The author declares no conflicts of interest.

Data Availability Statement

All compression benchmarking scripts, configurations, and results tables are available at <https://github.com/novak-compress/compression-bench> under Apache 2.0. Compressed model checkpoints for BERT-Large are deposited on Hugging Face Hub (nina-novak/compression-bench). Results deposited in Zenodo under DOI: 10.5281/zenodo.19614793. Released under CC BY 4.0.

Ethical Approval

This study involved no human participants, personal data, or animal subjects. Ethics

committee approval was not required.

Appendix A

Compression Recipe Selection Framework

Select the recommended compression recipe based on deployment constraint priority. Recipes can be combined (e.g. quantisation plus 2:4 sparsity) for compound compression.