BENCHMARKING PARAMETER-EFFICIENT FINE-TUNING: TRADE-OFFS IN ACCURACY, SPEED, AND SPARSITY

Shaan Yadav, Christian Bepler, Ishan Vyas, Andrew McKnight, Tate Staples
ECE 685D - Fall 2025
Duke University

ABSTRACT

Fine-tuning large language models (LLMs) is computationally expensive due to their massive parameter counts. This paper investigates four parameter-efficient fine-tuning (PEFT) methods: LoRA (Low-Rank Adaptation), Sparse LoRA, QLoRA (Quantized LoRA), and High-Rank LoRA (as a proxy for HiRA). We evaluate these methods on three benchmark datasets (IMDB, SST-2, and WikiText-2) using DistilBERT and BERT-base architectures, measuring task accuracy, inference speed, memory footprint, and training efficiency. Because QLoRA is implemented on BERT-base while the other methods use DistilBERT, its gains in classification accuracy (93.28% on IMDB) reflect a combination of quantization and increased backbone capacity rather than a pure adapter-level improvement. We find that QLoRA offers substantial memory savings (about 48% model footprint reduction) but incurs a 2.5–3.5$\times$ training time penalty. High-Rank LoRA demonstrates superior language modeling performance (perplexity of 2.62 on WikiText-2), while Sparse LoRA achieves 50% adapter sparsity with minimal accuracy loss, offering a viable path for efficient deployment. Standard LoRA provides the most balanced trade-off across all metrics on a fixed DistilBERT backbone.

1 INTRODUCTION

Large language models (LLMs) deliver impressive accuracy on many NLP benchmarks, yet their billions of parameters make full fine-tuning prohibitive in terms of memory, wall-clock time, and energy. Updating every weight of a 65B-parameter model can require dozens of high-end GPUs, severely limiting accessibility. Parameter-efficient fine-tuning (PEFT) mitigates this barrier by freezing the model backbone and learning small adapter modules. LoRA and its variants exemplify this approach by injecting low-rank or structured updates that capture task-specific behavior without modifying the full model weights.

This paper benchmarks four adapter families—standard LoRA, Sparse LoRA, QLoRA, and High-Rank LoRA—on three representative tasks (IMDB, SST-2, WikiText-2). Because QLoRA is deployed on a larger backbone (BERT-base) than the other methods (DistilBERT), we treat its results as a separate "quantized large-backbone" configuration rather than a strictly apples-to-apples adapter comparison. We address three practical research questions: (i) How much accuracy is retained compared to full fine-tuning? (ii) What memory and compute savings does each variant offer? (iii) Which method is optimal under constraints such as training speed, deployment latency, or generative quality?

2 LITERATURE REVIEW

We briefly highlight the ingredients behind each adapter variant evaluated in this work.

LoRA freezes the base model and learns a rank-r update $\Delta W = BA$ injected into attention blocks (1). Because $r \ll d, k$, the adapter introduces only $r(d + k)$ parameters, can be merged back into W at inference, and leaves latency unchanged.

Sparse LoRA pushes efficiency further by pruning half of the LoRA weights through magnitude-based schedules and L1 regularization (2). The sparsity pattern is fixed post-training, enabling faster inference once a sparse kernel is available.

QLoRA combines LoRA with 4-bit NF4 quantization, double quantization of scaling factors, and paged optimizers to enable fine-tuning of massive models on a single GPU (3). Only the adapters remain in higher precision, yielding substantial memory savings at the cost of additional compute overhead.

HiRA relaxes the low-rank assumption by composing two rank-r matrices via Hadamard products, effectively emulating a higher rank without quadratic parameter growth (4). This extra expressiveness particularly helps generative tasks. In our project, we approximate this added expressiveness by replacing HiRA's Hadamard-composed factors with a High-Rank LoRA module, which increases the effective update rank through a structured sum of low-rank components while preserving parameter efficiency.

DistilBERT serves as our compact backbone with 66M parameters (6 transformer layers, 768 hidden dimensions, 12 attention heads) while retaining 97% of BERT's language understanding capability (5). It is distilled from **BERT-base-uncased**, which has 110M parameters (12 transformer layers, 768 hidden dimensions, 12 attention heads). QLoRA uses BERT-base because current 4-bit quantization kernels better support that architecture.

3 METHODOLOGY

We evaluate four PEFT strategies using the Hugging Face `peft` library and PyTorch. All experiments were conducted on the Duke Compute Cluster (DCC).

3.1 MODELS AND DATASETS

We employ **DistilBERT** (66M parameters) as the primary backbone for LoRA, Sparse LoRA, and High-Rank LoRA due to its efficiency. For **QLoRA**, we utilize **BERT-base-uncased** (110M parameters, 12 transformer layers, 768 hidden dimensions, 12 attention heads) because 4-bit quantization support for DistilBERT is limited in current libraries. BERT-base's larger capacity—nearly $1.7\times$ the parameters of DistilBERT—provides a richer representation space, which partly explains QLoRA's classification advantage. While this introduces a model size discrepancy, it reflects real-world usage where QLoRA is typically applied to larger architectures to maximize memory savings from quantization. Throughout the paper we therefore interpret QLoRA as a qualitatively different regime (quantized larger backbone) and avoid attributing its absolute accuracy gains solely to the adapter design.

We evaluate on three datasets:

- **IMDB**: Binary sentiment classification (25k train / 25k test).
- **SST-2**: Short-text binary sentiment classification from GLUE.
- **WikiText-2**: Causal language modeling to evaluate generative capabilities.

3.2 ADAPTER CONFIGURATIONS

- **LoRA**: Rank $r = 8$, alpha $\alpha = 32$, dropout 0.1. Targets query/value projections.
- **Sparse LoRA**: Identical to LoRA ($r = 8$) but incorporates magnitude pruning during training to induce 50% sparsity in the adapter weights. We employ a *cubic sparsity schedule* where the sparsity ratio s_t at step t follows $s_t = s_f \cdot \left(1 - \left(1 - \frac{t - t_0}{t_f - t_0}\right)^3\right)$ for $t \in [t_0, t_f]$, with $s_f = 0.5$ (final sparsity), t_0 (pruning start), and t_f (pruning end). This gradual ramp-up allows the network to adapt before aggressive pruning. At each pruning step, weights with the smallest magnitudes are zeroed out. We also apply L1 regularization ($\lambda = 10^{-5}$) to encourage weight sparsity throughout training.
- **QLoRA**: Uses 4-bit NormalFloat (NF4) quantization for the base model with double quantization, combined with LoRA adapters ($r = 8$).

- **High-Rank LoRA (HiRA Proxy)**: To test the hypothesis that higher capacity improves complex tasks, we implement a high-rank variant with $r = 32$ and $\alpha = 64$. This serves as a proxy for High-Rank Adaptation (HiRA) to evaluate the impact of increased trainable parameters on capacity-intensive tasks.

3.3 Evaluation Metrics

For classification tasks, we report **Accuracy** and **F1 Score**. For language modeling, we report **Loss** (cross-entropy) and **Perplexity**, computed as $\text{PPL} = \exp(\mathcal{L})$ where $\mathcal{L}$ is the average negative log-likelihood per token. Lower perplexity indicates better predictive performance. Across all tasks, we measure **Training Time** (minutes), **GPU Memory** (MB), **Trainable Parameter %** relative to the base model, and **Throughput** (samples/second) during inference.

4 Results

4.1 Classification Performance (IMDB & SST-2)

Table 1 summarizes the classification results on IMDB. **QLoRA** achieves the highest absolute accuracy (93.28%), but this improvement is confounded by its use of the larger BERT-base backbone rather than DistilBERT. In other words, the gain reflects both increased model capacity and the quantized-adapter recipe. This performance also comes at significant cost: training time increases by approximately $3\times$ compared to standard LoRA. Sparse LoRA maintains competitive accuracy (within 0.04 percentage points of LoRA) while achieving 50% sparsity, validating the redundancy hypothesis in adapter weights.

Table 1: IMDB Sentiment Classification Results. Note: QLoRA uses BERT-base (110M params); all others use DistilBERT (66M params).

Method	Accuracy	F1	Train %	Time	Memory
LoRA	92.52%	0.9252	1.09%	35.2 min	258 MB
Sparse LoRA	92.48%	0.9248	1.09%	35.2 min	258 MB
QLoRA[†]	**93.28%**	**0.9328**	**0.40%**	100.8 min	**134 MB**
High-Rank	92.93%	0.9293	1.75%	35.4 min	258 MB

[†]QLoRA uses BERT-base due to 4-bit quantization library constraints.

Results on SST-2 (Table 2) mirror these findings. QLoRA provides a +2.18 percentage point accuracy improvement but again benefits from the larger BERT-base backbone and requires significantly longer training. Standard LoRA and High-Rank LoRA offer balanced profiles, completing training in roughly 14 minutes, making them suitable for rapid iterative development.

Table 2: SST-2 Sentiment Classification Results. QLoRA uses BERT-base; others use DistilBERT.

Method	Accuracy	F1	Train %	Time	Memory
LoRA	90.02%	0.9002	1.09%	13.6 min	258 MB
Sparse LoRA	89.22%	0.8922	1.09%	13.6 min	258 MB
QLoRA[†]	**92.20%**	**0.9220**	**0.40%**	46.9 min	**134 MB**
High-Rank	90.14%	0.9014	1.75%	13.6 min	258 MB

[†]BERT-base backbone (see Section 3.1).

Figure 1 visualizes the accuracy comparison. The trade-off is clear: QLoRA delivers the highest absolute accuracy, but this is confounded by the larger base model capacity (BERT-base vs. DistilBERT), so we do not interpret this as a clean method-level win over LoRA. While QLoRA achieves the best raw performance, it is not Pareto-optimal if training speed is a primary constraint, and its advantage diminishes when normalized for base model size.

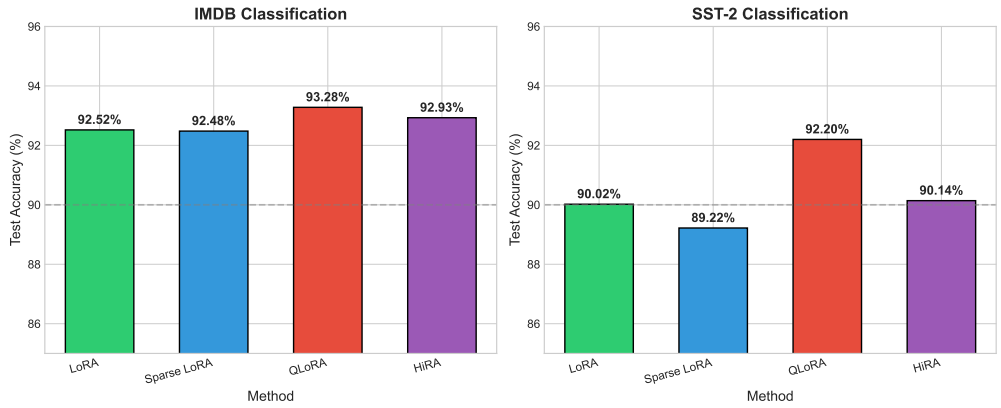


Figure 1: Classification accuracy comparison across IMDB and SST-2 datasets. QLoRA leads in accuracy, while LoRA and High-Rank variants remain competitive with significantly lower computational overhead.

4.2 Language Modeling

For language modeling on WikiText-2 (Table 3), the trend reverses. **High-Rank LoRA** achieves the lowest perplexity (2.62), suggesting that generative tasks benefit substantially from increased adapter capacity on a fixed DistilBERT backbone. QLoRA lags behind with a perplexity of 3.20, indicating that aggressive 4-bit quantization on a larger backbone may degrade performance in fine-grained next-token prediction tasks relative to dense DistilBERT adapters.

Table 3: WikiText-2 Language Modeling Results. QLoRA uses BERT-base; others use DistilBERT.

Method	Loss	Perplexity	Train %	Time	Throughput
LoRA	0.9867	2.68	0.22%	24.1 min	181.7 samples/s
Sparse LoRA	0.9938	2.70	0.22%	24.1 min	∼180 samples/s
QLoRA[†]	1.1631	3.20	0.40%	60.4 min	100.1 samples/s
High-Rank	**0.9642**	**2.62**	0.87%	24.2 min	169.9 samples/s

[†]BERT-base backbone (see Section 3.1).

4.3 Efficiency Metrics

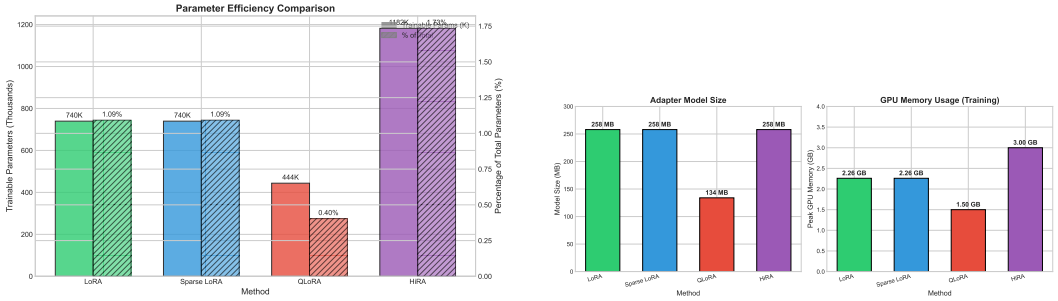
Figure 2 compares parameter counts and memory footprint. QLoRA shrinks adapter size by roughly half relative to the other methods, while Sparse LoRA matches LoRA's footprint but unlocks structured pruning. Training-time differences are captured in Figure 3; the hatched QLoRA bars highlight the 2.5–3.5× slowdown despite similar epochs and batch sizes.

Sparsity. Sparse LoRA reaches the target 50% sparsity on every dataset with negligible accuracy drift. The result hints at sizeable redundancy inside low-rank adapters and motivates sparse kernels for inference speedups.

5 Discussion

Task sensitivity. Under our setup, the quantized BERT-base configuration (QLoRA) improves classification accuracy (+0.76–2.18 percentage points on IMDB/SST-2) but yields 19% worse perplexity on WikiText-2 compared to dense DistilBERT adapters. Consequently, we recommend QLoRA primarily for memory-bound classification workloads where using a larger backbone is desirable; High-Rank LoRA or standard LoRA on DistilBERT are safer choices for language modeling.

Speed vs. accuracy. Figure 4 shows that QLoRA occupies the high-accuracy corner in absolute terms, even though LoRA and High-Rank LoRA are three times faster to fine-tune. Sparse LoRA tracks the LoRA point despite pruning half its parameters, making it attractive when rapid iteration matters.



(a) Parameter efficiency (b) Memory comparison

Figure 2: Efficiency metrics comparison. (a) QLoRA achieves the lowest trainable parameter percentage (0.40%), reflecting both its adapter design and the larger BERT-base backbone. (b) 4-bit quantization of the BERT-base backbone reduces the overall model memory by approximately 48% relative to the dense DistilBERT baseline.

Figure 3: Training time comparison across all datasets. QLoRA (hatched bars) takes 2.5-3.5× longer than other methods due to quantization overhead.

Capacity and sparsity. Two key insights emerge from our results. First, *adapter rank matters for generation*: within the DistilBERT family, High-Rank LoRA's $r = 32$ adapters achieve the best perplexity without increasing training time, indicating that language modeling benefits from higher-capacity updates even when classification does not. Standard LoRA's $r = 8$ appears sufficient for sentiment tasks but under-fits the more complex token prediction objective. Second, *low-rank adapters contain redundancy*: Sparse LoRA prunes 50% of adapter weights (Figure 5) with negligible accuracy loss (<0.8 percentage points), suggesting that sparse inference kernels could halve adapter compute at deployment.

Multi-metric view. The radar chart in Figure 6 summarizes the trade space: relative to the dense DistilBERT adapters, QLoRA (on BERT-base) dominates parameter and memory efficiency but at the cost of training speed; Sparse LoRA leads on speed among the DistilBERT-based methods, and the High-Rank variant excels at expressiveness for generation.

Recommendations.

- **Memory-constrained classification**: Pick QLoRA and accept the 3× longer training time in exchange for the ability to deploy a larger BERT-base backbone with approximately 50% lower memory footprint relative to a dense DistilBERT baseline.
- **Fast iteration or hyper-parameter sweeps**: Use standard LoRA or HiRA; both finish in 13–35 minutes while remaining within 0.8 percentage points of QLoRA.
- **Deployment/inference efficiency**: Sparse LoRA keeps accuracy intact, opens the door to sparse kernels, and matches LoRA's training cost. Figure 7 shows LoRA-style adapters also retain the best throughput.

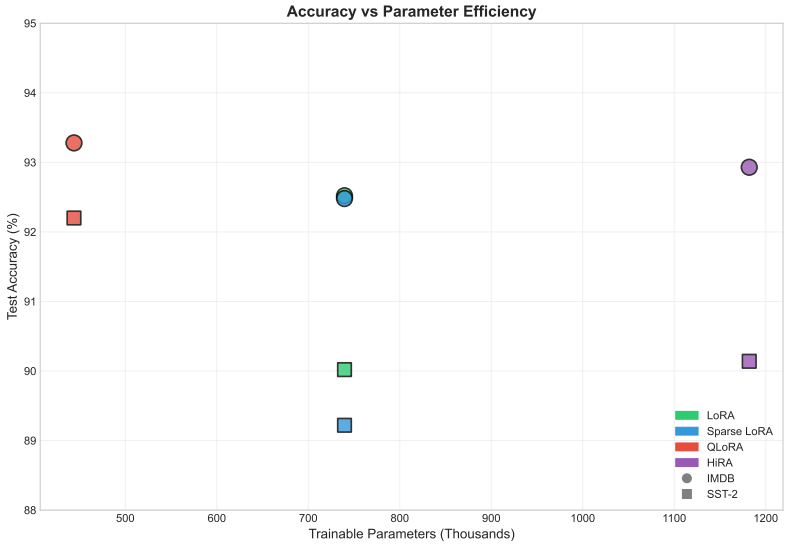


Figure 4: Accuracy vs. parameter efficiency (upper-left is better). Circles denote IMDB, squares denote SST-2.

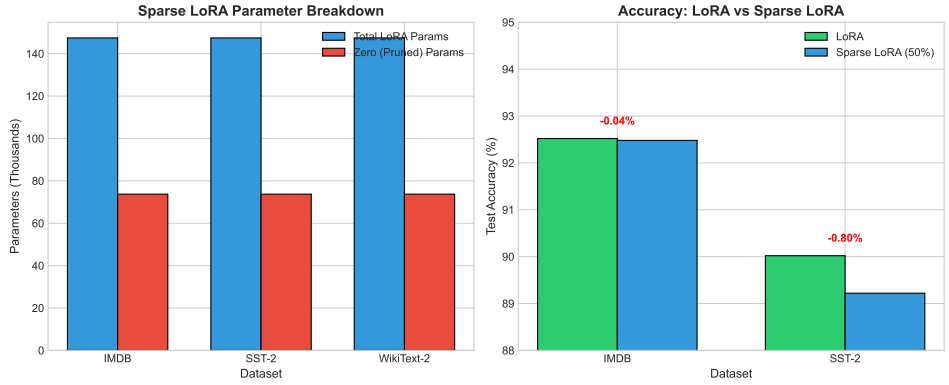


Figure 5: Sparse LoRA reaches 50% sparsity (left) while staying within 0.8 percentage points of dense LoRA on classification (right).

- **Text generation**: Prefer HiRA for superior perplexity; avoid QLoRA unless memory is the overriding constraint.

Limitations. QLoRA requires BERT-base, so parameter percentages and memory figures are not directly comparable to DistilBERT-based adapters; its absolute accuracy gains therefore conflate backbone capacity and quantization. We also report only single-run metrics without variance estimates, so stability across random seeds remains an open question. Our experiments cover sentiment analysis and language modeling only; other tasks (QA, summarization, dialogue) may yield different conclusions. Sparse LoRA speedups assume availability of sparse kernels, and our High-Rank LoRA implementation approximates rather than replicates the full HiRA Hadamard design.

6 Conclusion

This study benchmarks the trade-offs among accuracy, speed, and memory across four PEFT strategies. We find that no single method is universally optimal: in our setting, QLoRA enables a larger BERT-base backbone under a tighter memory budget but suffers from slow training and does not dominate DistilBERT-based adapters on generation; High-Rank LoRA is critical for generative quality on a fixed DistilBERT backbone; and Sparse LoRA offers a path to efficient inference by ex-

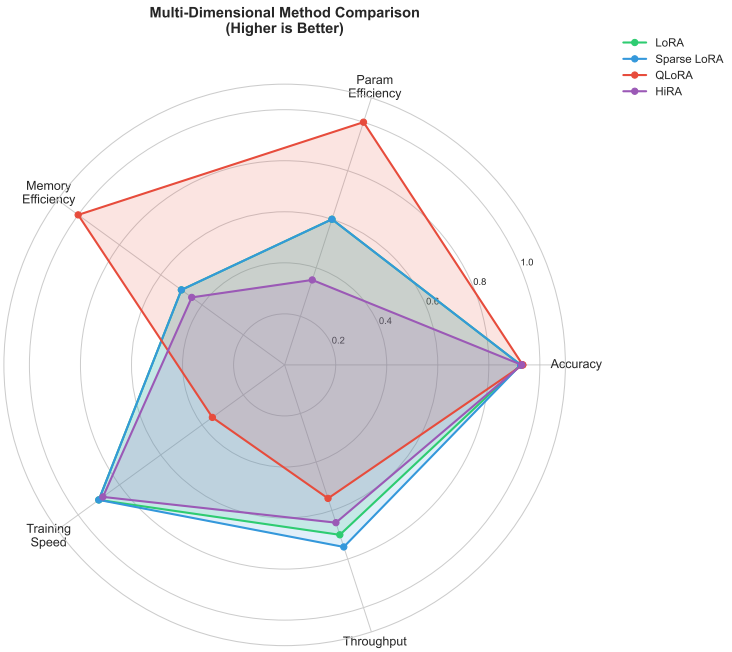


Figure 6: Multi-dimensional comparison across accuracy, parameter efficiency, memory, training speed, and throughput.

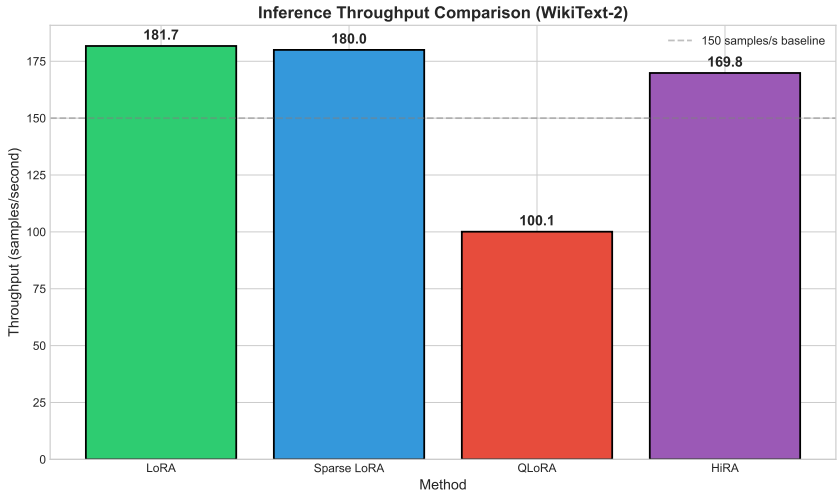


Figure 7: Inference throughput on WikiText-2. Sparse LoRA mirrors LoRA, while QLoRA drops to 100 samples/s because of dequantization overhead.

posing structured sparsity. For general-purpose applications on DistilBERT-sized models, standard LoRA remains the most balanced choice. Future work should explore hybrid approaches—such as combining quantization with sparsity—and investigate scaling laws of these methods on 7B+ parameter models, with strict backbone-normalized comparisons and multiple random seeds.

Acknowledgments: We thank the course instructors and TAs for their guidance. We also acknowledge the Hugging Face team for their open-source libraries.

REFERENCES

[1] Hu, Edward J., et al. *LoRA: Low-Rank Adaptation of Large Language Models.* In International Conference on Learning Representations (ICLR), 2022.

[2] Khaki, Samir, et al. *SparseLoRA: Accelerating LLM Fine-Tuning with Contextual Sparsity.* arXiv preprint arXiv:2506.16500, 2025.

[3] Dettmers, Tim, et al. *QLoRA: Efficient Finetuning of Quantized LLMs.* arXiv preprint arXiv:2305.14314, 2023.

[4] Huang, Qiushi, et al. *HiRA: Parameter-Efficient Hadamard High-Rank Adaptation for Large Language Models.* In The Thirteenth International Conference on Learning Representations (ICLR), 2025.

[5] Sanh, Victor, et al. *DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter.* arXiv preprint arXiv:1910.01108, 2019.