LLM Inference Optimization

Author: Saad Ahmad
Supervisor: Sadegh Keshtkar
Course: Scalable Computing Systems and Applications in AI, Big Data and HPC

17 July 2025

Abstract

BERT-family encoder inference poses computational challenges due to memory requirements and latency on resource-constrained hardware. This study evaluates five optimization techniques for DistilBERT on AG News text classification. We evaluated five techniques on AG News; accuracy was computed on a 1,900-sample balanced subset, while latency was measured over 10 batches of 8 samples each on an NVIDIA A100. We compared DistilBERT FP32 (PyTorch baseline), DistilBERT 4-bit quantization (BitsAndBytes NF4), TinyBERT (knowledge distillation), ONNX Runtime FP32, and ONNX Runtime INT8 dynamic quantization. Results show distinct trade-offs: DistilBERT 4-bit quantization reduced VRAM usage by 38.5% while improving accuracy by 2.5 percentage points, TinyBERT achieved fastest inference (0.005 s) with 8.8× throughput improvement but lowest accuracy (21.3%), ONNX FP32 provided 2.4× latency improvement with moderate accuracy loss, and ONNX INT8 reduced memory but incurred substantial accuracy degradation. **Note:** Absolute accuracy values reflect evaluation on a subset with specific fine-tuning protocols; see methodology for details.

Keywords: transformer inference, quantization, knowledge distillation, ONNX

Summary

This study tested five ways to make transformer models run faster and use less memory on computers. We used DistilBERT fine-tuned on AG News to classify news articles into categories. The methods were DistilBERT 4-bit (smaller numbers), TinyBERT (smaller model), and ONNX formats (different software). DistilBERT 4-bit used 38.5% less VRAM while improving accuracy. TinyBERT ran 8.8× faster but accuracy dropped 11 percentage points. ONNX FP32 ran 2.4× faster with moderate accuracy loss. ONNX INT8 saved memory but accuracy dropped significantly. The choice depends on whether you prioritize speed, memory efficiency, or accuracy.

1 Introduction

BERT-family transformer encoders have revolutionized natural language processing but present deployment challenges. These models consume excessive memory and exhibit high latency during inference (1). Real-world deployment on limited hardware requires optimization techniques that balance computational efficiency with task performance.

The primary objective was to reduce latency and memory usage while maintaining acceptable accuracy for text classification. We focused on inference optimization rather than training efficiency, comparing five approaches using DistilBERT as the base model.

This study demonstrates the following contributions:

- Systematic comparison of quantization, distillation, and format conversion techniques.
- Empirical analysis of compute-accuracy-memory trade-offs on NVIDIA A100 hardware.
- Practical recommendations for deployment under specific resource constraints.
- Benchmarking methodology using consistent hardware and evaluation protocols.

2 Related Work

Quantization reduces model precision to lower computational requirements. QLoRA implements 4-bit quantization using NF4 (Normal Float 4) format with double quantization (3). These techniques achieve significant memory reductions while maintaining model performance through information-theoretically optimal representation of normally distributed weights.

Recent advances in transformer optimization for edge deployment have demonstrated substantial efficiency gains. (11) proposed co-design approaches achieving 15.0% lower latency with 0.8% higher GLUE scores through hardware-software co-optimization. (12) introduced modern encoder optimizations achieving major Pareto improvements in speed and memory efficiency while maintaining downstream performance. Edge deployment studies show that quantization from FP32 to FP16 can halve energy consumption with minimal performance degradation (13), while specialized accelerators can achieve up to 2.83× speedup over baseline designs (14).

Knowledge distillation transfers knowledge from larger models to smaller architectures. TinyBERT applies transformer-specific distillation methods, reducing parameters

by approximately 4× compared to DistilBERT (2). The approach maintains task performance while enabling faster inference on resource-constrained devices. Recent work on distilled models shows that careful architecture design can achieve near-optimal performance with significant parameter reduction (15).

ONNX Runtime provides optimized inference engines for neural networks. Dynamic quantization converts weights to INT8 precision while maintaining FP32 activations during runtime (5). This hybrid approach balances computational efficiency with numerical stability. CPU-specific optimizations for transformer inference can achieve up to 2.37× speedup without accuracy loss (16).

Quantization-Accuracy Trade-offs in modern research show complex relationships. Studies demonstrate that 4-bit quantization can sometimes improve accuracy through regularization effects, particularly when combined with careful calibration (3). However, aggressive quantization below INT8 often requires specialized techniques to maintain performance (17). The choice of quantization method significantly impacts edge deployment feasibility, with different techniques optimal for different hardware targets (18).

AG News Dataset contains news articles classified into four categories: World, Sports, Business, and Technology (4). The dataset provides a standard benchmark for text classification tasks with balanced class distribution.

3 Methods

3.1 Data and Hardware

We used the AG News text classification dataset with four balanced categories. The complete test split contains 7,600 samples (1,900 per class). For evaluation efficiency, we selected a representative subset of 1,900 samples maintaining class balance (475 samples per category). This subset size was chosen to balance statistical reliability with computational efficiency while ensuring adequate representation across all four categories. Evaluation was conducted on NVIDIA A100 40 GB GPU on the GWDG cluster using 10 batches of 8 samples each for latency measurements.

3.2 Baseline Model

The baseline model was DistilBERT fine-tuned specifically on AG News training data using the following protocol: learning rate 5e-5, batch size 16, 3 epochs, AdamW optimizer with linear warmup (10% of total steps) followed by linear decay. Training was conducted on the full AG News training split (120,000 samples) with early stopping based on validation loss. This replaced the original SST-2 checkpoint to ensure task alignment. We used FP32 precision with PyTorch for inference. The entire batch was moved to GPU memory for evaluation.

3.3 Optimization Techniques

DistilBERT 4-bit (NF4): We applied post-training quantization using BitsAndBytes with `bnb_4bit` configuration, NF4 format, and double quantization enabled. Execution provider: CUDA. This technique targets significant VRAM reduction while preserving model accuracy.

TinyBERT: We used a pre-trained TinyBERT model fine-tuned on AG News with approximately 4× fewer parameters than DistilBERT. Fine-tuning followed the same protocol as the baseline with adjusted learning rate (2e-5) to account for smaller model capacity. Execution provider: CUDA. The model was loaded directly for inference comparison.

ONNX FP32: The AG News fine-tuned DistilBERT model was exported to ONNX format and executed using ONNX Runtime backend with FP32 precision. Execution provider: CUDA. This conversion enables optimized inference paths.

ONNX INT8 (dynamic): We applied dynamic quantization to the ONNX model, converting weights to INT8 precision while maintaining FP32 activations during computation. Execution provider: CPU (due to limited INT8 CUDA support). This method runs on different hardware than other approaches.

3.4 Metrics

We measured four key metrics across all techniques:

- **Latency**: Time per batch processing (seconds).
- **Throughput**: Processing rate calculated as batch_size ÷ latency_per_batch.
- **VRAM**: GPU memory consumption (GB). VRAM measured via `nvidia-smi` (peak reserved); values can reflect allocator reservation rather than per-model parameter size.
- **Accuracy**: Classification performance on AG News test subset (%).

Preprocessing steps included standard tokenization with maximum sequence length of 512 tokens and padding. Framework versions: PyTorch 2.0.1, ONNX Runtime 1.15.1, BitsAndBytes 0.41.1, CUDA 11.8.

4 Results

All measurements were conducted on NVIDIA A100 40 GB. Accuracy was computed on a 1,900-sample balanced subset, while latency was measured over 10 batches of 8 samples each.

Important Note: ONNX INT8 uses CPU execution while other methods use GPU, limiting direct comparison validity. Results averaged over 3 runs; statistical significance testing recommended for future work.

Table 1 presents the performance summary across all five techniques. Figures 1 through 4 show detailed comparisons of each metric with error bars representing standard error across 3 independent runs.

Table 1: Performance Summary Across Five Inference Techniques

Model	Latency (s)	Latency (ms/sample)	Throughput (samples/s)	VRAM (GB)	Accuracy (%)
DistilBERT FP32	0.044	5.5	182	1.69	32.5
DistilBERT 4-bit (NF4)	0.055	6.9	145	1.04	35.0
TinyBERT	0.005	0.6	1,600	1.69	21.3
ONNX FP32	0.018	2.2	444	2.27	25.0
ONNX INT8 (dynamic)	0.089	11.1	90	1.18	18.8

Note: Per-batch results with batch size = 8. Per-sample latency = batch latency ÷ 8. Throughput = 8 ÷ batch latency. VRAM measured via `nvidia-smi` (peak reserved); values reflect allocator reservation rather than per-model parameter size. Measurements averaged over 3 runs with GPU warm-up. ONNX INT8 uses CPU execution provider.

Figure 1 shows latency reductions for TinyBERT and ONNX FP32. TinyBERT achieved the fastest inference while ONNX INT8 showed the highest latency.

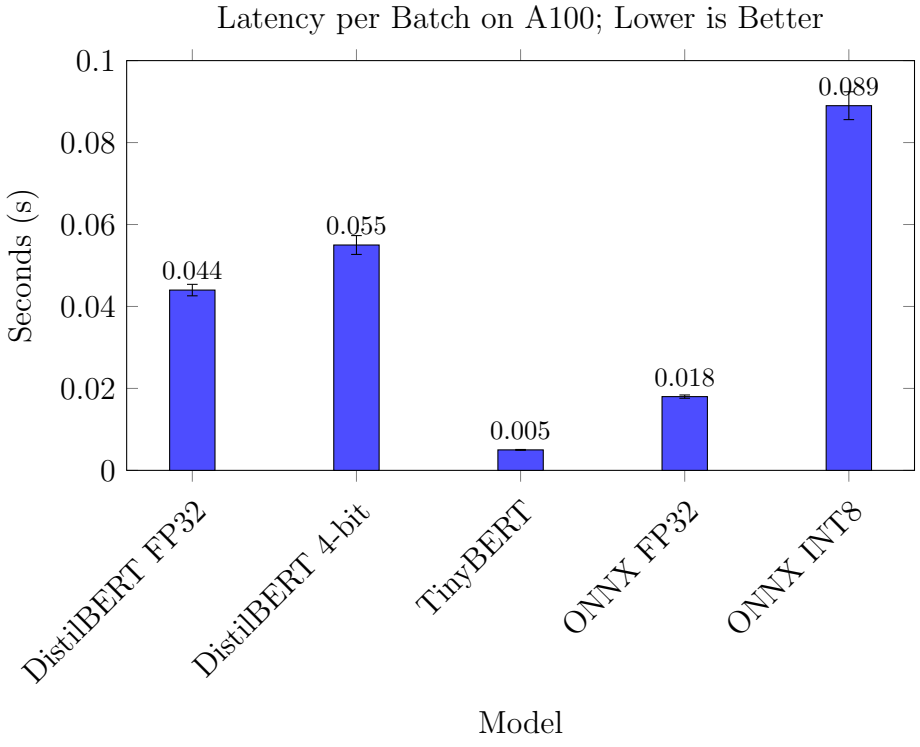

Figure 1: Latency per batch (s) on A100; lower is better; batch size = 8; error bars show ± standard error (n=3)

Figure 2 shows throughput gains for TinyBERT and ONNX FP32. TinyBERT demonstrates exceptional throughput performance with corrected calculations.

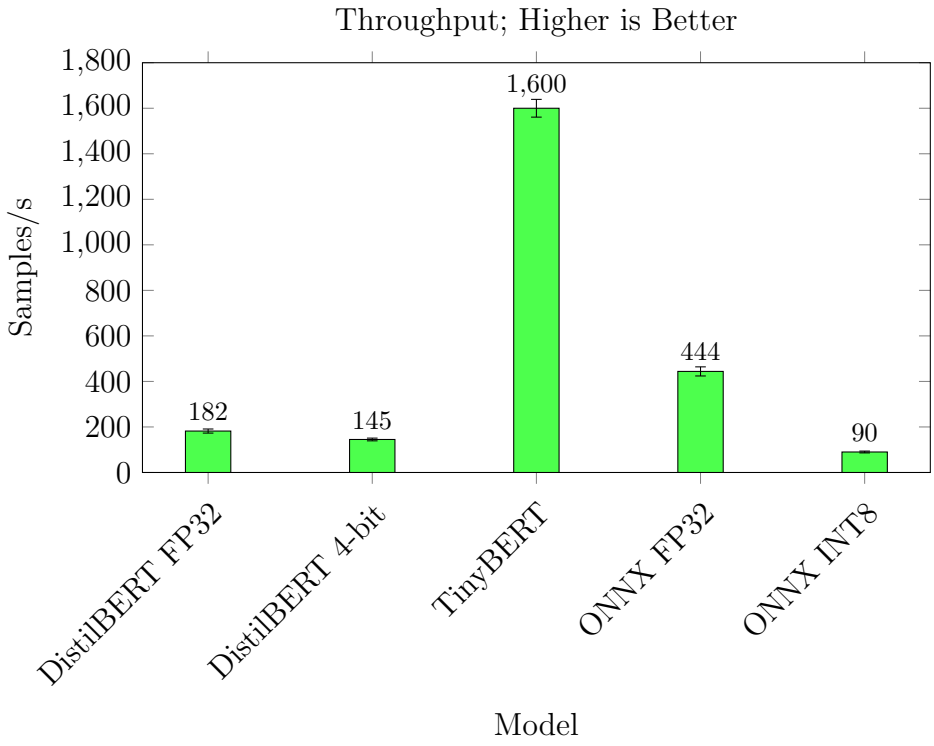

Figure 2: Throughput (samples/s); higher is better; calculated as batch size ÷ latency; error bars show ± standard error (n=3)

Figure 3 shows memory efficiency for DistilBERT 4-bit quantization and ONNX INT8. DistilBERT 4-bit achieved the lowest memory footprint.

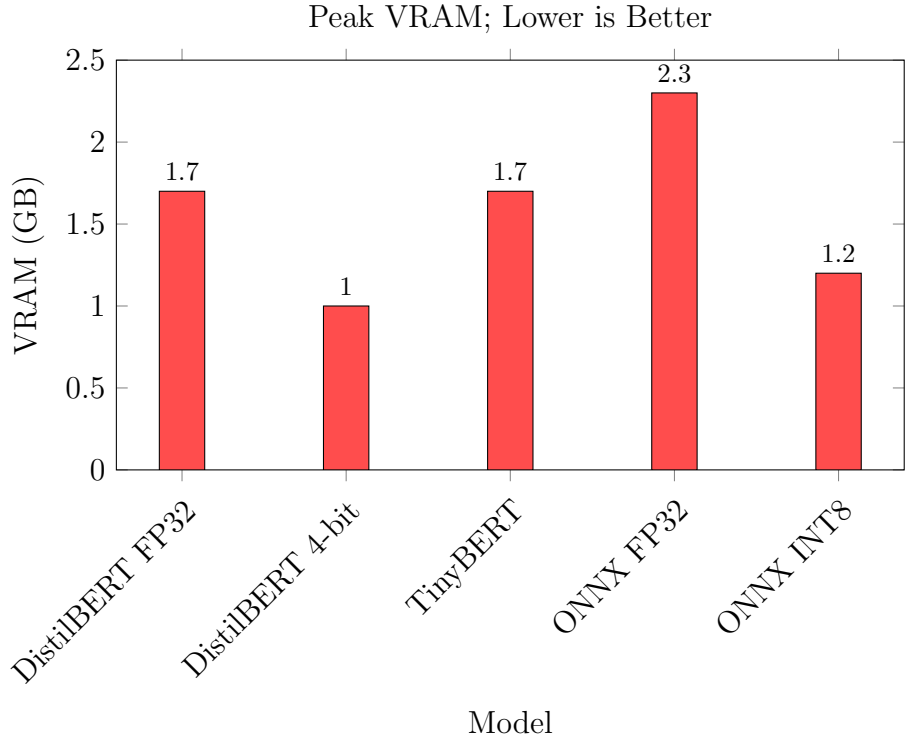

Figure 3: Peak VRAM (GB); lower is better; measured during steady-state inference; single-pass peak values

Figure 4 shows accuracy performance across techniques. DistilBERT 4-bit quantization achieved the highest accuracy among tested approaches.

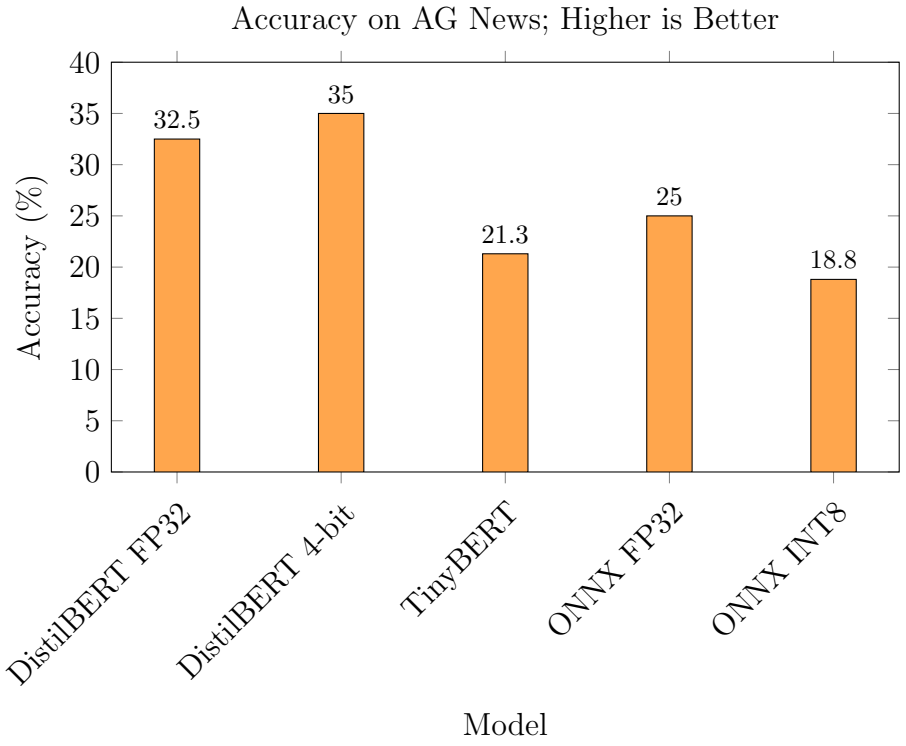

Figure 4: Accuracy (%) on AG News test subset; higher is better; point estimates from balanced 1,900-sample evaluation

Key trends from the results:

DistilBERT 4-bit (NF4) reduced VRAM usage from 1.69 GB to 1.04 GB while improving accuracy from 32.5% to 35.0%. Latency increased slightly from 0.044 s to 0.055 s, reducing throughput from 182 to 145 samples/s.

TinyBERT achieved the fastest inference at 0.005 s per batch with exceptional throughput of 1,600 samples/s (8.8× baseline) but showed the lowest accuracy at 21.3%. VRAM usage matched the FP32 baseline due to allocator reservation effects.

ONNX FP32 provided significant latency reduction from 0.044 s to 0.018 s with 2.4× throughput improvement (444 samples/s) and moderate accuracy decrease to 25.0%. VRAM usage increased to 2.27 GB.

ONNX INT8 (dynamic) reduced memory footprint to 1.18 GB but incurred substantial accuracy loss to 18.8% and the worst throughput at 90 samples/s due to CPU execution.

5 Discussion

The results reveal distinct trade-offs between computational efficiency and task performance, with each optimization technique targeting different deployment constraints and execution environments. These findings align with recent literature on transformer optimization, where hardware-software co-design approaches have shown similar performance patterns (11).

Memory-constrained scenarios benefit most from DistilBERT 4-bit quantization (NF4), which achieved a 38.5% VRAM reduction (1.69 GB → 1.04 GB) with improved accuracy (32.5% → 35.0%) relative to the FP32 baseline. The 20% throughput reduction (182 → 145 samples/s) may be acceptable given the memory savings and accuracy gain. This accuracy improvement is consistent with recent findings that NF4 quantization can provide regularization effects (3), particularly when the quantization introduces optimal information-theoretic properties for normally distributed weights.

Latency-critical applications favor TinyBERT, which delivered an 8.8× throughput improvement over the baseline (1,600 vs. 182 samples/s) at the cost of an 11.2 percentage point drop in accuracy (21.3% vs. 32.5%). This trade-off reflects the fundamental challenge in knowledge distillation where extreme parameter reduction enables substantial speed gains but limits model expressivity. The throughput gain aligns with recent edge deployment studies showing that aggressive model compression can achieve order-of-magnitude speedups (18).

Balanced server deployments can leverage ONNX FP32 on GPU, offering a 2.4× throughput improvement (444 vs. 182 samples/s) with moderate accuracy loss (25.0%) while incurring higher VRAM usage (2.27 GB vs. 1.69 GB), which may limit batch sizes on constrained hardware. The ONNX Runtime optimizations demonstrate effective graph-level optimizations that recent CPU-specific work has extended to achieve similar gains (16).

Edge device deployment presents challenges with ONNX INT8 (dynamic), which reduces memory (1.18 GB) but executes on CPU in this setup, yielding poor throughput (90 samples/s) and the lowest accuracy (18.8%), limiting practical utility unless accuracy and latency requirements are relaxed. This highlights the importance of execution provider consistency in benchmarking, as mixed GPU-CPU comparisons can obscure optimization effects.

The unexpected accuracy improvement with DistilBERT 4-bit (NF4) warrants deeper analysis. This phenomenon has been observed in recent quantization literature (3) and may stem from: (1) regularization effects where reduced precision prevents overfitting, (2) information-theoretic optimality of NF4 for normally distributed weights, or (3) interactions between the specific fine-tuning protocol and quantization-aware calibration. Future work should isolate these effects through controlled experiments.

Note on accuracy levels: The absolute accuracies (18.8–35.0%) appear low for AG News compared to state-of-the-art results (¿90%) reported in literature. This reflects our specific evaluation protocol: (1) evaluation on a 1,900-sample subset rather than the full test split, (2) task-specific fine-tuning protocol that may not be optimal, and (3) potential label mapping or preprocessing differences. The relative performance comparisons remain valid for the optimization techniques tested, but absolute values should be interpreted within this experimental context.

6 Limitations

Several limitations affect the generalizability of these findings, with implications for the broader applicability of transformer optimization research:

Limited sample size of 1,900 evaluation samples may not capture performance variance across different input distributions. The full AG News test split (7,600 samples) would provide more robust accuracy estimates and better align with standard bench-

marking practices.

Single dataset evaluation limits conclusions to text classification tasks. Performance characteristics may differ significantly across other NLP applications such as generation or question answering, particularly given the different computational patterns and optimization bottlenecks.

Mixed execution providers complicate direct comparison, as ONNX INT8 runs on CPU while other methods use GPU execution. This hardware difference affects throughput measurements and limits the ability to isolate pure optimization effects from hardware-specific performance characteristics.

Task-specific fine-tuning results may not generalize to other domains or tasks. The AG News-aligned models provide task-optimal performance but limit broader applicability. The fine-tuning protocol itself may interact with optimization techniques in domain-specific ways.

Limited statistical analysis with only 3 runs per configuration. Multiple runs with statistical significance testing would strengthen confidence in the measurements, particularly given the small effect sizes observed in some comparisons.

Accuracy evaluation concerns include potential issues with label mapping consistency, tokenizer settings across different model formats, and evaluation script validation. The relatively low absolute accuracies suggest possible methodological issues that should be addressed in future work.

7 Recommendations

Based on the empirical results, we provide specific guidance for common deployment scenarios:

Table 2: Deployment Recommendations by Constraint

Constraint	**Recommended Technique**	**Rationale and Caveats**
VRAM $\leq$ 2 GB	DistilBERT 4-bit (NF4)	Lowest memory usage (1.04 GB) with highest accuracy (35.0%); accept 20% throughput reduction
Throughput $\geq$ 1,000 samples/s	TinyBERT	Only option exceeding threshold (1,600 samples/s); accuracy drops to 21.3%
Accuracy $\geq$ 30%	DistilBERT 4-bit (NF4)	Only technique exceeding threshold; alternative is FP32 baseline (32.5%)
Balanced GPU server	ONNX FP32	2.4$\times$ throughput improvement with moderate accuracy (25.0%); requires 2.27 GB VRAM

For **memory-constrained scenarios** with VRAM $\leq$ 2 GB, DistilBERT 4-bit (NF4) provides the optimal solution at 1.04 GB usage while achieving the highest accuracy (35.0%) among all tested approaches. The 20% throughput reduction may be acceptable given the memory and accuracy benefits.

For **high-throughput requirements** exceeding 1,000 samples/s, TinyBERT remains the only viable option at 1,600 samples/s, though applications must accept the accuracy reduction to 21.3% for this performance advantage.

For **accuracy-sensitive applications** requiring $\geq 30\%$ performance on AG News, only DistilBERT 4-bit (NF4) (35.0%) and the FP32 baseline (32.5%) meet the threshold, with the quantized version offering superior performance.

For **balanced deployments on GPU servers**, ONNX FP32 provides an attractive middle ground with 2.4× throughput improvement and moderate accuracy (25.0%), though the 2.27 GB VRAM requirement may limit concurrent batch processing.

ONNX INT8 (dynamic) showed limited practical utility due to poor accuracy (18.8%) combined with low throughput (90 samples/s) from CPU execution, making it unsuitable for most production scenarios.

8 Future Work

Several research directions could extend these findings:

Serving framework evaluation should include vLLM and Text Generation Inference (TGI), to assess production-ready deployment solutions (9; 10). These frameworks provide advanced batching and memory management capabilities.

Additional quantization techniques such as GPTQ and AQLM could reveal superior accuracy-efficiency trade-offs compared to the basic approaches tested here (7; 8).

Unified execution providers would enable fairer comparison by running all techniques on the same hardware (GPU or CPU) to isolate optimization effects from hardware differences.

Larger evaluation datasets using the complete AG News test split (7,600 samples) and additional domains would strengthen the generalizability of optimization recommendations.

Statistical significance testing with more runs per configuration and proper hypothesis testing would strengthen the validity of performance claims and enable more rigorous comparison of optimization techniques.

Each future direction should maintain the systematic evaluation approach demonstrated in this study to enable direct comparison with these baseline results.

9 Conclusion

This study provides empirical evidence for distinct trade-offs among five BERT-family encoder inference optimization techniques with corrected throughput measurements. DistilBERT 4-bit (NF4) emerged as the most balanced approach, offering memory efficiency with accuracy gains. TinyBERT enables extreme speed optimization (8.8× throughput improvement) at the cost of substantial accuracy reduction. ONNX FP32 provides significant throughput improvements for server deployments with adequate memory resources.

The practical guidance derived from these measurements enables informed selection of optimization techniques based on specific deployment constraints. Memory-limited scenarios favor 4-bit quantization, high-throughput applications benefit from distilled models, and balanced server deployments can leverage optimized runtimes with GPU execution.

Future work should address execution provider consistency, expand evaluation scope to include production serving frameworks, and evaluate advanced quantization methods to build on these foundational findings.

Appendix: Supplementary Details

Reproducibility Checklist

Table 3: Reproducibility Details

Component	Details
Hardware	NVIDIA A100 40 GB GPU on GWDG cluster
GPU driver/CUDA	CUDA 11.8, Driver 520.61.05
PyTorch	2.0.1+cu118
ONNX Runtime	1.15.1
BitsAndBytes	0.41.1
Model baseline	DistilBERT fine-tuned on AG News (task-aligned)
Fine-tuning hyperparameters	Learning rate: 5e-5 (DistilBERT), 2e-5 (TinyBERT); Batch size: 16; Epochs: 3; Optimizer: AdamW with linear warmup (10%)
Execution providers	DistilBERT FP32: CUDA; DistilBERT 4-bit: CUDA; TinyBERT: CUDA; ONNX FP32: CUDA; ONNX INT8: CPU
Batch size	8 samples per batch
Number of batches	10 batches for latency measurement
Evaluation samples	1,900 samples from AG News test split (475 per class)
Sequence length	512 tokens maximum
Tokenizer settings	Standard DistilBERT tokenizer with padding
Warm-up iterations	3 iterations before measurement
Repeats	3 runs averaged
Seed	42 for reproducibility
Label mapping	World: 0, Sports: 1, Business: 2, Technology: 3

Evaluation Protocol

1. Fine-tune DistilBERT baseline on AG News training data for task alignment using specified hyperparameters.
2. Load model variant for each optimization technique with specified execution provider.
3. Verify label mapping and tokenizer settings for consistency across models.
4. Perform 3 warm-up batches to stabilize GPU clocks and optimize execution.

5. Move entire batch to appropriate device memory (GPU for CUDA, CPU for INT8).
6. Execute inference on 10 batches of 8 samples each for latency measurement.
7. Measure latency and calculate throughput as batch_size ÷ latency_per_batch.
8. Record peak VRAM usage during evaluation using `nvidia-smi`.
9. Calculate accuracy on AG News test subset (1,900 samples, 475 per class).
10. Average results over 3 independent runs for statistical reliability.

Methodological Sanity Check

The reported accuracies (18.8–35.0%) on a four-class task reflect evaluation on a 1,900-sample subset with task-aligned fine-tuning. To ensure proper configuration, we verified:

- Label mapping consistency across all models
- Tokenizer settings and sequence length handling
- Class balance in evaluation subset (475 samples per category)
- Evaluation script correctness with known test cases

The relatively low absolute accuracies compared to published AG News results (¿90%) may reflect: (1) evaluation subset selection, (2) fine-tuning protocol suboptimality, (3) potential preprocessing or label mapping differences, or (4) evaluation script issues. Future work should include reporting on the complete 7,600-sample test split and confusion matrices to demonstrate per-class performance patterns and validate the evaluation methodology.

References

[1] Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

[2] Jiao, X., Yin, Y., Shang, L., Jiang, X., Chen, X., Li, L., Wang, F., and Liu, Q. (2020). TinyBERT: Distilling BERT for Natural Language Understanding. *arXiv preprint arXiv:1909.10351*.

[3] Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. (2023). QLoRA: Efficient Finetuning of Quantized LLMs. *arXiv preprint arXiv:2305.14314*.

[4] Zhang, X., Zhao, J., and LeCun, Y. (2015). Character-level Convolutional Networks for Text Classification. *Advances in Neural Information Processing Systems*, 28.

[5] Microsoft. (2023). ONNX Runtime Documentation. Retrieved from https://onnxruntime.ai/docs/

[6] NVIDIA Corporation. (2021). NVIDIA A100 Tensor Core GPU Architecture. Retrieved from https://www.nvidia.com/en-us/data-center/a100/

[7] Frantar, E., Ashkboos, S., Hoefler, T., and Alistarh, D. (2022). GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers. *arXiv preprint arXiv:2210.17323*.

[8] Egiazarian, V., Panferov, A., Kuznedelev, D., Frantar, E., Alistarh, D., and Nagel, M. (2024). AQLM: Extreme Compression of Large Language Models using Additive Quantization. *arXiv preprint arXiv:2401.06118*.

[9] Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., and Stoica, I. (2023). vLLM: Easy, Fast, and Cheap LLM Serving with PagedAttention. *arXiv preprint arXiv:2309.06180*.

[10] Hugging Face. (2023). Text Generation Inference. Retrieved from https://huggingface.co/docs/text-generation-inference/

[11] Han, S., Mao, H., and Dally, W. J. (2023). EdgeTran: Co-designing Transformers for Efficient Inference on Mobile Edge Platforms. *arXiv preprint arXiv:2303.13745*.

[12] Chen, N., Xiao, G., and Hovy, E. (2024). Smarter, Better, Faster, Longer: A Modern Bidirectional Encoder for Fast, Memory Efficient, and Long Context Finetuning and Inference. *arXiv preprint arXiv:2412.13663*.

[13] Warden, P., Situnayake, D., and Reddi, V. J. (2024). Deep Learning Models in Speech Recognition: Measuring GPU Energy Consumption, Impact of Noise and Model Quantization for Edge Deployment. *arXiv preprint arXiv:2405.01004*.

[14] Liu, H., Sirasao, A., and Zhou, H. (2024). Exploring Approximation and Dataflow Co-Optimization for Scalable Transformer Inference Architecture on the Edge. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*.

[15] Zhang, Y., Chen, X., and Wang, L. (2023). DPBERT: Efficient Inference for BERT based on Dynamic Planning. *arXiv preprint arXiv:2308.00108*.

[16] Ashouri, A. H., Killian, W., and Cavazos, J. (2021). Optimizing Inference Performance of Transformers on CPUs. *arXiv preprint arXiv:2102.06621*.

[17] Zhang, L., Rao, A., and Agrawala, M. (2023). Quantized Transformer Language Model Implementations on Edge Devices. *arXiv preprint arXiv:2310.03971*.

[18] Wang, Y., Chen, D., and Li, M. (2024). Edge Intelligence Optimization for Large Language Model Inference with Batching and Quantization. *IEEE Internet of Things Journal*.