

Energy–Accuracy Trade-offs in Transformer-Based NLP Models: A Unified Benchmarking Study

Stanford CS224N Custom Project

Thibaud Clement

Department of Computer Science
Stanford University
txc@stanford.edu

Abstract

Modern natural language processing (NLP) systems rely on transformer-based architectures that achieve strong predictive performance but require substantial computational resources. As models continue to scale, understanding the trade-offs between accuracy and energy consumption has become increasingly important. However, prior work often evaluates efficiency techniques in isolation, making it difficult to systematically evaluate architectural compression methods and inference-time optimizations under consistent experimental conditions. In this project, we investigate energy–accuracy trade-offs in transformer-based NLP models using a controlled benchmarking framework that measures GPU power consumption during both training and inference. Using the SQuAD v2 question answering dataset, we evaluate several BERT-derived architectural variants alongside inference-time optimizations within a unified experimental setting. Precision reduction emerges as the most effective efficiency strategy, reducing inference energy by up to 75% while preserving predictive performance. More broadly, different techniques affect different parts of the computational pipeline: architectural modifications such as pruning and layer freezing primarily reduce training energy, whereas inference-time techniques shift the deployment-time energy–accuracy frontier. Sequence length truncation produces a smooth Pareto frontier between energy consumption and performance, allowing moderate energy reductions with limited accuracy degradation. In practice, the most energy-efficient strategy depends on the expected workload: architectures with lower training energy are preferable when inference volume is small, whereas configurations with lower per-example inference energy dominate at large deployment scales.

1 Key information to include

- Mentor: David Anugraha.
- External Collaborators (if you have any): No.
- Sharing project: No.
- Late Days (if you are using any): None.

2 Introduction

Recent progress in natural language processing (NLP) has been driven by transformer-based architectures that have dramatically improved performance across a wide range of language understanding tasks. Since the introduction of the transformer architecture [1] and pretrained language models such as BERT [2], increasingly large models have achieved state-of-the-art results in applications including question answering, natural language inference, and text generation. Empirical studies show that performance often improves with increased model size and training compute, reinforcing a trend toward ever larger models and training workloads [3, 4, 5]. However, these gains have come at the cost of rapidly growing computational requirements.

This expanding computational footprint has raised concerns about the environmental and economic sustainability of modern machine learning systems. Prior studies have shown that training and deploying large neural language models can require substantial energy resources—sometimes reaching hundreds of megawatt-hours for a single training run—and generate significant carbon emissions [6, 7]. These concerns have motivated growing interest in energy-efficient machine learning and the broader “Green AI” paradigm [8], which advocates treating efficiency as a primary evaluation criterion alongside model accuracy.

However, improving the efficiency of transformer models remains challenging. Because computational cost scales with both model size and input sequence length [1, 3], and many components of the network contribute to predictive performance [9], reducing computational cost without compromising accuracy is not straightforward. A wide range of efficiency techniques have therefore been proposed, including architectural compression methods such as knowledge distillation [10, 11], sparsity and pruning approaches that identify efficient subnetworks [12, 13], and hardware-aware inference optimizations such as reduced-precision computation [14, 15]. In practice, these methods are typically evaluated in isolation, making it difficult to systematically characterize the resulting energy–accuracy trade-offs under consistent experimental conditions.

In this project, we investigate the energy–accuracy trade-offs of transformer-based NLP models across architectural and inference-time efficiency strategies. We develop a controlled benchmarking framework that measures GPU power consumption during both training and inference (Section 4), enabling direct comparison between multiple efficiency techniques within a unified experimental setting (Section 5). Our analysis reveals that different efficiency strategies affect the computational pipeline in distinct ways—architectural compression primarily reduces training energy, while inference-time techniques shift the energy–accuracy frontier during deployment—and that the most energy-efficient model configuration depends strongly on the expected deployment regime (Section 6). These findings provide practical insight into how efficiency techniques should be selected when deploying NLP models under different computational and energy constraints.

3 Related work

A growing body of work has examined the energy and environmental costs associated with modern machine learning systems [16]. Early studies showed that training large neural language models can require substantial energy resources and produce significant carbon emissions [6]. Subsequent work advocated reporting computational cost alongside model accuracy and highlighted the importance of developing more energy-efficient machine learning systems [7]. These efforts have helped motivate the broader “Green AI” paradigm [8], which emphasizes evaluating models not only in terms of predictive performance but also in terms of computational efficiency.

Architectural compression techniques aim to reduce the size or complexity of pretrained language models while preserving predictive performance. Knowledge distillation methods train smaller models that approximate the behavior of larger pretrained networks [10, 11], while other approaches reduce model complexity through parameter sharing or factorized embeddings [17]. Sparsity and pruning methods attempt to identify efficient subnetworks within large models, suggesting that large neural networks contain smaller subnetworks capable of achieving comparable performance when trained in isolation [12]. Building on this idea, pruning strategies such as movement pruning [13] have been applied to transformer architectures, while related approaches modify transformer depth during training [18], allowing subnetworks of different depths to be evaluated at inference time with limited performance degradation.

Complementary work explores inference-time techniques that reduce computational cost during deployment. Mixed-precision computation and quantization methods [14, 15] can accelerate neural network inference while reducing memory usage. Reducing the maximum input sequence length is another simple strategy for lowering computational cost, since the complexity of transformer self-attention grows quadratically with sequence length [1]. Other approaches rely on dynamic inference, where the model adapts its computation to the input, for example by introducing confidence-based early exiting [19], or by progressively pruning tokens during inference [20].

Although these techniques improve efficiency, they are typically studied within separate methodological frameworks that focus on a single optimization strategy, such as distillation [10], pruning [13], or dynamic inference [19]. As a result, there is limited empirical understanding of how architectural compression strategies and inference-time optimization techniques interact when evaluated under consistent experimental conditions. Our work addresses this gap by introducing a unified energy-aware benchmarking framework that evaluates multiple efficiency strategies within a controlled experimental setting and characterizes the resulting energy–accuracy trade-offs for transformer-based NLP models.

4 Approach

4.1 Model architectures and baselines

Our study evaluates several architectural variants of transformer-based language models derived from the BERT family. BERT remains a widely used and well-understood transformer architecture, making it a suitable reference model for controlled comparisons. In addition, the moderate size of BERT-base—a pretrained transformer model with 12 encoder layers and approximately 110 million parameters [2]—enables multiple architectural variants to be trained and evaluated within the computational constraints of this project. Accordingly, we adopt BERT-base as our primary baseline. Public benchmarks¹ indicate that it achieves roughly 75–77 F1 on SQuAD v2 depending on configuration. We also evaluate DistilBERT [10], a distilled variant that reduces model size and computational cost while retaining roughly 97% of BERT’s performance on downstream tasks.

To enable a controlled comparison with this distilled baseline, we construct four BERT-derived variants that represent distinct efficiency strategies explored in prior work. Following parameter-freezing strategies commonly used in transfer learning [21], BERT-Freeze6 freezes the lower six encoder layers during fine-tuning, reducing training cost while preserving the upper layers’ task adaptation. BERT-Edge updates only the first and final encoder layers, representing a more aggressive restriction of trainable parameters motivated by observations that many transformer layers exhibit functional redundancy [18]. BERT-Prune50 applies magnitude pruning to remove approximately 50% of model parameters, followed by a brief recovery fine-tuning stage, reflecting sparsity-based compression approaches inspired by the Lottery Ticket Hypothesis [12]. Finally, BERT-EE modifies BERT to support early-exit inference, enabling the model to terminate computation before the final layer once sufficient confidence is reached, following the dynamic inference paradigm explored in prior work such as DeeBERT [19].

All architectural variants share the same pretrained initialization and fine-tuning pipeline so that performance differences arise from the efficiency strategy rather than training differences. BERT, DistilBERT, BERT-Freeze6, BERT-Edge, and BERT-Prune50 are used throughout the main architectural and inference-time experiments. BERT-EE is evaluated separately for early-exit analysis.

4.2 Inference-time efficiency techniques

Beyond architectural variants, we evaluate several techniques identified in preliminary experiments (Appendix A) to reduce computational cost during model deployment. First, we study sequence length truncation, which limits the maximum number of tokens processed by the model. Because the computational complexity of transformer self-attention scales quadratically with sequence length [1], reducing the number of input tokens can substantially decrease inference cost. Second, we evaluate reduced numerical precision. This technique performs inference using lower-precision numerical formats, which can reduce arithmetic and memory costs while often preserving predictive performance. Third, we explore token pruning, which dynamically reduces the number of tokens processed by later transformer layers. We score context tokens by similarity to the question embedding, retain the top-ranked tokens under a fixed keep ratio, and preserve neighboring tokens within a small local window to maintain span structure. Together, these architectural and inference-time strategies define the experimental axes used to evaluate energy–accuracy trade-offs.

4.3 Measurement framework

We evaluate energy–accuracy trade-offs by measuring the computational cost of training and inference across model configurations. All experiments are executed within a unified benchmarking harness implemented by us², ensuring that model architectures, inference reductions, and hardware conditions are evaluated consistently. Energy consumption is measured by sampling instantaneous GPU power during execution and integrating power over time. With $P(t)$ denoting instantaneous GPU power in watts at time t and T the total runtime of a given operation, the total energy consumption E , expressed in joules, is computed as $E = \int_0^T P(t)dt$. Power measurements are obtained through periodic GPU telemetry queries (via `nvidia-smi`). This allows us to estimate the energy required for each training or inference workload. To facilitate comparison across configurations, we report normalized metrics including energy per example (J/example).

¹<https://rajpurkar.github.io/SQuAD-explorer/>

²<https://github.com/thibaudclement/nlpfrontiers-research/>

5 Experiments

5.1 Data

We conduct our experiments on the Stanford Question Answering Dataset v2 (SQuAD v2) [22]. SQuAD v2 is a widely used benchmark for extractive question answering that extends the original SQuAD [23] dataset by introducing unanswerable questions. Each example consists of a context paragraph and a question. The task is to predict either a contiguous answer span within the context or that no answer is present. The dataset contains approximately 130,000 training examples and 12,000 development examples. Following standard practice, we fine-tune pretrained transformer models on the training split and evaluate on the development split. Inputs are tokenized using the corresponding model tokenizer and truncated to a maximum sequence length when necessary. For each example, the model predicts start and end token positions within the context, along with a score for the null (no-answer) prediction. The varying context lengths and unanswerable questions make SQuAD v2 suitable for studying efficiency–accuracy trade-offs.

5.2 Evaluation method

We evaluate predictive performance using the standard SQuAD v2 metrics: Exact Match (EM) and token-level F1 [22]. Exact Match measures the percentage of predictions that exactly match the ground-truth answer span, while F1 measures token overlap between the predicted and gold answers. We report both metrics in the architectural baseline comparison, and use F1 as the primary accuracy measure in energy–accuracy plots. To evaluate computational efficiency, we report inference energy per example and total training energy. Energy consumption is measured using the methodology described in Section 4. Because SQuAD v2 includes unanswerable questions, models produce a score for the null (no-answer) prediction in addition to span predictions. Following standard evaluation practice, a threshold on this score determines whether a prediction is treated as unanswerable. We determine this threshold on the development set and then keep it fixed across all experiments to ensure consistent evaluation of EM and F1.

5.3 Experimental details

All experiments are conducted on a single NVIDIA L4 GPU using PyTorch³ with CUDA support and Hugging Face Transformers⁴. Unless otherwise stated, full-run training uses random seed 42, maximum sequence length 384, document stride 128, learning rate 3×10^{-5} , weight decay 0.01, gradient accumulation 1, per-device training batch size 8, per-device evaluation batch size 16, and two training epochs. We use a linear learning-rate schedule with 500 warm-up steps, log every 100 steps, and run intermediate evaluation every 1000 steps. Checkpoints are saved every 10000 steps, with at most one checkpoint retained.

For architectural baselines, BERT serves as the reference model. BERT-Freeze6 uses the same configuration while freezing the lowest six encoder layers, and BERT-Edge trains only layers 0, 1, 10, and 11. BERT-Prune50 applies global magnitude pruning with target sparsity 0.50 over linear layers, including the QA head, followed by a one-epoch recovery fine-tune using the same optimizer and batch-size settings. BERT-EE is trained separately to support early-exit inference.

Inference-time sweeps are evaluated at maximum sequence lengths {384, 320, 288, 256, 224, 192}, precision modes {FP32, FP16, BF16}, and token-pruning keep ratios {1.00, 0.80, 0.60, 0.40, 0.30}. Lower-precision formats such as FP8 were not supported on the available hardware platform and are therefore left for future work. Evaluation uses 10 warm-up batches, `n_best_size` = 20, maximum answer length 30, and a fixed no-answer probability threshold of 0.06458. Instantaneous GPU power is sampled via `nvidia-smi` every 0.2 seconds and integrated over time to compute energy consumption.

5.4 Results

5.4.1 Architecture comparison

Table 1 summarizes the performance and energy characteristics of the architectural baselines. As expected, the fully fine-tuned BERT model achieves the highest predictive performance, with an

³<https://github.com/pytorch/pytorch>

⁴<https://huggingface.co/docs/transformers/en/index>

F1 score of 75.97 and Exact Match of 73.18, and serves as the accuracy reference point for the remaining variants. DistilBERT provides the most efficient inference configuration, reducing energy consumption per example by approximately 50% and achieving the lowest latency, but at a noticeable cost in predictive performance.

The BERT-derived variants illustrate different trade-offs between accuracy and training cost. BERT-Freeze6 maintains performance close to the full BERT model while modestly reducing training energy. BERT-Prune50 achieves nearly identical predictive performance while substantially reducing training energy through 50% magnitude pruning followed by a recovery fine-tune. By contrast, BERT-Edge lowers training cost but incurs a larger drop in predictive performance, suggesting that aggressively restricting the set of trainable layers reduces adaptation capacity too severely for this task.

Importantly, the inference energy and latency of BERT, BERT-Freeze6, and BERT-Prune50 remain nearly identical. BERT-Edge also remains much closer to these models than to DistilBERT in inference cost, despite its lower accuracy. This indicates that architectural changes that reduce training cost do not necessarily improve inference-time efficiency.

Model	EM	F1	Energy / example (J)	Training Energy (kJ)	Latency / example (ms)
BERT	73.18	75.97	0.419	626.22	5.87
DistilBERT	66.54	68.63	0.209	320.70	2.95
BERT-Freeze6	72.50	75.50	0.420	550.83	5.87
BERT-Edge	66.53	69.35	0.419	522.88	5.86
BERT-Prune50	72.70	75.93	0.420	313.59	5.88

Table 1: Architectural baseline results on SQuAD v2. Best values in each column are shown in bold.

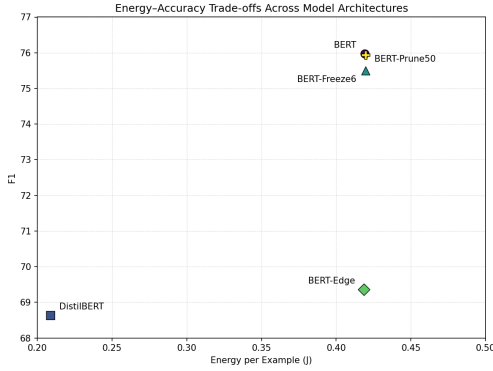


Figure 1: Inference energy per example vs. F1 across model architectures.

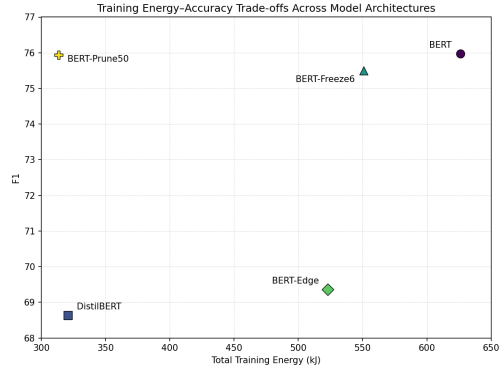


Figure 2: Training energy vs. F1 across model architectures.

Figures 1 and 2 visualize these trade-offs, plotting F1 against inference energy per example and F1 against total training energy, respectively. In Figure 1, DistilBERT occupies the low-energy region, reflecting its substantially reduced inference cost, while BERT lies at the high-accuracy end of the spectrum. BERT-Freeze6 and BERT-Prune50 remain close to BERT in inference cost while preserving much of its accuracy. BERT-Edge moves downward in accuracy without achieving a comparable reduction in inference energy. Figure 2 shows a different pattern. DistilBERT and BERT-Prune50 substantially reduce training energy relative to BERT, but only BERT-Prune50 remains near the baseline in accuracy. BERT-Freeze6 provides a more moderate reduction in training energy with minimal loss in F1, while BERT-Edge achieves only modest training-energy savings relative to the drop in predictive performance.

Taken together, these results distinguish architectural strategies that primarily reduce training cost from those that also reduce inference cost. DistilBERT lowers both training and inference energy, whereas pruning and layer freezing mainly reduce the cost of fine-tuning while leaving inference behavior largely unchanged.

5.4.2 Inference-time optimization

We next examine inference-time optimization techniques that do not require retraining the model. These methods modify runtime configuration and can therefore be applied directly at deployment. We evaluate reduced numerical precision, maximum sequence length truncation, and dynamic token

pruning. For each technique, we sweep the corresponding parameter and measure the resulting changes in model accuracy and inference energy. To facilitate comparison, all plots (Figures 3-6) use identical axis scales for energy and F1.

Precision reduction provides the most favorable trade-off among these methods. As shown in Figure 3, moving from FP32 to FP16 or BF16 dramatically reduces inference energy while leaving accuracy nearly unchanged. For example, the BERT baseline decreases from approximately 0.674 J/example in FP32 to 0.168 J/example in FP16, while F1 remains essentially constant (75.97 vs. 75.96). Similar behavior is observed for DistilBERT, where BF16 slightly improves F1 relative to FP32 while reducing inference energy by more than 4 $\times$. These results indicate that reduced precision is a highly effective inference optimization on the tested hardware.

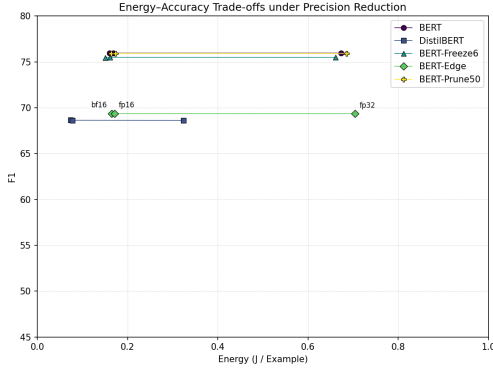


Figure 3: Precision reduction: inference energy per example vs. F1.

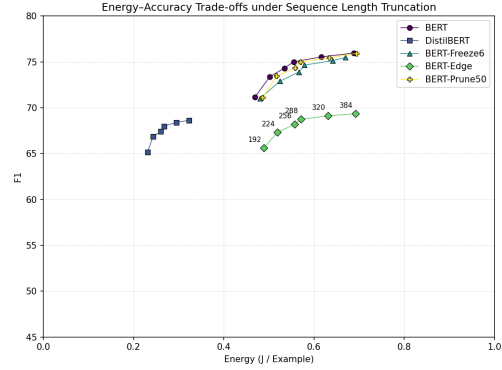


Figure 4: Sequence length truncation: inference energy per example vs. F1.

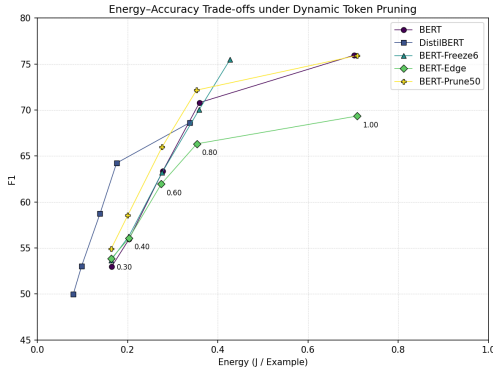


Figure 5: Dynamic token pruning: inference energy per example vs. F1.

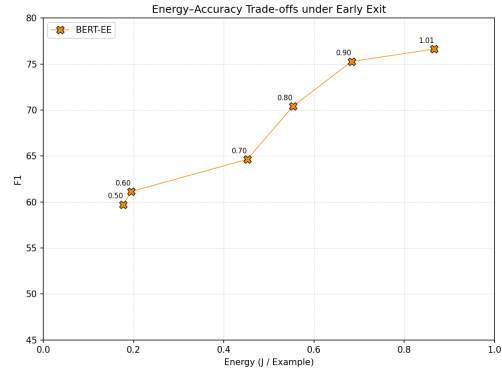


Figure 6: Early exit: inference energy per example vs. F1.

Sequence length truncation produces a different type of trade-off. Figure 4 shows a smooth Pareto frontier in which moderate truncation yields meaningful energy savings with limited accuracy loss, while more aggressive truncation leads to steeper degradation. On BERT-Edge, reducing the maximum sequence length from 384 to 320 tokens lowers inference energy by approximately 8.8% while decreasing F1 by only 0.34%. By contrast, reducing the sequence length from 224 to 192 tokens lowers energy by roughly 5.8% but incurs a larger 2.5% drop in F1, indicating a clear knee in the trade-off curve.

Dynamic token pruning yields a steeper energy-accuracy trade-off than the previous techniques. As shown in Figure 5, moderate pruning can yield substantial energy reductions, but accuracy degrades more rapidly. On BERT-Prune50, reducing the token keep ratio from 1.0 to 0.8 decreases energy per example by approximately 50% while reducing F1 by about 5%. More aggressive pruning shows diminishing returns: on DistilBERT, reducing the keep ratio from 0.8 to 0.6 lowers energy by about 21% but incurs an additional 8.6% drop in F1. This suggests that token pruning can achieve large energy reductions, but only over a relatively narrow accuracy-preserving regime. Notably, several inference-time configurations allow BERT-based models to approach the inference energy of DistilBERT while maintaining higher accuracy.

5.4.3 Early exit

Early exit is evaluated separately because it requires modifying and retraining the model architecture. Our BERT-EE variant attaches auxiliary classifier heads to intermediate transformer layers, allowing inference to terminate once a confidence threshold is reached. Figure 6 shows the resulting trade-offs as the early-exit confidence threshold varies. When the threshold is set to 1.01, BERT-EE never exits early and evaluates all layers, effectively reproducing full-depth inference while achieving an F1 score of 76.64 after the additional training epoch. Lower thresholds allow more examples to exit earlier, reducing inference energy. For example, lowering the threshold to 0.90 reduces energy per example by 21.1% while decreasing F1 by 1.77%. Further reductions continue to decrease energy, but with progressively larger accuracy degradation. Compared with token pruning, early exit provides a more adaptive trade-off by varying computation across inputs rather than uniformly reducing computation for every example. In our experiments, however, increasingly aggressive thresholds still lead to substantial accuracy loss, indicating that the benefits of early exit depend sensitively on the chosen operating point.

6 Analysis

6.1 Interpreting the energy–accuracy frontier

The experiments reveal two distinct ways of improving transformer efficiency. Architectural modifications primarily reduce the cost of fine-tuning, whereas inference-time techniques operate after training and only reduce deployment cost. For example, pruning and layer freezing lower training energy while leaving inference energy largely unchanged, whereas reduced precision and sequence-length truncation directly reduce inference-time cost. These differences help explain the structure of the observed energy–accuracy frontier. Precision reduction yields the most favorable trade-off because it lowers arithmetic cost without materially changing model representations, producing large energy savings with negligible accuracy loss. Sequence-length truncation produces a smoother Pareto frontier because it progressively removes input information. By contrast, token pruning and increasingly aggressive early exit tend to produce steeper trade-offs, since they remove or skip intermediate computation that may still carry useful contextual information. More broadly, the results show that different efficiency techniques act on different parts of the computational pipeline and therefore shift models along the frontier in qualitatively different ways.

6.2 Efficiency per accuracy loss

Figure 7 compares the relative efficiency of several techniques by plotting energy savings against accuracy loss relative to the BERT baseline. Points closer to the lower-right corner correspond to the most favorable trade-offs. Precision reduction dominates this region. Switching BERT from FP32 to FP16 reduces inference energy by roughly 75% while leaving F1 essentially unchanged. Moderate sequence truncation also remains attractive: for example, BERT with sequence length 320 achieves a smaller but still favorable reduction in energy with only a limited loss in accuracy. Other configurations are clearly less efficient. BERT-Edge with sequence length 288 achieves only modest energy savings while incurring a substantially larger loss in F1, and aggressive early-exit settings for BERT-EE achieve large energy reductions only at the cost of the largest accuracy losses in the plot. Token-pruning configurations occupy an intermediate region: some, such as BERT-Freeze6 with 0.8 token pruning, remain competitive, but they generally incur larger accuracy penalties than precision reduction for comparable savings. Overall, the figure reinforces that precision reduction is the most efficient way to move along the frontier, while sequence truncation offers a useful secondary option and token pruning or early exit are better suited to settings where larger accuracy losses are acceptable.

6.3 Deployment regime analysis

Figure 8 shows how the relative efficiency of different configurations depends on the expected inference workload. Each curve represents total energy consumption as a function of the number of inference queries, computed as the sum of one-time training energy and cumulative inference energy per example. When the number of queries is small, total energy is dominated by training cost, making lower-training-energy configurations such as DistilBERT and BERT-Prune50 with token pruning more attractive. As the number of queries grows, cumulative inference energy dominates, so configurations with lower per-example inference cost become increasingly favorable. In this regime,

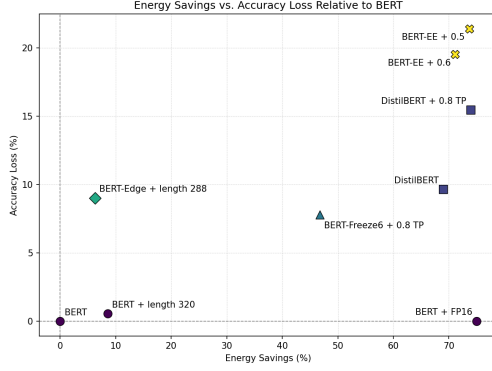


Figure 7: Energy savings vs. accuracy loss relative to the BERT baseline.

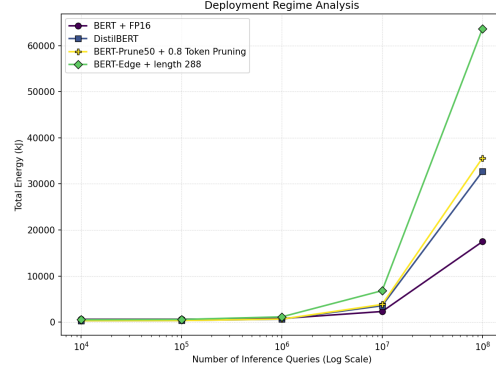


Figure 8: Total energy vs. number of inference queries across representative configurations.

BERT with FP16 precision becomes particularly attractive because its much lower inference energy outweighs its higher training cost over time. These results show that the best efficiency strategy depends not only on model architecture, but also on the expected deployment regime.

7 Conclusion

In this project, we investigated the energy–accuracy trade-offs of transformer-based NLP models across both architectural and inference-time efficiency strategies. We developed a controlled benchmarking framework that measures GPU power consumption during training and inference, enabling direct comparison within a unified experimental setting. We evaluated several BERT-derived architectural variants alongside inference-time optimizations including precision reduction, sequence length truncation, token pruning, and early exit. Our results show that these techniques affect different parts of the computational pipeline: architectural modifications primarily influence training energy, while inference-time techniques shift the deployment-time energy–accuracy frontier.

Several key patterns emerge from our experiments. First, architectural compression methods such as distillation and pruning can substantially reduce training energy while leaving inference cost largely unchanged. Second, precision reduction consistently provides the most favorable inference-time trade-off, yielding large energy savings with negligible impact on accuracy. Third, sequence length truncation produces a smooth Pareto frontier between energy consumption and predictive performance. Finally, the deployment regime analysis shows that the most efficient strategy depends on the expected workload: architectures with lower training energy are preferable when inference volume is small, whereas configurations with lower per-example inference energy dominate at large deployment scales.

Several limitations should be noted. Our experiments focus on BERT-family models, which are smaller than many contemporary language models; evaluating whether these trends hold for larger architectures would be an important extension. In addition, the study considers a single task (SQuAD v2), and the generality of the findings across other NLP tasks remains an open question. Some techniques could also be explored more thoroughly. For example, more systematic pruning schedules or improved early-exit implementations might yield different trade-offs. Finally, energy consumption was estimated through GPU power sampling; more precise measurements could be obtained through hardware-level power telemetry or external power instrumentation.

Future work could extend this analysis in several directions. Exploring lower-precision formats such as FP8 or 4-bit quantization may reveal additional gains. Evaluating additional architectures, including parameter-sharing models such as ALBERT [17], would help determine whether the observed trends generalize across transformer families. Dynamic inference methods also warrant further study, including improved early-exit and token pruning. Expanding the framework could help establish systematic benchmarks for energy-efficient NLP systems.

References

- [1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Informa-*

- tion Processing Systems (NeurIPS), 2017.
- [2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019.
 - [3] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.
 - [4] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.
 - [5] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack W. Rae, and Laurent Sifre. Training compute-optimal large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.
 - [6] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. In *Association for Computational Linguistics (ACL)*, 2019.
 - [7] David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluís-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*, 2021.
 - [8] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green ai. *Communications of the ACM*, 63(12):54–63, 2020.
 - [9] David Anugraha, Genta Indra Winata, Chenyue Li, Patrick Amadeus Irawan, and En-Shiun Annie Lee. Proxylm: Predicting language model performance on multilingual tasks via proxy models. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, 2025.
 - [10] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.
 - [11] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling bert for natural language understanding. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2020.
 - [12] Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. The lottery ticket hypothesis for pre-trained bert networks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS 20)*, 2020.
 - [13] Victor Sanh, Thomas Wolf, and Alexander M. Rush. Movement pruning: Adaptive sparsity by fine-tuning. In *34th International Conference on Neural Information Processing Systems (NeurIPS 2020)*, pages 20378–20389, 2020.
 - [14] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training. In *International Conference on Learning Representations (ICLR)*, 2018.
 - [15] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Llm.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

- [16] Zhenya Ji and Ming Jiang. A systematic review of electricity demand for large language models: Evaluations, challenges, and solutions. *Renewable and Sustainable Energy Reviews*, 225, 2025.
- [17] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations (ICLR)*, 2020.
- [18] Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout. In *International Conference on Learning Representations (ICLR)*, 2020.
- [19] Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. Deebert: Dynamic early exiting for accelerating bert inference. In *Association for Computational Linguistics (ACL)*, 2020.
- [20] Saurabh Goyal, Anamitra Roy Choudhury, Saurabh M. Raje, Venkatesan T. Chakaravarthy, Yogish Sabharwal, and Ashish Verma. Power-bert: Accelerating bert inference via progressive word-vector elimination. In *International Conference on Machine Learning (ICML)*, 2020.
- [21] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning (ICML)*, 2019.
- [22] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don’t know: Unanswerable questions for SQuAD. In *Association for Computational Linguistics (ACL)*, 2018.
- [23] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2016.

A Initial Experimental Framework (V1)

Prior to the final experimental design presented in this report, an earlier version of the project (V1⁵) was developed until the Project Milestone. This initial phase focused exclusively on the BERT-base model [2] and inference-time efficiency techniques, and included a custom evaluation harness designed to systematically measure performance, latency, and energy consumption across multiple NLP tasks. Experiments were conducted on SST-2⁶, QQP⁷, and SQuAD v1.1⁸, and explored inference-time reduction strategies, including (i) sequence length truncation via fixed maximum token limits, (ii) encoder depth variation by retaining the first k layer, (iii) reduced floating-point precision at inference, and their combination.

While this framework produced a substantial set of experiments and visualizations, the project was later restructured and rebuilt from scratch in order to pursue the broader and more systematic exploration of the energy–accuracy trade-offs in NLP systems presented in this report. Figures 9–19 in this appendix document the experiments and results produced during the V1 phase of the project, illustrating the experimental infrastructure and exploratory analyses that informed our final methodology.

⁵<https://github.com/thibaudclement/nlpfrontiers-research/tree/main/v1>

⁶<https://huggingface.co/datasets/stanfordnlp/sst2>

⁷<https://huggingface.co/datasets/nyu-mll/glue/viewer/qqp>

⁸<https://huggingface.co/datasets/rajpurkar/squad>

A.1 SST-2

Sequence length truncation exhibits a wide flat region in the accuracy–energy frontier (Figure 9): reducing the maximum input length from 128 to 48 tokens decreases energy consumption by approximately $12\times$ while preserving accuracy. Below 48 tokens, accuracy degrades progressively. In contrast, varying encoder depth leads to rapid and monotonic accuracy degradation (Figure 10), suggesting that depth is less redundant than input length or arithmetic precision for this task. Numerical precision reduction from FP32 to FP16 reduces energy consumption by over $4\times$ without measurable accuracy loss (Figure 11). Combining precision reduction (FP16) with sequence length truncation (48 tokens) yields approximately $28\times$ lower energy consumption relative to the full baseline while preserving accuracy (Figure 12).

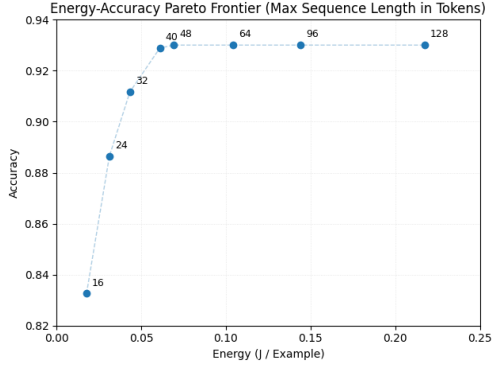


Figure 9: SST-2: Energy-Accuracy Pareto Frontier (Max Sequence Length).

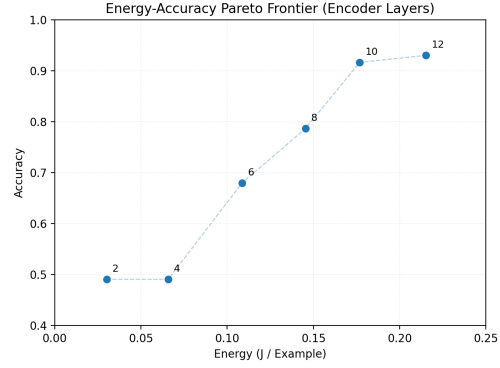


Figure 10: SST-2: Energy-Accuracy Pareto Frontier (Encoder Layers).

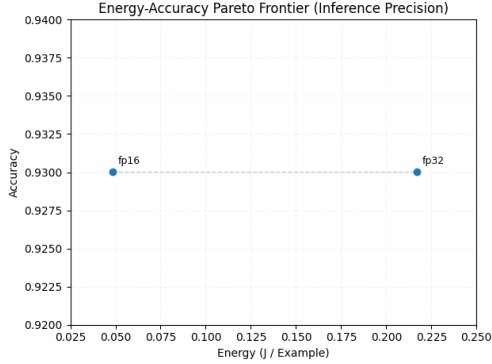


Figure 11: SST-2: Energy-Accuracy Pareto Frontier (Inference Precision).

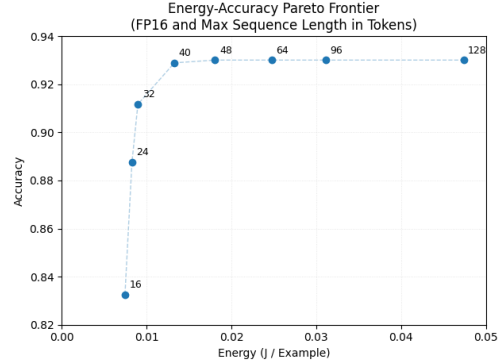


Figure 12: SST-2: Energy-Accuracy Pareto Frontier (FP16 and Max Sequence Length).

A.2 QQP

Sequence length truncation again exhibits a wide flat region in the accuracy–energy frontier (Figure 13): reducing the maximum input length from 128 to 48 tokens decreases energy consumption by approximately $4\times$ while preserving accuracy. Below 48 tokens, accuracy degrades progressively. In contrast, varying encoder depth leads to substantial and largely monotonic accuracy degradation (Figure 14), suggesting that depth remains a critical component of model capacity for this task. Numerical precision reduction from FP32 to FP16 reduces energy consumption by roughly $3\times$ without measurable accuracy loss (Figure 15). Combining precision reduction (FP16) with sequence length truncation (48 tokens) yields over $9\times$ lower energy consumption relative to the full baseline while preserving accuracy (Figure 16).

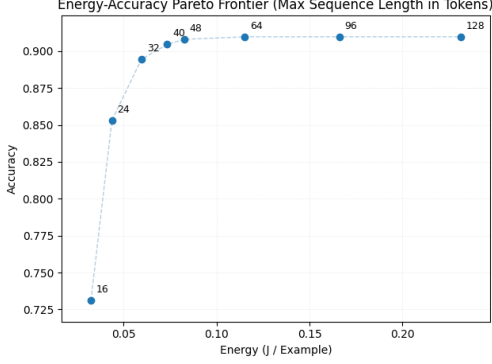


Figure 13: QQP: Energy-Accuracy Pareto Frontier (Max Sequence Length).

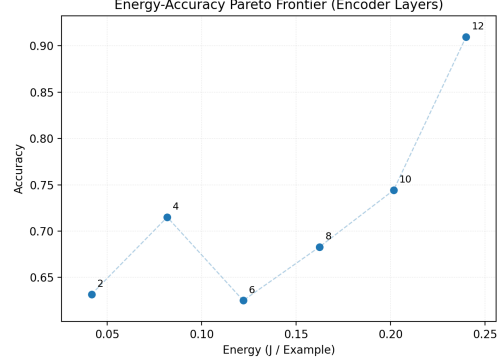


Figure 14: QQP: Energy-Accuracy Pareto Frontier (Encoder Layers).

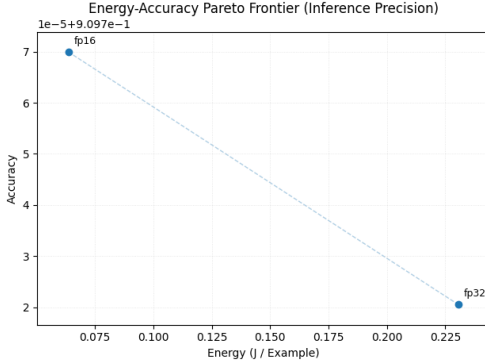


Figure 15: QQP: Energy-Accuracy Pareto Frontier (Inference Precision).

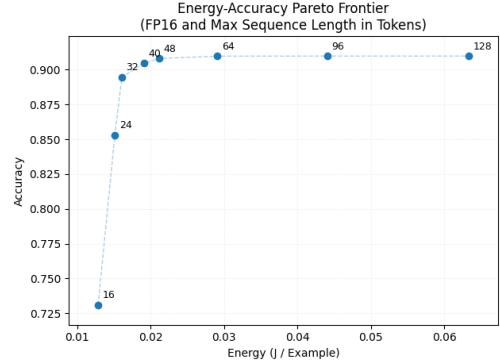


Figure 16: QQP: Energy-Accuracy Pareto Frontier (FP16 and Max Sequence Length).

A.3 SQuAD v1.1

Experiments on SQuAD v1.1 reveal that the energy–F1 trade-off is less clearly structured than in the classification tasks with this specific evaluation pipeline. Sequence length variation produces only modest changes in F1 despite substantial differences in energy consumption (Figure 17), suggesting that the evaluation metric is relatively insensitive to moderate truncation within the tested range. Numerical precision reduction from FP32 to FP16 again lowers energy consumption without measurable impact on F1 (Figure 19). In contrast, varying encoder depth leads to large and irregular drops in F1 (Figure 18), indicating that aggressive architectural reduction quickly undermines extractive question answering performance. Overall, these results suggest that the energy–F1 frontier for SQuAD v1.1 is difficult to interpret in this experimental setup, as the task appears highly sensitive to architectural changes while remaining comparatively insensitive to several inference-time efficiency techniques.

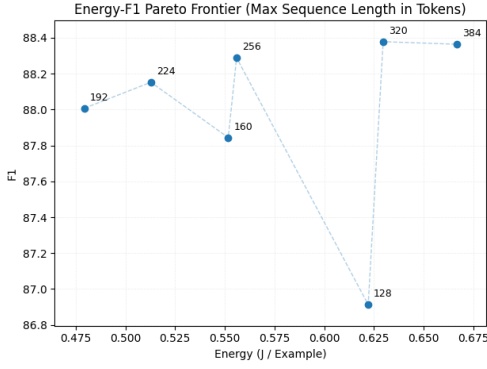


Figure 17: SQuAD v1.1: Energy-Accuracy Pareto Frontier (Max Sequence Length).

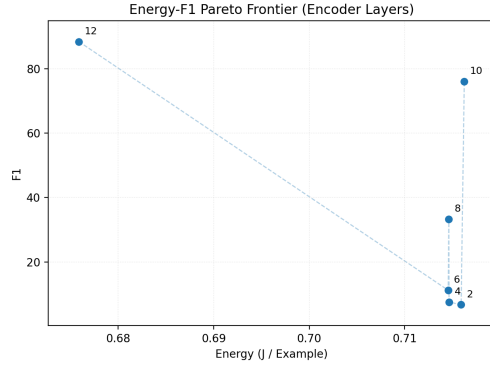


Figure 18: SQuAD v1.1: Energy-Accuracy Pareto Frontier (Encoder Layers).

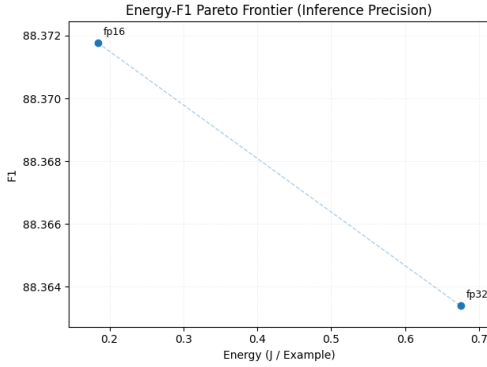


Figure 19: SQuAD v1.1: Energy-Accuracy Pareto Frontier (Inference Precision).