Humboldt-Universität zu Berlin
Mathematisch-Naturwissenschaftliche Fakultät
Institut für Informatik

Lean Transformers for Biomedical NER: Evaluating Pruning, Dynamic Batching, and Quantization Toward Sustainable Inference

Expose zur Masterarbeit

eingereicht von:	Julius Seiferth
geboren am:	05.04.2002
geboren in:	Bad Salzungen
Matrikelnummer:	642962
Gutachter/innen:	Prof. Dr. Ulf Leser

Inhaltsverzeichnis

1 Introduction

The volume of biomedical literature has continued to expand at a record pace—PubMed now indexes more than 35 million citations and releases over one million new records annually, creating an urgent need for automated text–mining pipelines that can keep domain experts abreast of the latest findings [1]. Transformer–based language models, popularised by the seminal *Attention Is All You Need* architecture [33], have become the de-facto backbone of state-of-the-art named-entity recognition (NER) systems; domain-adapted variants such as BioBERT push performance further by pre-training on biomedical corpora [22]. Tools like HunFlair2 now deliver accurate, end-to-end recognition and normalisation of genes, species and diseases without task-specific engineering, enabling large-scale information extraction in life-science workflows [32].

Yet the computational cost of running Transformer models has raised widespread environmental and financial concerns: training a single large NLP model can emit as much CO2 as five American cars over their lifetime [31], and inference already dominates the energy bill for many production systems [13]. The "Green AI" agenda therefore calls for research that values efficiency on par with accuracy [30], a message echoed by recent carbon-footprint studies on large-language-model deployment [10]. Regulatory momentum is also accelerating; the EU Artificial Intelligence Act urges transparency over energy use and resource consumption for high-impact AI applications [2].

Model-side optimisations such as structured pruning [12], movement pruning [29], integer-only quantisation [17], and 8-bit BERT compression [38] have shown promise in reducing compute and memory demands, while serving-time techniques like continuous dynamic batching ($vLLM$) can boost throughput by an order of magnitude without extra latency [8]. However, their combined impact on energy efficiency for domain-specific NER remains under-explored. Motivated by these gaps, this thesis provides the first systematic comparison of pruning, dynamic batching and quantisation—individually and in combination—on HunFlair2 inference, reporting both task performance and fine-grained energy metrics.

2 Fundamentals and Related Work

A sound understanding of this thesis requires situating its experiments within the broader scientific context of neural-network architectures and efficiency research. Accordingly, the present chapter first reviews the Transformer framework and the HunFlair2 biomedical NER toolkit before surveying three complementary optimisation strategies—pruning, dynamic batching and quantisation—that aim to reconcile state-of-the-art accuracy with sustainable computation [12, 18, 33].

2.1 Transformer Models

The Transformer is a deep-learning architecture introduced by Vaswani *et al.* in 2017 that forgoes recurrence in favour of multi-head self–attention, yielding highly parallelisable sequence processing and superior performance on translation benchmarks [33]. A canonical Transformer consists of stacked encoder and decoder blocks, each combining a self-attention sublayer with a position-wise feed-forward network, wrapped by residual connections and layer normalisation to ensure stable training [33]. Self-attention allows every token to attend to all other positions in a single input sentence, enabling efficient modelling of long-range dependencies relative to recurrent neural networks [33]. Bidirectional encoder-only adaptations such as BERT [9] and RoBERTa [23] pre-train on large corpora with a masked-language objective and then fine-tune on downstream tasks. Decoder-only variants underlie generative systems like GPT-3, which showcased few-shot abilities at unprecedented quality [6], whereas unified encoder–decoder models such as T5 treat every NLP task in a text-to-text fashion [28]. Domain-specific pre-training further extends these gains; BioLinkBert, for instance, markedly outperforms general BERT models on biomedical NER, relation extraction and QA [21]. Prior to the Transformer era, transfer learning in NLP largely relied on recurrent models like ULMFiT [14], but these have since been superseded by Transformer variants. Despite their success, Transformers are computationally and energy intensive, motivating research into pruning, dynamic computation and quantisation—optimisations that this thesis will benchmark.

2.2 HunFlair2

HunFlair2 is an open-source biomedical NER+entity-normalisation toolkit built on the Flair NLP framework [4]. It employs a single Transformer-based sequence tagger that jointly recognises five biomedical entity types—genes/proteins, chemicals, diseases, species and cell lines—leveraging contextual embeddings from domain-adapted language models such as BioBERT [21, 32]. Sharing parameters across entity types through multi-task learning reduces memory consumption and improves robustness on heterogeneous corpora []. The model integrates a neural linker that maps each mention to standard identifiers (e.g. MeSH, Entrez Gene), producing normalised outputs suitable for downstream knowledge-base integration [32]. In a 26-corpus benchmark, HunFlair2 achieved the highest average F_1 among nine competing biomedical NER/NEN tools, highlighting its cross-corpus generalisation [32]. These gains arise from extensive cross-corpus training, residual character-level embeddings and a CRF decoding layer added on top of Transformer representations [36]. Thanks to its simple Python API, GPU-accelerated PyTorch backend and permissive MIT licence, HunFlair2 lets researchers quickly experiment with efficiency techniques such as pruning, dynamic batching and quantisation in biomedical NER [4, 36].

2.3 Pruning

Neural network *pruning* is a model compression technique that involves removing unnecessary parameters or structures (such as weights, neurons, or filters) from a trained model to reduce its size and computational requirements. This idea dates back to early work like Optimal Brain Damage by LeCun *et al.* (1990), which showed that removing low-saliency weights can have minimal impact on accuracy [20]. Modern approaches have revived and expanded pruning for deep networks; for instance, Han *et al.* (2015)

demonstrated that iteratively pruning and retraining a network can reduce the parameter count by an order of magnitude without significant loss in accuracy [12]. Pruning can be *unstructured* (removing individual weights) or *structured* (removing entire neurons or channels), the latter of which is often more hardware-efficient. By eliminating redundant computations and memory accesses, pruning not only accelerates inference but also lowers energy consumption. For example, in NLP models like BERT, fine-tuning with pruning methods (e.g., movement pruning) can yield highly sparse models that maintain performance while using considerably less computational resources [29].

2.4 Dynamic Batching

Dynamic batching is a technique to improve the efficiency of neural network inference by aggregating multiple incoming requests and processing them together as a single batch. Unlike static batching with a fixed batch size, dynamic batching adaptively forms batches on the fly (often based on short time windows or queue thresholds) to maximize hardware utilization without incurring excessive latency. Grouping several requests into one forward pass allows the model (especially on parallel hardware like GPUs or TPUs) to amortize memory and compute overheads across multiple inputs, significantly increasing throughput per unit of energy. This approach leads to better resource utilization. For instance, the Clipper serving system employed adaptive batching to boost throughput while meeting strict latency constraints [7]. In the context of large language models, recent systems like *vLLM* implement continuous dynamic batching to achieve order-of-magnitude higher token throughput while maintaining low latency [8]. Overall, dynamic batching reduces idle time and ensures that compute units operate near their optimal capacity, thereby improving energy efficiency per query.

2.5 Quantization

Quantization is the process of reducing the numerical precision of a model's parameters and operations, typically by converting 32-bit floating-point weights and activations to lower bit-width representations such as 8-bit integers. This compression technique dramatically lowers the memory footprint of neural networks (e.g., 8-bit quantization cuts memory usage by $4\times$) and enables the use of faster, low-precision arithmetic instructions. By replacing high-precision operations with low-cost integer or fixed-point operations, quantization can substantially increase inference speed and reduce energy consumption [18]. Proper quantization (via post-training calibration or quantization-aware training) can preserve most of the model's accuracy, as demonstrated by Zafrir *et al.* (2019) who achieved near-original performance with an 8-bit BERT model [38]. Moreover, modern hardware units (e.g., NVIDIA Tensor Cores and Google TPUs) are optimized for low-precision math, which amplifies the efficiency gains of quantization. In extreme cases, researchers have even explored 1-bit or binary neural networks [15], which maximize efficiency (though typically at the cost of some accuracy). Overall, quantization is a crucial technique for deploying deep models in resource-constrained environments, offering a favorable trade-off between model fidelity and operational efficiency.

3 Evaluation

This section briefly presents the biomedical NER datasets (MedMentions, Bio-ID (Bio-Creative VI), tmVar3.0, SourceData- NLP, and the Variome Corpus) and the hardware telemetry interfaces (NVML and RAPL) used to evaluate model accuracy and energy consumption.

3.1 Datasets

This study evaluates model performance on five contemporary biomedical NER corpora: MedMentions, Bio-ID (BioCreative VI), tmVar3.0, SourceData-NLP, and the Variome Corpus [3, 5, 24, 35, 37].

MedMentions comprises 4 392 PubMed titles and abstracts published in 2016, exhaustively annotated with more than 350 000 UMLS-linked concept mentions spanning 21 semantic types [24]. Its breadth and size make it a strong benchmark for broad-coverage biomedical NER.
Bio-ID supplies figure-legend text (plus full-text context) curated for eight entity classes—including gene/protein, miRNA, small-molecule, organism, cell line and tissue mentions—within the BioCreative VI Interactive ID Assignment track [5]. The legend modality complements abstract- and full-text corpora and stresses fine-grained normalisation.
tmVar 3.0 offers 500 full-text PubMed articles densely annotated for genetic variant mentions of diverse types (e. g. SNPs, copy-number variants) and normalised to dbSNP or ClinGen allele identifiers [37]. This corpus targets sequence-variant recognition and bridging to reference databases.
SourceData-NLP collects figure-legend annotations from 18 689 figures in 3 223 molecular- and cell-biology papers, totalling over 620 000 entity mentions across eight biomedical classes, together with experimental-role metadata [3]. Its scale and multimodal focus enable evaluation in settings with long, context-rich sentences.
Variome Corpus is a domain-specific resource of inherited-colorectal-cancer full-text articles annotated with 11 entity types (variants, genes, diseases, etc.) and 13 relations, providing a challenging low-resource benchmark for variant-centric NER and relation extraction [35].

3.2 Energy Measurement Tools

To quantify the energy footprint of each experimental run we rely on two software telemetry interfaces: NVIDIA's Management Library (NVML) for GPUs and Intel's Running Average Power Limit (RAPL) for CPUs [16, 25]. NVML is the C-level API that powers `nvidia-smi`; it delivers real-time readings of GPU power draw, temperature and utilisation that can be polled programmatically at millisecond resolution [25]. By integrating NVML sampling (e.g. 100 ms interval) into training or inference loops, researchers obtain per-batch energy profiles without external meters [34]. RAPL, exposed via model-specific registers, accumulates joule-level energy consumption for the CPU package, cores and DRAM domains, enabling precise CPU-side measurements with negligible overhead [16]. These counters can be accessed through Linux `perf` or user-space libraries, providing millisecond-granularity power data for CPU-bound inference workloads [16]. Comparative studies show strong correlation between NVML/RAPL readings and external power meters, validating their use for energy-efficiency benchmarking in machine-learning research [19].

4 Implementation

4.1 Dataset Assembly

To create a heterogeneous yet compact benchmark, we merge the sentence–level splits of MedMentions (genes/proteins, diseases, species) [24], Bio-ID (figure-legend genes/proteins, species) [5], tmVar 3.0 (gene/protein symbols in full texts) [37], SourceData-NLP (legends with gene products, diseases, species) [3], and the Variome Corpus (inherited-cancer texts with genes and diseases) [35] into a single evaluation set. All corpora are converted to a uniform scheme and collapsed onto three unified labels(`GENE`, `SPECIES`, and `DISEASE`)without further re-training, as the study focuses strictly on inference-time behaviour.

4.2 Baseline Inference Setup

We run the off-the-shelf HunFlair2 tagger provided by the Flair framework [11] on the merged corpus, recording its latency, F_1, memory footprint, and power draw. Because HunFlair2 is already pretrained on a broad mix of biomedical texts, no fine-tuning or domain adaptation is performed; this establishes a "vanilla" inference reference.

4.3 Single Optimisation Passes

Pruning — Weights are removed by magnitude-based movement pruning at sparsity levels [29]; the pruned checkpoints are evaluated *as-is* to isolate pure inference effects.

Quantisation — We employ two precision settings: float16 (FP16) for rapid prototyping and int8 dynamic quantisation following the Q8BERT recipe [38], letting us compare moderate versus aggressive bit-width reduction.

4.4 Dynamic Batching

— Inference is served via *TorchServe* using its native dynamic-batching feature (`max_batch_size` = 16, `max_batch_delay` = 5 ms), which aggregates requests that arrive within the delay window into a single micro-batch. This strategy amortises pre-/post-processing overhead and drives the GPU at higher utilisation; prior evaluations report up to 10–15 $\times$ throughput gains for encoder-style Transformer models under similar settings [26, 27].

4.5 Combined Pipeline

For the end-to-end test we first quantise the model to int8, then prune to 50 % sparsity, and finally deploy it via dynamic batching. This ordering minimises memory traffic before sparsity masking and leaves batching logic hardware-agnostic.

4.6 Logging and Metrics

GPU power, utilisation and VRAM are sampled every 100 ms via NVIDIA's NVML API [25], while CPU package energy is captured with Intel's RAPL counters [16]. From these traces we derive average/peak power (W), total energy (J), maximum RAM/VRAM (MB), wall-clock runtime (s) and micro-averaged F_1.

Literatur

[1] 2024 PubMed baseline release. `https://www.ncbi.nlm.nih.gov/feed/rss.cgi?ChanKey=PubMedNews`, 2024. Accessed 2 July 2025.

[2] Regulation (EU) 2024/1689 on harmonised rules for artificial intelligence (AI act). `https://eur-lex.europa.eu/eli/reg/2024/1689/oj`, 2024. Official Journal of the European Union.

[3] J. Abreu-Vicente, H. Sonntag, T. Eidens, and T. Lemberger. The sourcedata-nlp dataset: Integrating curation into scientific publishing for training large language models. *arXiv preprint arXiv:2310.20440*, 2023.

[4] A. Akbik, T. Bergmann, D. Blythe, K. Rasul, S. Schweter, and R. Vollgraf. Flair: An easy-to-use framework for state-of-the-art nlp. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 54–59, 2019.

[5] C. Arighi, L. Hirschman, T. Lemberger, S. Bayer, R. Liechti, D. Comeau, and C. Wu. Bio-id track overview. In *Proceedings of the BioCreative VI Workshop*, volume 482, page 376, 2017.

[6] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33*, 2020.

[7] D. Crankshaw, X. Wang, G. Zhou, M. J. Franklin, J. E. Gonzalez, and I. Stoica. Clipper: A low-latency online prediction serving system. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 613–627, 2017.

[8] C. Daniel, Y. Jiang, Y. Ge, C. Wei, B. Jiang, and I. Stoica. vllm: Easy and fast llm serving with continuous batching. In *Proceedings of the 29th ACM Symposium on Operating Systems Principles*, 2023.

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2019.

[10] A. Faiz, S. Kaneda, R. Wang, R. Osi, P. Sharma, F. Chen, and L. Jiang. LLMCarbon: Modeling the end-to-end carbon footprint of large language models. *arXiv preprint arXiv:2309.14393*, 2023.

[11] Flair NLP Team. *HunFlair2 Documentation.* Humboldt-Universität zu Berlin, 2025. Available online: `https://flairnlp.github.io/flair/master/tutorial/tutorial-hunflair2/overview.html` (accessed 2 July 2025).

[12] S. Han, J. Pool, J. Tran, and W. J. Dally. Learning both weights and connections for efficient neural networks. In *Advances in Neural Information Processing Systems*, pages 1135–1143, 2015.

[13] P. Henderson, J. Hu, J. Romoff, E. Brunskill, D. Jurafsky, and J. Pineau. Towards the systematic reporting of the energy and carbon footprints of machine learning. *Journal of Machine Learning Research*, 21(248):1–43, 2020.

[14] J. Howard and S. Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 328–339, 2018.

[15] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks. In *Advances in Neural Information Processing Systems*, pages 4107–4115, 2016.

[16] Intel Corporation. *Running Average Power Limit (RAPL) Interface for Energy Monitoring*, 2022. Available: `https://software.intel.com/content/www/us/en/developer/articles/technical/running-average-power-limit-energy-reporting.html`.

[17] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of CVPR 2018*, pages 2704–2713, 2018.

[18] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2704–2713, 2018.

[19] N. Jovanović, T. Đ urić, and N. Anđelić. Comparison and analysis of software and hardware energy measurement methods for machine learning workloads. *Computer Science and Information Systems*, 2025(to appear), 2025. Preprint: `https://comsis.org/pdf.php?id=17544`.

[20] Y. LeCun, J. S. Denker, and S. A. Solla. Optimal brain damage. In *Advances in Neural Information Processing Systems*, pages 598–605, 1990.

[21] J. Lee, S. Kim, W. Yoon, S. Kim, D. Kim, C. So, and J. Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020.

[22] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang. Biobert: A pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020.

[23] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. In *arXiv preprint arXiv:1907.11692*, 2019.

[24] S. Mohan and D. Li. Medmentions: A large biomedical corpus annotated with umls concepts. In *Proceedings of the Automated Knowledge Base Construction Conference*, 2019.

[25] NVIDIA Corporation. *NVIDIA Management Library (NVML) Developer Guide*, 2023. Available: `https://developer.nvidia.com/management-library-nvml`.

[26] NVIDIA Corporation. *Concurrent Inference and Dynamic Batching with NVIDIA Triton Inference Server*, 2024. Example guide, accessed 3 July 2025.

[27] PyTorch Foundation. *Batch Inference with TorchServe.* PyTorch, 2024. Documentation, accessed 3 July 2025.

[28] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67, 2020.

[29] V. Sanh, T. Wolf, and A. Rush. Movement pruning: Adaptive sparsity by fine-tuning. In *Advances in Neural Information Processing Systems*, 2020.

[30] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni. Green AI. *Communications of the ACM*, 63(12):54–63, 2020.

[31] E. Strubell, A. Ganesh, and A. McCallum. Energy and policy considerations for deep learning in NLP. In *Proceedings of ACL 2019*, pages 3645–3650, 2019.

[32] M. Sänger, S. Garda, X. D. Wang, L. Weber-Genzel, P. Droop, B. Fuchs, A. Akbik, and U. Leser. Hunflair2 in a cross-corpus evaluation of biomedical named entity recognition and normalization tools. In *Bioinformatics*, volume 40, page btae564, 2024.

[33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008, 2017.

[34] K. Velicheti. nvml-power: Measuring NVIDIA gpu power consumption via NVML. `https://github.com/kajalv/nvml-power`, 2023.

[35] K. Verspoor, A. Jimeno Yupes, L. Cavedon, T. McIntosh, A. Herten-Crabb, Z. Thomas, and J.-P. Plazzer. Annotating the biomedical literature for the human variome. *Database*, 2013:bat019, 2013.

[36] L. Weber, M. Sänger, J. Münchmeyer, M. Habibi, U. Leser, and A. Akbik. Hunflair: an easy-to-use tool for state-of-the-art biomedical named entity recognition. *Bioinformatics*, 37(17):2792–2794, 2021.

[37] C.-H. Wei, A. Allot, K. Riehle, A. Milosavljevic, and Z. Lu. tmvar 3.0: An improved variant concept recognition and normalization tool. *Bioinformatics*, 38(18):4449–4456, 2022.

[38] O. Zafrir, G. Boudoukh, P. Izsak, and M. Wasserblat. Q8bert: Quantized 8bit bert. *arXiv preprint arXiv:1910.06188*, 2019.