CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Methods for Optimizing BERT Model on Edge Devices

Accelerating Biomedical NLP with Pruned and Quantized BERT Models

Master's thesis in Complex Adaptive Systems

Atefeh Mirzabeigi
Amir Ali Barani

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2025

Master's thesis 2025

Methods for Optimizing BERT Model on Edge Devices

Accelerating Biomedical NLP with Pruned and Quantized BERT Models

Atefeh Mirzabeigi
Amir Ali Barani

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2025

A Chalmers University of TechnologyLaTeX
Accelerating Biomedical NLP with Pruned and Quantized BERT Models
Atefeh Mirzabeigi, Amir Ali Barani

Supervisor: Mehrdad Farahani, Computer science and engineering
Advisor: Miguel Carmona, Michaël Ughetto, AstraZeneca
Examiner: Richard Johansson, Computer science and engineering
Examiner: Mats Granath, Physics

Master's Thesis 2025
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Description of the picture on the cover page (if applicable)

Typeset in LaTeX
Gothenburg, Sweden 2025

A Chalmers University of TechnologyLaTeX
Accelerating Biomedical NLP with Pruned and Quantized BERT Models
Atefeh Mirzabeigi, Amir Ali Barani
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Named-entity recognition (NER) of clinical efficacy endpoints in oncology abstracts supports downstream discovery pipelines at AstraZeneca. Yet, the fine-tuned transformer models currently used are too slow and over parameterized for large-scale CPU deployment. This thesis evaluated whether post-training model compression techniques can accelerate inference without retraining or harming extraction quality. In the first stage of this project, standard BERT and BioBERT were individually pruned with a three-stage, Fisher-guided structured pruning workflow at three levels of sparsity. Subsequently, in the second stage, dynamic 8-bit integers quantization using ONNX Runtime was applied to standard BERT, BioBERT, and DistilBERT. The third stage involved combining both pruning and quantization, further optimizing the pre-trained standard BERT and BioBERT transformers. Experiments were run on annotated MEDLINE sentences covering 25 efficacy labels, with F1 score and inference latency per sample serving as primary metrics.
A 25% structured-sparsity level yielded no measurable drop in F1 score, and the additional 8-bit integers dynamic step cut latency further. The best configuration, 25%-pruned + 8-bit integers BioBERT, reduced mean CPU inference time from 32.52 ms to 12.02 ms (2.6-fold speed-up) while accuracy fell only from 0.982 to 0.980 and F1 score from 0.954 to 0.948.
The Post-training structured pruning combined with 8-bit integers dynamic quantization makes the oncology-NER pipeline about three times faster in inference time on standard CPUs without compromising the extraction quality or needing special hardware or libraries.

Keywords: Natural language processing, Named entity recognition, Post-training quantization, Structured pruning, Model compression

Acknowledgements

We want to start by expressing our gratitude to AstraZeneca for allowing us to work on this thesis and for being among the kind and supportive people, even for a short time. Moreover, to provide all resources and support to successfully finish this project and learn more about real applications in machine learning and AI in industry. We are especially grateful to our supervisors, Mehrdad Farahani, Michaël Ughetto, and Miguel Carmona, for their invaluable guidance, support, and dedication throughout the project. Additionally, we would like to thank our examiner, Richard Johansson, for his insightful guidance and contributions, which have helped us complete this project.

Atefeh Mirzabeigi & Amir Ali Barani , Gothenburg, 2025-06-12

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis:

AI	Artificial Intelligence
BERT	Bidirectional Encoder Representations from Transformers
CPU	Central Processing Unit
FFN	Feed-Forward Network
FLOPs	Floating-Point Operations
GELU	Gaussian Error Linear Unit
GPU	Graphics Processing Unit
I2E	Interactive Information Extraction
KD	Knowledge Distillation
LLM	Large Language Model
LSTM	Long Short-Term Memory
MEDLINE	MEDLINE biomedical citations database
MLM	Masked Language Modelling
NER	Named-Entity Recognition
NLP	Natural Language Processing
NSP	Next-Sentence Prediction
ONNX	Open Neural Network eXchange
PTQ	Post-Training Quantisation
QAT	Quantisation-Aware Training
RNN	Recurrent Neural Network

Contents

List of Figures

List of Tables

1

Introduction

The idea of creating an artificial intelligence that mirrors the capability of the human brain has likely existed for millennia. A compelling example of this dream can be found in the ancient Greek myth of Talos, as told by the poets Hesiod and Homer around 700 B.C. [1] The myth describes Talos as a giant bronze man built by Hephaestus, the Greek god of invention and blacksmithing. Tasked with guarding the island of Crete, Talos would circle the island three times a day, hurling boulders at approaching enemy ships.

While Talos was a product of imagination and divine craftsmanship, modern advancements in artificial intelligence (AI) have transformed this ancient dream into reality. Over the past century, AI has transformed from simple rule-based systems to sophisticated machine learning models that can learn and adapt from data. Among these achievements, machine learning has become a crucial part of AIs success. It allows systems to recognize patterns, make decisions, and generate human-like language. Within machine learning models, natural language models such as GPT-3/4, BERT [2], RoBERTa [3], and others have shown impressive performance in solving complex tasks. They have repeatedly yielded better outcomes over traditional methods in areas such as text classification, named entity recognition (NER), and question answering.

Among the tasks mentioned, NER is considered essential in scientific research. This is due to the fact that extracting relevant information from a vast and rapidly growing corpus of scientific literature can be both difficult and challenging. This is especially true in fields like medicine, where countless articles and discoveries are published daily. To access the most recent results efficiently, scientists can leverage models like BERT. Using BERT for NER not only makes this process faster and more convenient to search through millions of articles, but also enables researchers to identify key concepts and relationships that would have been computationally intensive and difficult to find manually.

However, despite their impressive capabilities, training and developing these models remains time-, memory-, and energy-intensive [4]–[6]. As these models continue to achieve better performance and become more sophisticated, they also tend to grow over-parameterized and larger. This raises the question of how to reduce their size while maintaining the same performance level as the original model or at least delivering results at an acceptable rate.

For the project in question, the company AstraZeneca has a BERT model trained on two datasets generated using an index of MEDLINE in i2e [7]. The current model has a high runtime and significant memory usage, making it a priority for AstraZeneca to reduce these costs. To extract oncology efficacy endpoints from scientific literature, the company utilizes multiple fine-tuned pipelines. These pipelines consist of various BERT-based models, including BioBERT [8], PubMedBERT [9], DistilBERT [10], and the foundational BERT architecture. [11]. However, training large deep-learning models is both computationally expensive and time-consuming. Given that hundreds of thousands of new articles are published monthly, using these models to extract the latest information becomes increasingly slow and resource-intensive. To address this challenge, potential solutions are being explored to reduce the models' runtime and computational costs.

A significant amount of research has focused on minimizing the size of BERT models while maintaining their performance. For instance, BinaryBERT [12], Q8BERT [13], I-BERT [14], and BI-BERT [15] use quantization techniques to minimize model size while maintaining reasonable performance metrics. Other approaches, including DistilBERT [10] and TinyBERT [16], reduce models size by transferring knowledge from larger models to smaller ones. Additionally, pruning techniques, as applied in O-BERT [17], reduce the number of parameters to minimize model size while limiting the impact on accuracy.

Although these methods are efficient and deliver excellent results, some require specialized GPUs that may not be readily accessible, while others demand significant time and resources, such as re-training BERT models. For this project, the focus is on methods that can be applied in post-training and are compatible with a wide range of models. Among these, post training quantization and pruning stand out as particularly promising approaches due to their ability to reduce model size and computational demands without significantly compromising performance.

Building on this foundation, the project aims to investigate the following research questions:

- To what extent is it feasible to optimize the inference time of these BERT-based models, and what methodologies can be implemented to achieve this optimization without compromising the critical performance benchmarks essential for accurate oncology efficacy endpoint extraction?

- To what extent can the memory footprint of these BERT-based models be minimized without compromising their operational efficiency and scalability in real-world oncology endpoint extraction tasks?

- What strategies can be employed to effectively reduce the architectural size of these deep learning models while preserving their competitive performance metrics, particularly in terms of accuracy and F1-score?

1.1 Limitations

The main constraint encountered during this research stemmed from the necessity of optimizing the models on CPU nodes. Since extensive research has been conducted on quantization techniques to accelerate inference time and memory footprint on GPU nodes, optimizing these BERT-based models while preserving performance metrics on CPUs was challenging. Additionally, Integer-Only BERT (I-BERT) was implemented to quantize the model with 8-bit integers. Although it worked on CPU and GPU, to fully leverage the promised inference accelerations of I-BERT, a special type of hardware was needed. Consequently, the decision was made to move on to other methods that work effectively on CPUs, based on the company's requirements.

Furthermore, the Hugging Face website was internally filtered by AstraZeneca. Given that, the training and inference process of the existing models were based on Hugging Face [18], this made the process of training the models on internal Scientific Computing Platform (SCP) CPU and GPU nodes more difficult, and the training process was subsequently migrated to Azure Databricks [19].

2

Theory

In the following section, the theoretical foundations is introduced to support a clearer understanding of the models and techniques used throughout the study.

2.1 Transformer

Transformers have become dominant models in almost all natural language processing tasks. Before the introduction of the article "Attention Is All You Need" [20], recurrent neural networks (RNNs) and long short-term memory networks (LSTMs) faced challenges with sequential processing, which made the training process slow and inefficient for long sequences. Additionally, their limited ability to capture long-term dependencies due to the vanishing gradient problem was a significant obstacle.

The improvements made to encoder-decoder RNNs for sequence-to-sequence applications, such as machine translation [21], can be considered the foundation behind the Transformer model. In RNN models, the entire input sequence is processed sequentially and compressed into a single fixed-length vector, typically the final hidden state of the encoder, which is then fed into the decoder to generate the output sequence. The key innovation of the attention mechanism was that, instead of relying on the same input vector at each decoding step, the decoder could focus on specific parts of the input sequence as needed. At each decoding step, the decoder receives a vector consisting of a weighted sum of the input representations, which allows it to dynamically emphasize the most relevant information [22]. In this weighted sum, each weight reflects the importance of a specific input token. Crucially, these weights must be distinct and well-defined so that the model can effectively learn and capture the nuances within its parameters.

In 2017, Vaswani et al. introduced the state-of-the-art Transformer architecture, featuring an attention mechanism that could not only attend to different parts of a sequence by assigning different weights to them but also read and process all words simultaneously. This parallel processing makes Transformers faster and more efficient, especially when handling large datasets and long sequences. An illustration of transformer model architecture can be seen in Figure 2.1. It indicates the use of multi-head, self-attention, and feed-forward layers in both the encoder and decoder structures, which enables efficient modeling of sequence relationships.

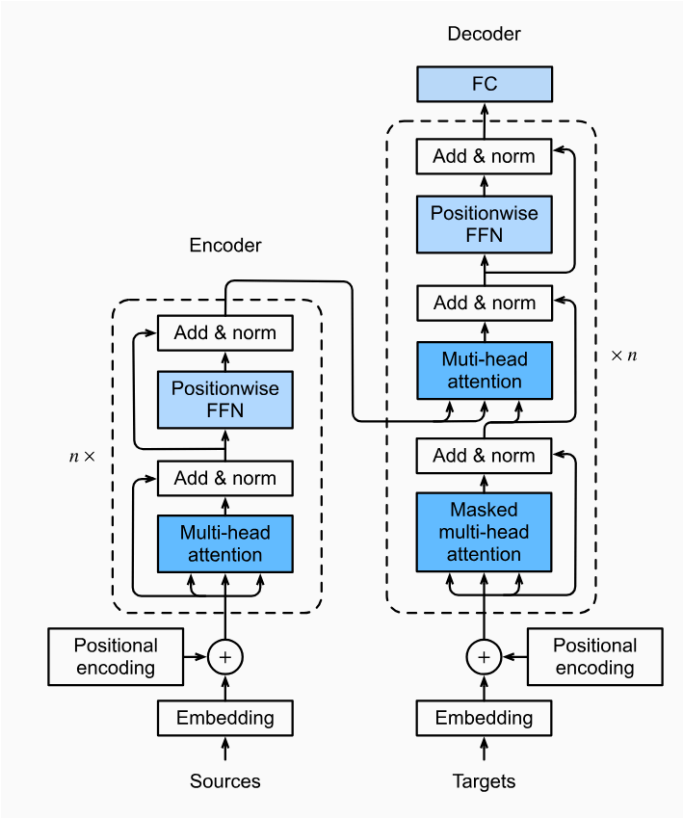

Figure 2.1: The Transformer architecture, including the encoder and decoder components. Reproduced from Dive into Deep Learning [23], under the Apache 2.0 license

2.2 Attention

Attention is the fundamental mechanism in transformer models like BERT that allows the model to capture relationships between different parts of the input sequence, regardless of how far apart they are. The core idea behind attention is to assign a weight to each input, which allows the model to compute a weighted combination that indicates where the focus should be and how much attention each part requires. In other words, it shows the relation between one token and another token in an input, and which word or words are more important compared to others in a specific sequence. Formally, the attention mechanism can be described as:

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{2.1}$$

Where Q (query), K (key), and V (Value) are projections of the input embeddings, and d_k is the dimensionality of the key vectors. The Softmax function ensures that the attention weights are normalized and can be interpreted as probabilities.

This mechanism allows the model to capture the meaning of the entire sentence more effectively, as not all words contribute equally to the overall meaning. Certain

words may carry more information depending on the context, and attention enables the model to recognize and leverage this information. This mechanism is one of the key reasons why transformer models achieve better performance on tasks such as translation, text classification, and question answering. In practice, transformers do not have just a single attention computation component; rather, they contain a multi-head attention mechanism. This mechanism allows the model to learn different types of relationships simultaneously between different parts of the input by projecting the input into lower-dimensional spaces, applying attention separately in each space, and then finally combining the results.

The computation for multi-head attention can be formally expressed as follows:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{2.2}$$

$$\text{where} \quad \text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{2.3}$$

[1]

2.3 BERT

BERT (Bidirectional Encoder Representations from Transformers) is a transformer-based machine learning model for natural language processing (NLP). It was introduced in 2019 by researchers at Google and is one of the first models to deeply understand the full context of a text from both directions, left to right and right to left, which is why it is called "bidirectional". This means that BERT reads and understands the entire sentence at once. Despite the fact that transformer-based models typically consist of both an encoder and a decoder in their architectures, BERT is based only on the encoder part of the transformer architecture. Each encoder layer contains multi-head self-attention, a feedforward neural network (a fully connected layer applied after attention), layer normalization, residual connections, and dropout to help with training.

BERT was pre-trained on two major tasks before being fine-tuned: masked language modeling (MLM) and next sentence prediction (NSP). In the MLM task, 15% of the words in a sentence are randomly masked, and the model is trained to predict the missing (masked) words. MLM requires BERT to understand the complete content of a sentence, both before and after the masked word. In the NSP task, BERT is trained to predict whether two given sentences are consecutive in the original text. This helps BERT learn the relationships between sentences, not just the relationships between words inside a sentence.

2.3.1 DistilBERT

In deep learning, there are different compression techniques such as pruning, knowledge distillation, and quantization that play important roles in reducing the computational costs and accelerating models' inference time. Among these techniques,

Knowledge distillation or model distillation is the process of transferring knowledge from a large pre-trained model (teacher) to a smaller model (student) without loss of performance and validity. The distilled models can be effectively deployed on resource-constrained hardware since they are faster and less expensive to evaluate. [24].

Knowledge distillation (KD), is typically applied on large deep neural networks with massive numbers of parameters and layers. This process makes it useful especially in the case of Natural Language Processing (NLP) and the domain of Large Model Languages (LLM), which often involve models with millions of parameters. The primary objective of KD is to train a more compact student model to mimic the predictions made by a more complex teacher model.

In 2020, Sanh et al.[10] proposed a distilled version of BERT with almost the same architecture; the main difference between DistilBERT and standard BERT is in the Transformer encoder layers and the number of parameters. DistilBERT consists of 6 Transformer encoder layers and 66 million parameters, which is 40% smaller than standard BERT. Although BERT is pre-trained using two tasks, Masked Language Modeling (MLM) and Next Sentence Prediction (NSP), in DistilBERT, the NSP objective was removed completely, which simplifies the training and reduces overall model complexity.

Furthermore, during the initial training of DistilBERT, cross-entropy, MLM loss, and cosine embedding loss were utilized for the DistilBERT (student model) to emulate the behavior of the BERT (teacher model) while preserving as much of its performance as possible. Cross-entropy loss, which was used in the distillation process, compares the students predicted probabilities to those of the teacher model. In addition, cosine embedding loss was incorporated into the cross-entropy loss to align the directions of the student and teacher hidden state vectors. Finally, MLM loss was added to the last two losses so that the student model learns to predict masked tokens.

Despite this reduction, DistilBERT achieves 97% of BERT's performance on many NLP tasks, which makes it an efficient alternative for real-world applications.

2.3.2 BioBERT

BioBERT (Bidirectional Encoder Representations from Transformers for Biomedical Text Mining) is a variant of BERT, published in 2019 by Lee et al. from Korea University and Clova AI Research, NAVER Corporation. BioBERT shares the same architecture as the BERT model, consisting of 12-layer transformers with 786 hidden units and 12 attention heads, but the main difference is the training corpus of BioBERT. While BERT was pre-trained on general domain corpora (BookCorpus and Wikipedia), BioBERT was initialized with BERT's weights and then further pretrained on approximately 18 billion words of biomedical text from PubMed abstracts and PMC full-text articles using the same masked language modeling and next sentence prediction objectives. This additional domain-specific training enables

BioBERT to have a deeper contextual understanding of biomedical terminology and language patterns that the general model may struggle with.

2.4 Pruning

Pruning is a fundamental method for compressing neural network models by eliminating non-essential parameters while preserving the performance of the model at equivalent level of the original state. Language models have reached sizes with billions of parameters and pruning methods have been used as an important method alongside other compression techniques to accelerate inference time on resource-constrained environments. There are two critical questions regarding pruning approaches: (1) what to prune, and (2) how to prune. In the following sections, we answer these important questions.

2.4.1 Pruning Units

Pruning units refer to elements that are selected for removal during the pruning process, including elements such as weights, neurons, layers, attention heads, etc. Based on pruning units, pruning methods can be categorized as structured and unstructured pruning [25].

2.4.1.1 Unstructured Pruning

Unstructured pruning or weight-wise pruning is the finest-grained case. In general, individual weights are identified based on importance criteria and zeroed out, but in practice, for small or medium models like BERT, unstructured pruning usually does not directly set the weights to 0, but their corresponding masks M are set to 0. The mathematical representation of unstructured pruning can be written as the following constrained optimization problem:

$$\min_{w,M} L(w \odot M; D) = \min_{w,M} \frac{1}{N} \sum_{i=1}^{N} \ell(w \odot M; (x_i, y_i)) \tag{2.4}$$

S.t. $\|M\|_0 \leq k$ where the binary mask M is multiplied element-wise into the model by $\odot$, $w = \{w_1, w_2, ..., w_M\}$ represents neural network weights, D is a dataset composed of N input x_i and output y_i pairs, and k is the target non-sparsity ratio. However, one shortcoming of the unstructured pruning method is that it results in irregular sparse patterns that limit the computational efficiency benefits of standard hardware architectures that are not designed for sparse computation.

2.4.1.2 Structured Pruning

This theoretical approach removes entire filters, such as attention heads, neurons in feed-forward networks (FFNs), or entire layers. While this method typically achieves lower sparsity ratios than unstructured approaches, it does not require the support

of special hardware and software and can directly speed up networks and reduce the size of the neural networks.

2.4.2 Pruning Metrics

These metrics can be categorized into three principal theoretical frameworks:

2.4.2.1 Magnitude-based Metrics

Magnitude-based metrics employ the magnitude (absolute values) of weights or activations to determine importance. The theoretical principle underlying this approach is that weights with smaller magnitudes contribute proportionally less to model outputs.

2.4.2.2 Loss-based Metrics

Loss-based metrics constitute a more theoretically sophisticated approach that assesses a pruning unit's importance based on its impact on the loss function. The central theoretical principle is that removing parameters that minimally affect the loss function will preserve model performance. Loss-based metrics are further subdivided into two theoretical frameworks:

2.4.2.3 Second-order Methods

These methods incorporate the Hessian matrix (second-order derivatives) to more accurately approximate the change in loss when parameters are removed. The theoretical expression for importance in second-order methods is:

$$I = \frac{1}{2}(w - w^*)^T H_L(w^*)(w - w^*) \tag{2.5}$$

Where $H_L(w^*)$ is the Hessian matrix. This formulation captures parameter interactions and provides a more accurate theoretical estimate of importance, though at a higher computational cost.

The second-order methods derive their theoretical foundation from a Taylor expansion of the loss function. For a well-trained model with weights w^*, the change in loss when moving to a pruned state w can be approximated as:

$$L(w) - L(w^*) \approx (w - w^*)^T \nabla L(w^*) + \frac{1}{2}(w - w^*)^T H_L(w^*)(w - w^*) \tag{2.6}$$

Since $\nabla L(w^*) \approx 0$ for a well-trained model, the second-order term dominates the approximation. This theoretical insight suggests that the importance of a parameter can be effectively estimated using the second-order information contained in the Hessian matrix.

2.5 Quantization

Quantization is a technique that maps values from a large (often continuous) set to a smaller, finite set to reduce the memory and computational costs of large language models during inference. It transforms high-precision floating-point values (typically 32-bit) to lower-precision formats (such as 8-bit or 4-bit integers), significantly reducing model size and improving inference speed, especially on hardware optimized for low-bit operations. For example, quantizing weights from 32-bit float to 4-bit integer can compress the model to approximately 1/8 of its original size.

2.5.1 Uniform and Non-uniform Quantization

Under uniform symmetric quantization, a real number x is mapped to an integer value $q \in [-2^{b-1}, 2^{b-1} - 1]$, where b specifies the quantization bit precision:

$$q = Q(x, b, S) = \text{Int}(\frac{\text{clip}(x, -\alpha, \alpha)}{S}) \tag{2.7}$$

Where Q is the quantization operator, Int is the integer map, clip is the truncation function, α is the clipping parameter for outlier control, and S is the scaling factor defined as $\alpha/(2^{b-1} - 1)$. The dequantization process is:

$$\tilde{x} = DQ(q, S) = Sq \approx x \tag{2.8}$$

Non-uniform quantization does not use equally spaced intervals, which allows a better representation of neural network weight distributions. The general formula is:

$$Q(r) = Q_i, \text{ if } r \in [\Delta_i, \Delta_{i+1}) \tag{2.9}$$

where Q_i represents quantization levels and Δ_i defines the intervals. While non-uniform methods may better capture parameter distributions, they often require lookup tables that create deployment overhead on hardware.

2.5.1.1 Post-Training Quantization (PTQ)

Quantization techniques are broadly categorized into Post-Training Quantization (PTQ) and Quantization-Aware Training (QAT) approaches. PTQ determines quantization parameters without re-training, making it fast and computationally efficient, but potentially less accurate for low-precision settings. The model can be directly used for inference after quantization and it's generally easier to implement.

2.5.1.2 Quantization-Aware Training (QAT)

Conversely, QAT methods involve re-training, and the model is trained with simulated quantization effects, which helps recover the error introduced by quantization. The following text describes different quantization techniques for the self-attention layer in the Transformer architecture. The first approach represents a common quantization method where the parameters are quantized and stored in 8-bit integer

format. Since the operations are performed in floating-point numbers, the outputs are dequantized before each operation. The second approach is a quantization method in which the model is partially quantized by integer arithmetic. Before each Softmax operation, the inputs are dequantized for the operation to be performed in floating-point numbers, and then after Softmax, the outputs are quantized back to feed to the matrix multiplication operation. The third approach corresponds to I-BERT quantization, which will be discussed further in the methods chapter. In this case, the model is fully quantized, and there is no dequantization or floating-point arithmetic involved throughout the process.

Additionally, QAT addresses the perturbation caused by quantization by either simulating the process in re-training or using additional parameters to fine-tune the quantized model. This perturbation helps the model converge to a point with better post-quantization performance. While QAT typically achieves higher accuracy, it requires significant computational resources for re-training, which may not always be possible for models with billions of parameters.

2.6 Optimization

Apart from quantization, various optimization techniques can be employed to enhance the performance of transformer models such as BERT. These optimizations aim to reduce memory usage and computational complexity while preserving model accuracy. By implementing these techniques, inference time will be speed up significantly and these powerful models will be more efficient for real-world applications. Additionally, a core part of transformer architecture models like BERT are attentions which are one of the most computationally expensive parts of the model. By optimizing attention, the model can be improved effectively. Spare attentions, attention head pruning and optimized matrix multiplication are the attention optimization techniques that enable models like BERT to handle longer sequences better and reduce their memory requirements.

2.6.1 Operator Fusion

Operator fusion is a key optimization technique that combines multiple operations into a single, more efficient operation without changing the output. This process reduces memory access and computational overhead, which leads to performance improvement [26]. The fusion of operations minimizes data movement between memory and processing units, which is often a bottleneck in model execution. This fusion reduces the number of memory read/write operations and kernel launches, which results in lower latency and better utilization of hardware resources. However, it is important to note that the implementation of operator fusion must carefully consider numerical stability and potential interactions with other optimizations [27], [28] .

2.6.2 Node Elimination

Another optimization technique is node elimination. This process involves identifying and removing unnecessary or redundant nodes from the computational graph of the model that do not contribute to the final output. By simplifying the graph structure, both the memory footprint and the computational requirements of the model can be reduced [29].

In BERT and similar transformer models, common targets for node elimination include identity operations that don't modify the data, redundant reshape or transpose operations, and unused output from multi-output operations that potentially can reduce the model size [30].

2.7 Performance Metrics

To evaluate how well a machine learning model performs on a given task, performance metrics are used. In the case of transformer-based architectures (such as BERT, DistilBERT, or BioBERT), the choice of metrics depends on the specific task such as classification, sequence labeling, question answering, or text generation.

For sequence labeling tasks (e.g., Named Entity Recognition or Part-of-Speech tagging), where transformers assign a label to each token, common evaluation metrics include:

- Token-level accuracy, which measures the overall correctness of token predictions.
- Entity-level precision, recall, and F1-score, which evaluate the correctness of predicted entity spans.
- Span-level F1-score, which considers entire entity spans rather than individual token matches.

The standard evaluation metrics for classification tasks are commonly defined as follows: true positives (TP), which are correctly identified positive cases; false positives (FP), which are incorrectly labeled as positive; true negatives (TN), which represent correctly identified negative cases; and false negatives (FN), which occur when negative cases are mistakenly classified as positive.

2.7.1 Accuracy

Accuracy is the ratio of the correct predictions to total predictions and measures the overall correctness of our model. It can be defined as:

$$\text{Accuracy} = \frac{\text{number of correct predictions}}{\text{number of total predictions}} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{2.10}$$

2.7.2 Recall

It refers to the ability of the model to find all instances of the positive cases.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.11}$$

2.7.3 Precision

Precision refers to the fraction of positive predictions that were accurately predicted by the model. It determines how often our predicted positive cases are actually positive.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.12}$$

2.7.4 F1 scores

The F1 Score is the harmonic mean of precision and recall. It provides an overall trade-off metric between precision and recall, and it is especially helpful in situations where there is a need to manage an imbalanced distribution between the outputs.

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2 \times \text{TP}}{2 \times \text{TP} + \text{FP} + \text{FN}} \tag{2.13}$$

3

Methods

Building upon the theoretical foundation established in the previous chapter, this section presents the practical approaches and models utilized in our research, conducted as part of the AstraZeneca NER (Named Entity Recognition) project. The focus here is on the implementation and optimization of transformer-based models for biomedical text classification.

In this project, three BERT-based architectures have been utilized, including BERT-base, BioBERT, and DistilBERT. All of these models have been pre-trained on large text corpora, with BioBERT and PubMedBERT specifically adapted for biomedical NLP tasks. The objective was to optimize these existing models in two primary ways:

- Quantization, to accelerate inference time and reduce computational costs.
- Pruning, to decrease model size and memory footprint.

Both techniques aim to make the models more efficient for deployment, particularly in environments with limited resources, without compromising the predictive performance metrics, which were discussed in the previous chapter.

3.1 Data

The dataset used in this project is derived from the work of Gendrin-Brokmann et al. [11], who developed it to train BERT-based models for extracting efficacy endpoints from biomedical literature, with a particular focus on oncology trials. The corpus comprises annotated sentences sampled from the MEDLINE database [31], with initial sentence extraction and preprocessing performed using the I2E [32] natural language query platform developed by Linguamatics. I2E is designed to extract relevant facts, relationships, and entities from large collections of unstructured and semi-structured text data, especially in domains such as life sciences and healthcare. Also, it uses NLP capabilities, which often combine linguistic rules with statistical methods and machine learning, to read text and pull out specific pieces of information (like efficacy endpoints, diseases, genes, compounds, etc.).

For the purpose of our analysis, the dataset utilized contains entities annotated as clinically relevant efficacy endpoints such as survival metrics (e.g., overall survival, progression-free survival), response rates, and associated statistics such as confidence

intervals, hazard ratios, and p-values. These entities are represented in various textual formats, such as percentage-based or duration-based expressions, and were manually labeled to support fine-grained entity recognition. The annotation process was done by utilizing the Label Studio annotation tool [33], and involved two domain experts in clinical information science. Each sentence was independently labeled by both annotators, and any disagreements were subsequently resolved through review, which ensures high-quality and consistent labeling.

According to the summary statistics in Table 3.1, the training set comprises 5,392 annotated sentences, while the test set contains 983 sentences. The dataset is specifically tailored for token classification tasks and is well-suited for evaluating Named Entity Recognition (NER) models in the biomedical domain. In this work, this dataset is reused to train our models and systematically evaluate the impact of different fine-tuning and model compression techniques, such as quantization and pruning, on the task of efficacy endpoint extraction.

3.2 Pruning

To make the models smaller and faster during inference, the decision was to perform pruning in addition to quantization. The goal was to remove the redundant parts of the model while maintaining baseline NER performance. We opted for structured pruning, which involves removing masked attention and FFN neurons. Although higher sparsity could have been achieved with unstructured pruning and minimal accuracy drop, we chose structured pruning because unstructured sparse patterns don't necessarily lead to inference speedup on standard hardware [25]. Since structured pruning produces a smaller and denser model that can run on common hardware without the need for special libraries, three levels of structured sparsity (20%, 25%, and 30%) were applied to both Standard BERT and BioBERT in this project to reduce their memory footprint and accelerate inference time.

Our structured pruning process was inspired by the method introduced in A Fast Post-Training Pruning Framework for Transformers [34] article. In their post-training method, the entire dataset is not used for retraining; instead, a subset of the training dataset is sufficient. A sample of the dataset, along with a few algorithmic steps, is used to identify and remove the unimportant parts of the model. Their approach comprises three stages: Fisher Information mask search, mask rearrangement, and mask tuning. The first two stages of this approach were used, and the third stage was modified to suit our practical considerations. The pruning procedure can be summarized as follows:

Table 3.1: Total number of sentences in the training and test datasets, as well as the number of endpoints mentioned (words in entities) across datasets from two different annotators. Different formats of endpoints, such as durations, percentages, and confidence intervals, are treated as distinct entities. [11]

Endpoint Type	**Training Dataset (words in entities)**	**Test Dataset (words in entities)**
DFS	2907	56
DFS_CIH	1282	0
DFS_CIL	1176	0
DFS_percent	6847	1654
DFS_percent_CIH	1980	73
DFS_percent_CIL	1770	67
DoR	1931	23
DoR_CIH	677	6
DoR_CIL	737	6
ORR	3320	509
ORR_CIH	541	104
ORR_CIL	503	95
OS	5051	1092
OS_CIH	1418	198
OS_CIL	1372	180
OS_percent	5805	5489
OS_percent_CIH	2063	379
OS_percent_CIL	1880	343
PFS	5139	536
PFS_CIH	1332	143
PFS_CIL	1256	121
PFS_percent	1758	353
PFS_percent_CIH	959	24
PFS_percent_CIL	981	16
time_point	5925	2591
Total sentences: Training = 5392, Test = 983		

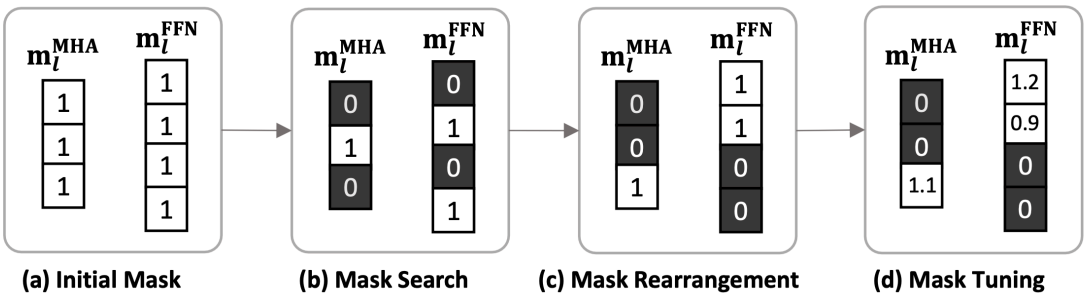

Figure 3.1: Overview of Post-Training Pruning Framework. (a) The mask variables are applied as 1. Then they undergo the three-stage process of (b) mask search, (c) rearrangement and (d) rescale. [34]

3.2.1 Fisher-based Mask Search

At first, the importance of each attention layer and hidden filter was evaluated in each feed-forward layer using a Fisher Information Matrix score. In this step, we measure how much the removal of each of those components' weights will impact the model's loss. This mask search was performed by computing the diagonal approximation of the Fisher Information Matrix of the model parameters by making a forward and backward pass on a small sample of the training dataset. Components with the lowest Fisher-based importance were marked as pruning candidates. Then a binary mask was applied over all heads and neurons such that components that were set to be pruned got a mask value of 0 while the rest remained assigned with mask 1.

3.2.2 Mask Rearrangement

After applying the binary mask, the pruning selection was refined through a mask rearrangement step. In the first stage the Fisher Information Matrix was simplified by diagonal approximation. However, that stage alone will not result in optimal masking pattern, hence the second step was to account for intra-layer interactions and distribution of pruned units across each different layer. Removing too many components from a single layer might harm model performance, even if each was low-importance in isolation. Thus, we adjusted the pruning mask to better balance the masking pattern in each layer.

Basically, this stage improves upon the initial Fisher mask by considering inter-layer trade-offs to maintain overall performance. We iterated this two-stage process until the mask pattern satisfied the global FLOPs constraint while keeping the loss minimal. At this point, the structured mask was finalized, and it was defined which attention heads and FFN neurons to remove and prune those components from the model architecture.

3.2.3 Mask Tuning

In the past two stages, the mask values were binary (1 or 0); in this stage, the nonzero values from the previous stage are tuned to real values to recover some accuracy loss. In the earlier stages, the model was pruned to satisfy the latency/Flop constraint by applying the binary mask. This binary mask selection and intra-layer rearrangement produced a pruned transformer. In this pruned model, this binary mask will cause a degradation in accuracy. The mask tuning stage is designed to recover this lost accuracy by adjusting the remaining components' masks without changing them. By allowing those remaining masks to deviate from 1, the model can compensate for removed components.

3.3 Model Quantization and Optimization

3.3.1 Optimization

To reduce the model size and complexity and to improve performance, the model was optimized with ONNX Runtime [35] Transformer optimization tool. This step involves a series of graph-level transformations specifically designed for BERT-like architectures. The goal is to streamline the models computational graph, reduce inference time, and prepare it for further compression through quantization by eliminating inefficiencies. The optimizations applied include operator fusion, removal of redundant operations, constant folding, and reordering and simplification of computation graphs. These techniques help consolidate operations, minimize runtime overhead, and make the model more efficient for quantization.

3.3.2 Post Training Quantization

In addition to pruning and optimization, another strategy applied to the existing model was ONNX Runtime dynamic quantization [36], which aimed to reduce the model size and improve inference efficiency, particularly on CPU-based systems. Specifically, dynamic quantization was used. The weights of the model, originally stored in 32-bit floating-point format, were quantized offline to 8-bit integers. Each weight tensor was assigned a scale and zero-point to map the 32-bit floating-point range into 8-bit integer, to reduce memory usage and speed up computations.

During inference, the activations (the outputs from previous layers) remain in 32-bit floating-point but are dynamically quantized right before they are used in quantized operations. This dynamic quantization process enables the system to convert the activations into 8-bit integers, which is more efficient for computation, while still retaining enough precision to ensure accurate results. After the quantized operations are performed, the output is dequantized back into 32-bit floating-point for further processing, maintaining the integrity of the models computation.

In practice, ONNX Runtime applies quantization selectively to certain linear opera-

tions, such as fully connected layers (GEMM), matrix multiplications, and element-wise additions. This targeted quantization improves the inference performance by accelerating the execution of these key operations without the need for model retraining or additional calibration data. By quantizing only specific operations, ONNX Runtime strikes a balance between speed optimization and model accuracy.

The quantization of weights can be represented as:

$$\text{quantized}_{\text{val}} = \text{round}\left(\frac{\text{float}_{\text{val}}}{\text{scale}}\right) + \text{zero}_{\text{point}} \tag{3.1}$$

where:

- $quantized_{val}$ is the resulting quantized value.
- $float_{val}$ is the original floating-point value.
- scale is the scaling factor that maps the floating-point range to the integer range.
- $Zero_{point}$ is the scaling factor that maps the floating-point range to the integer range.

This equation is used to convert a floating-point value ($float_{val}$) to an integer representation ($quantized_{val}$) by scaling and shifting, ensuring that the quantized values are as close as possible to the original floating-point values while using the reduced bit-width for storage and computation.

3.3.3 Quantization Aware Training: Integer Only BERT

Furthermore, I-BERT was chosen as one of the quantization methods for this project. I-BERT focuses on uniform symmetric quantization and employs static quantization, where the scaling factors remain fixed during inference. To enable fully integer-only computation, not just for the weights but also for the entire model pipeline, Sehoon Kim et al. [14] introduced different non-linear activation functions such as GELU and softmax by using polynomial approximation functions. Since polynomials involve only addition and multiplication, these operations can be efficiently executed using integer arithmetic alone and eliminate the need for floating-point operations during inference.

Additionally, matrix multiplications and embeddings in I-BERT are carried out using 8-bit integer precision, while the non-linear functionslike GELU, softmax, and Layer Normalization are computed using 32-bit integer precision. The reason for using 32-bit integers in these cases is that it avoids the need for dequantization and introduces minimal computational overhead. Although we apply this approach to RoBERTa-Base, the same technique can be extended to other transformer-based models, as long as they rely on similar non-linear operations.
The following equation approximates GELU activation, which first provides a polynomial approximation to the Gaussian error function $L(x)$, and then i-GELU is

defined. Here $a = -0.2888$, $b = -1.769$, and sgn(x) is the sign function.

$$L(x) = \text{sgn}(x) \cdot \left(\frac{a \cdot (\text{clip}(|x|, \max = -b) + b)}{2} + 1 \right) \tag{3.2}$$

$$\text{i-GELU}(x) := x \cdot \frac{1}{2} \left(1 + L \left(\frac{x}{\sqrt{2}} \right) \right) \tag{3.3}$$

For the softmax function, I-BERT uses a second-order polynomial to approximate the exponential function over the interval $(-\ln 2,\ 0]$. The following approximation is derived by minimizing the distance between the exponential function and the polynomial over this domain.

$$L(p) = 0.3585 \cdot (p + 1.353)^2 + 0.344 \approx \exp(p) \tag{3.4}$$

This leads to the integer-only exponential approximation:

$$\text{i-exp}(\tilde{x}) := \frac{L(p)}{2^z} \tag{3.5}$$

$$\text{Where} \quad z = \left\lfloor \frac{-x}{\ln 2} \right\rfloor, \quad p = \tilde{x} + z \cdot \ln 2 \tag{3.6}$$

This method avoids computing actual exponentials by relying on polynomial approximations and bit-shifting.

Furthermore, for Layer Normalization, I-BERT adopts the standard normalization process, but reformulates it to be compatible with integer arithmetic. The procedure is as follows:

$$\tilde{x} = \frac{x - \mu}{\sigma} \tag{3.7}$$

Where the mean μ and standard deviation σ are computed as:

$$\mu = \frac{1}{C} \sum_{i=1}^{C} x_i \tag{3.8}$$

$$\sigma = \sqrt{\frac{1}{C} \sum_{i=1}^{C} (x_i - \mu)^2} \tag{3.9}$$

This expression is then implemented with fixed-point approximations to make the operation integer-friendly while maintaining numerical stability.

Although I-BERT was implemented within this project, and its performance metrics are represented in the Results chapter, a decision was made not to proceed with I-BERT in the final approach. This was primarily because it necessitates specialized hardware for optimal 8-bit precision. Moreover, in comparison to other techniques that were utilized in this work, I-BERT showed notable accuracy and F1 drops and higher inference times.

3.4 Pruning and Quantization

To leverage both the benefits of quantization and pruning to compress the model and further accelerate inference time, dynamic quantization was implemented on the pruned model. In this regard, the model was first pruned using the Fisher Information Matrix Pruning method on the provided GPUs, and subsequently, it was quantized by ONNX Runtime's dynamic quantization library on CPU nodes. To quantize the pruned model, the same pruning amounts (20%, 25%, and 30%) were applied to neurons and attention heads in the model.

3.5 Training

To train the original models, BERT-based, DistilBERT, and BioBERT on the training dataset, the data was tokenized using the Hugging Face AutoTokenizer. In the tokenization process, sentences are divided into individual tokens, which are words or sub-word units, and for each token, one label is assigned. The models were fine-tuned with code adapted from the Hugging Face Transformers library. Additionally, during training, the model learned to classify each token according to a predefined set of NER tags extracted from the training data. All artifacts, including model checkpoints, configuration files, and training logs, were systematically organized in timestamped output directories for reproducibility and post-training analysis and quantization.

3.5.1 Hardware

The models were trained and evaluated on the Azure Databricks platform, where access was provided to 36 CPU cores, 440 GB of memory, and one NVIDIA A10 GPU. The GPU resources were particularly essential during the model pruning process in this project, where compute-intensive operations like the Fisher Information Matrix pruning approach required parallel processing capabilities. This hardware configuration provided by AstraZeneca was sufficient computational power for both the initial fine-tuning phases and the subsequent model optimization and quantization experiments.

3.6 Testing And Evaluation

To evaluate the quantization, pruning, and pruning-quantization techniques developed in this work and to compare their performances with original BERT variation models, they need to be sufficiently tested to ensure their robustness and functionality. Since for the dynamically quantized model only weights are quantized, and it comes in one configuration, it is not possible to test the model with different parameters. However, for both pruning alone and combined pruning and quantization, the models and implementations can be tested by varying the amount of pruning. This allows us to determine precisely how much the model can be pruned without sacrificing performance metrics and compromising its functionality.

In the inference phase of the project, the structural pruning was evaluated at three fixed sparsity levels (20%, 25%, and 30%) using the Fisher Information Matrix approach and standard PyTorch inference [37] with Hugging Face's tokenization pipeline. For both quantization and pruning-quantization experiments, the inference was exported to ONNX format using `torch.onnx.export`. Moreover, in pruning-quantization experiments, same as pruning experiments, the models were evaluated at three fixed sparsity levels (20%, 25%, and 30%). In all experiments implemented in this project, including pruning, quantization, and pruning-quantization, performance metrics such as F1-score, accuracy, precision, and recall were calculated using the `seqeval` library [38]. Additionally, benchmarks were conducted to measure inference latency and throughput with a batch size of 16.

4

Results

Building upon the methodologies detailed in the last chapter, this chapter presents the comprehensive results derived from our quantization, pruning, and combined quantization and pruning experiments.

4.1 Structural Pruning

This section quantifies the pruning effect on the BERT and BioBERT models within the NER pipeline. A three-stage structural-pruning procedure was applied, comprising Fisher-based mask search, mask rearrangement, and mask tuning. In Figure 4.1 and Figure 4.2, performance metrics such as accuracy, F1 score, precision, and recall were obtained for the standard models and for sparse models in which 20%, 25%, and 30% of attention heads and FFN neurons had been removed. The global performance of standard BERT remained flat up to 25% structured sparsity; accuracy slipped by 0.2% (0.973 to 0.971) and F1 score by only 1% (0.928 to 0.918). However, at 30% sparsity, recall collapsed by 16.5% (0.906 to 0.779), and the F1 score fell to 0.854. Paradoxically, precision increased with greater sparsity and peaked at 0.944 at 30% sparsity.

In contrast, BioBERT, which was pretrained in the medical domain, showed more resilient. Even at 30% sparsity, it retains 0.975 accuracy and 0.936 F1 score, only 0.7% and 1.8% below baseline, and its recall remains 12.5% higher than BERT at the same sparsity level. As in standard BERT, precision rose with higher sparsity.

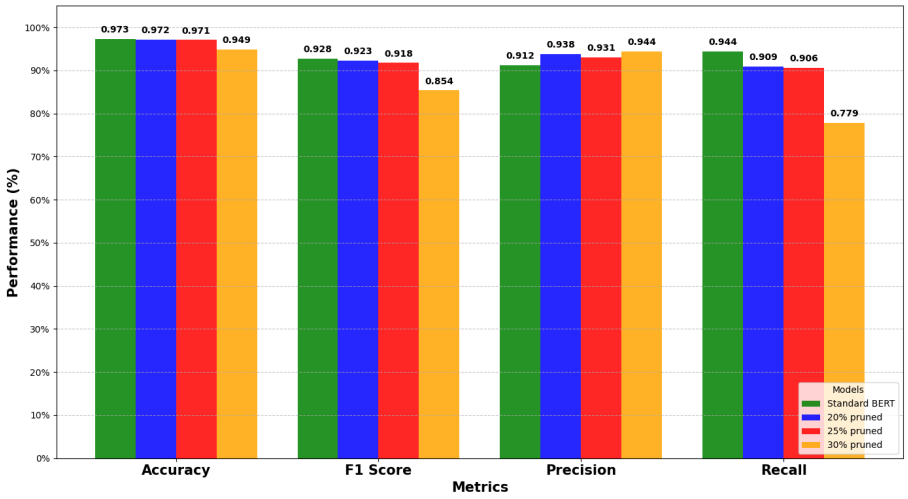

Figure 4.1: Global metrics comparison of BERT-base (green), 20% pruned (blue), 25% pruned (red), and 30% pruned (orange).

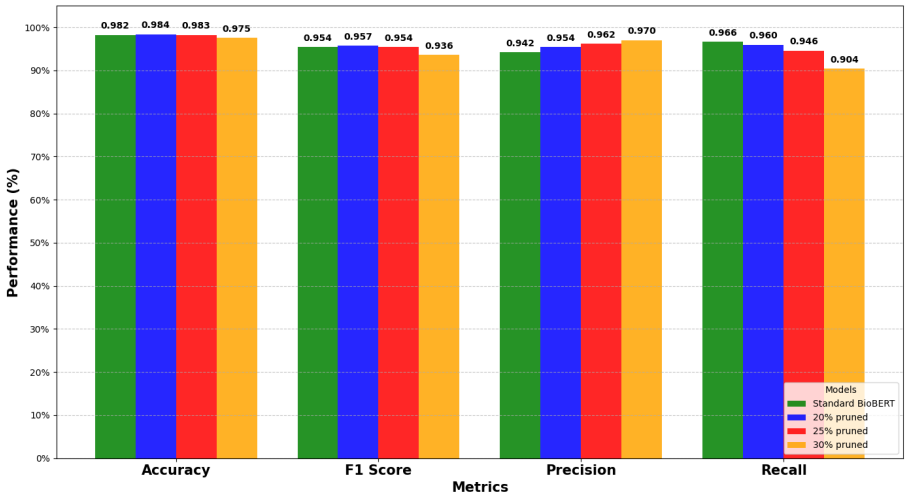

Figure 4.2: Global metrics comparison of BioBERT (green), 20% pruned (blue), 25% pruned (red), and 30% pruned (orange).

To visualize the effect of structural pruning on each oncology efficacy endpoint, Figure 4.4 shows radar charts of the F1 scores for the standard BioBERT (blue) against the versions pruned by 20% and 25% (green). The plots indicate that most medical efficacy endpoints suffer little or no degradation after the pruning process. The two polygons coincide on every high-count label, such as DFS, PFS, and OR, showing that 20% and 25% structural pruning is virtually loss-free. For rare DoR classes (`DoR`, `DoR_CIH`, `DoR_CIL`), which have very few examples in both the training and test sets, structured pruning actually improves performance.

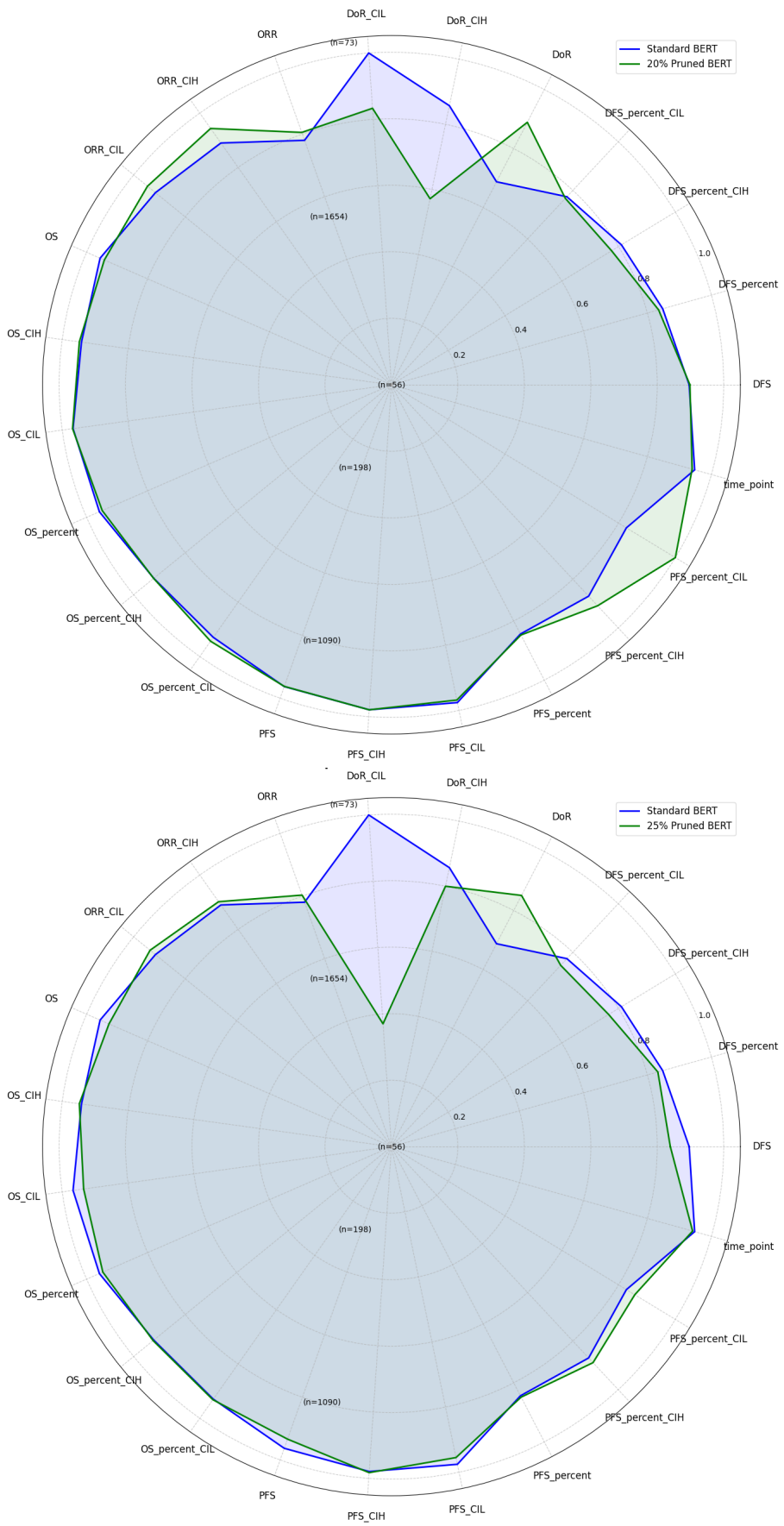

Figure 4.3: Radar chart comparing class-wise F1 scores for the standard BERT model (blue) and the version with 20% and 25% structured pruning (green).

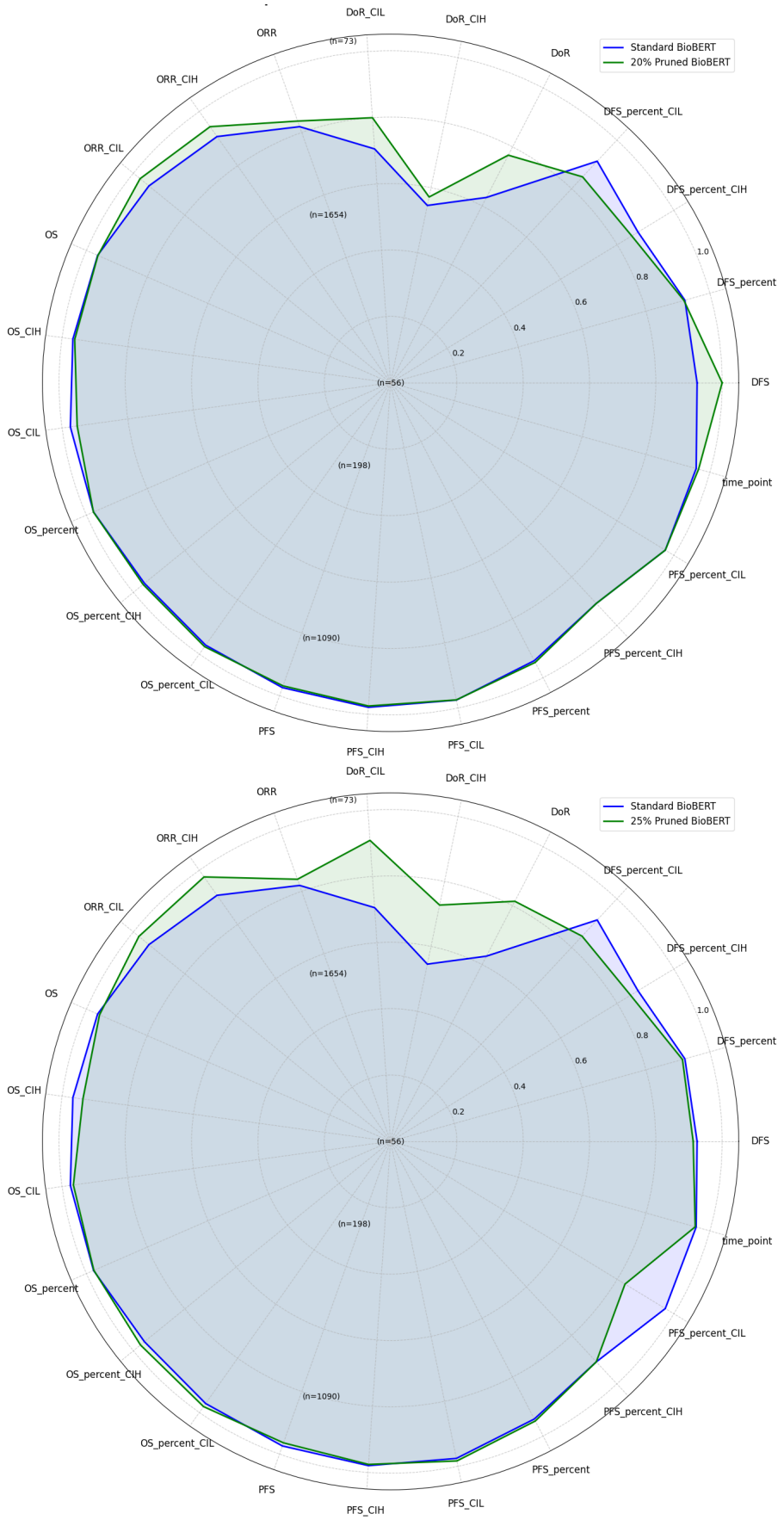

Figure 4.4: Radar chart comparing class-wise F1 scores for the standard BioBERT model (blue) and the version with 20% and 25% structured pruning (green).

4.2 Quantization

In this section, the results of our evaluations are presented by comparing the performances of BERT, DistilBERT, and BioBERT before and after applying dynamic quantization. Our primary objective is to optimize inference time and reduce model size while maintaining competitive accuracy. For each model, inference latency was calculated as the total time taken to process all samples (test dataset), divided by the total number of samples and expressed in milliseconds per sample. Throughput was simultaneously measured as the number of samples processed per second, obtained by dividing the total number of samples by the total inference time. To structure the presentation of our findings, we first analyze the impact of dynamic quantization of weights on inference time and throughput. Then we examine the models' F1 scores for each entity to assess how well the models generalize to the Name Entity Recognition (NER) task post-quantization. Finally, we delve into the models' global performances in terms of accuracy, F1 scores, recall, and precision across all models.

The standard BERT model as depicted in Figure 4.5 achieved an inference latency of 30.28 ms per sample, while the quantized version reduced it to 11.63 ms, which yields 2.6-fold faster inference. Additionally, throughput for BERT after quantization was increased from 33.0 to 86.0 samples per second. This means that the quantized model can handle about 2.6 times more samples within the same time frame. Regarding F1 scores, as illustrated in Figure 4.6, the F1 scores for all entities of standard BERT before quantization and the quantized BERT model are presented. For entities with a sufficient number of samples, the observed F1 score drop due to quantization was either small or negligible in most cases. Additionally, in PFS- entities, except for PFS-present, the quantized model showed a lower F1 score compared to the standard BERT model. Furthermore, the quantized model achieved better F1 scores for DoR, DoR-CIH, and DoR-CIL, despite the limited number of samples for these entities.

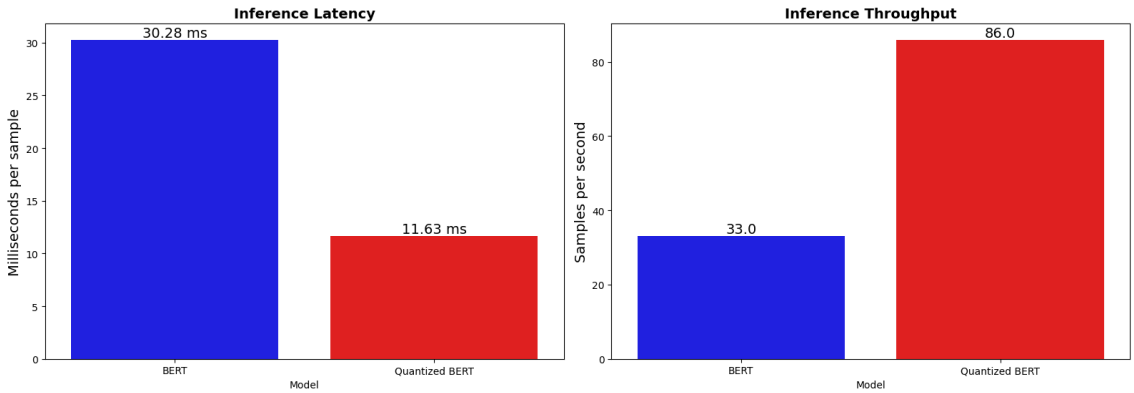

Figure 4.5: Inference latency and throughput per sample (ms) for standard BERT and Quantized BERT on the baseline test dataset on CPU

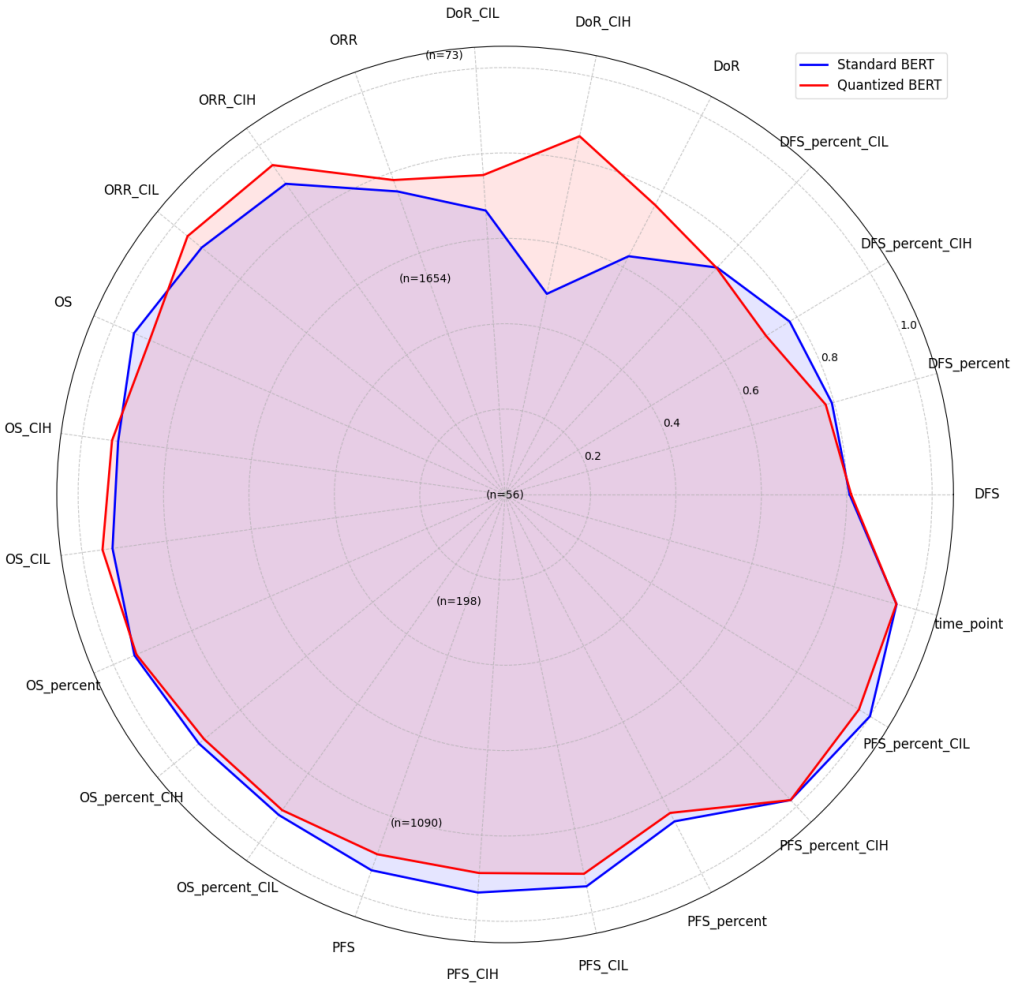

Figure 4.6: Radar charts comparing the F1 scores of quantized BERT (green) and standard BERT (blue) across all entities.

For the BioBERT model before and after applying dynamic quantization, as shown in Figure 4.7, before quantization, it achieved an inference time of 30.94 ms per sample. After quantization, the inference time was reduced to 14.14 ms per sample. Furthermore, throughput was increased from 32.2 to 70.7 samples per second. These observations yield 2.4-fold faster inference time, which indicates the model can handle 2.4 times more samples in the same amount of time. The F1 scores for all entities of BioBERT before and after applying dynamic quantization are shown in Figure 4.8. BioBERT experienced a clear drop, particularly in PFS- entities except for PFS-present. While the overall F1 drop was negligible, the quantized model for the entities with fewer than 30 samples, such as DoR-CIH, DoR-CIL, and PFS-percent-CIL, exhibited the most F1 drops among all entities, although the F1 score for DoR was higher in the quantized model.

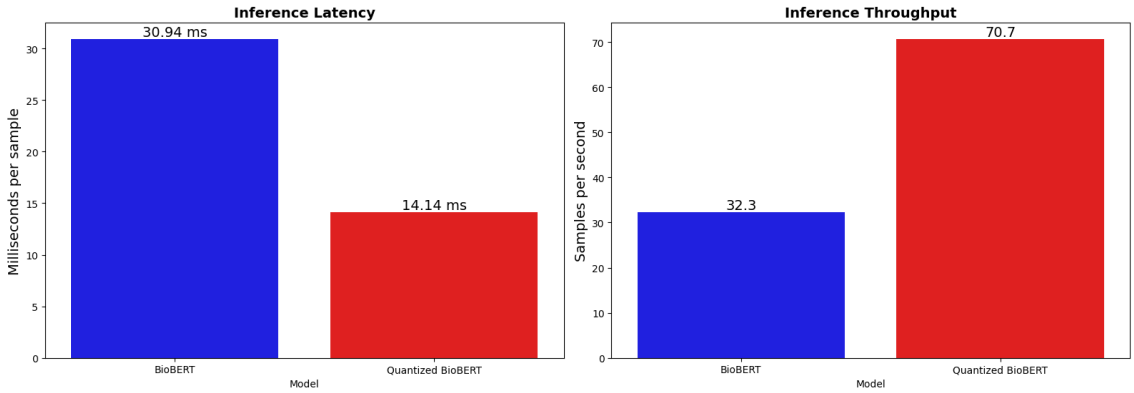

Figure 4.7: Inference latency and throughput per sample (ms) for standard BioBERT and quantized BioBERT on the baseline test dataset on CPU.

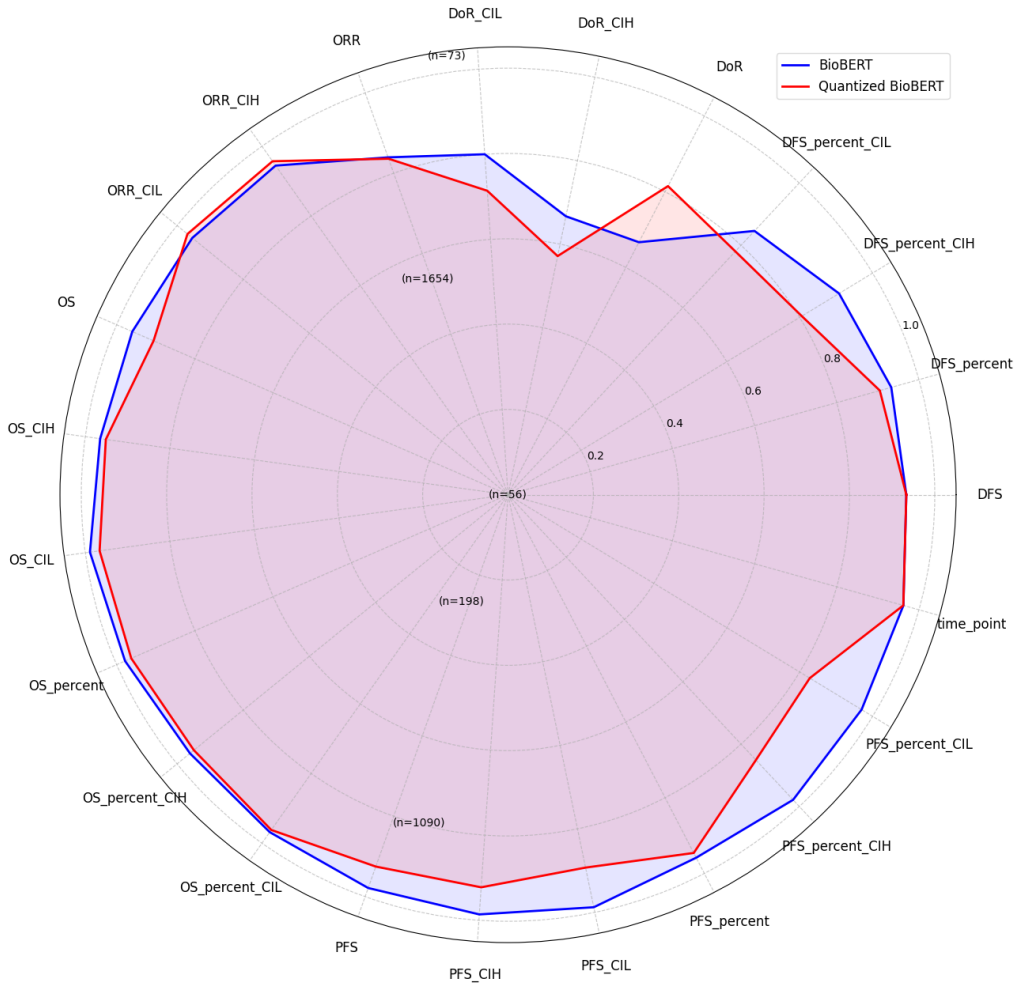

Figure 4.8: Radar charts comparing the F1 scores of quantized BioBERT (green) and standard BioBERT (blue) across all entities.

Similarly, for the DistilBERT model shown in Figure 4.9, the unquantized baseline exhibited an inference latency of 16.61 ms per sample. In the quantized model,

inference latency was reduced to 7.14 ms, which represents a 2.3-fold reduction in latency. Moreover, throughput for DistilBERT improved from 60.2 to 140.1 samples per second, which indicates a 2.3-fold increase in processing capacity. Lastly, the F1 scores of the DistilBERT model before and after dynamic quantization are shown in Figure 4.10. Similar to the other two models, the quantized DistilBERT also exhibits a better F1 score for DoR, despite this entity having fewer samples compared to the others. While the overall performance of the quantized DistilBERT remains acceptable, it experienced F1 score drops relative to the standard DistilBERT in entities such as PFS, PFS-CIL, PFS-CIH, OS-percent-CIL, and OS-percent-CIH.

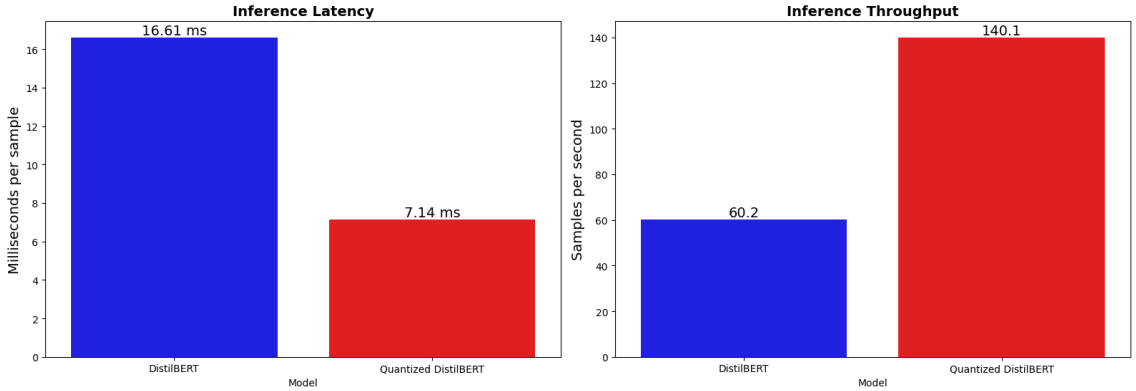

Figure 4.9: Inference latency and throughput per sample (ms) for standard DistilBERT and quantized DistilBERT on the baseline test dataset on CPU.

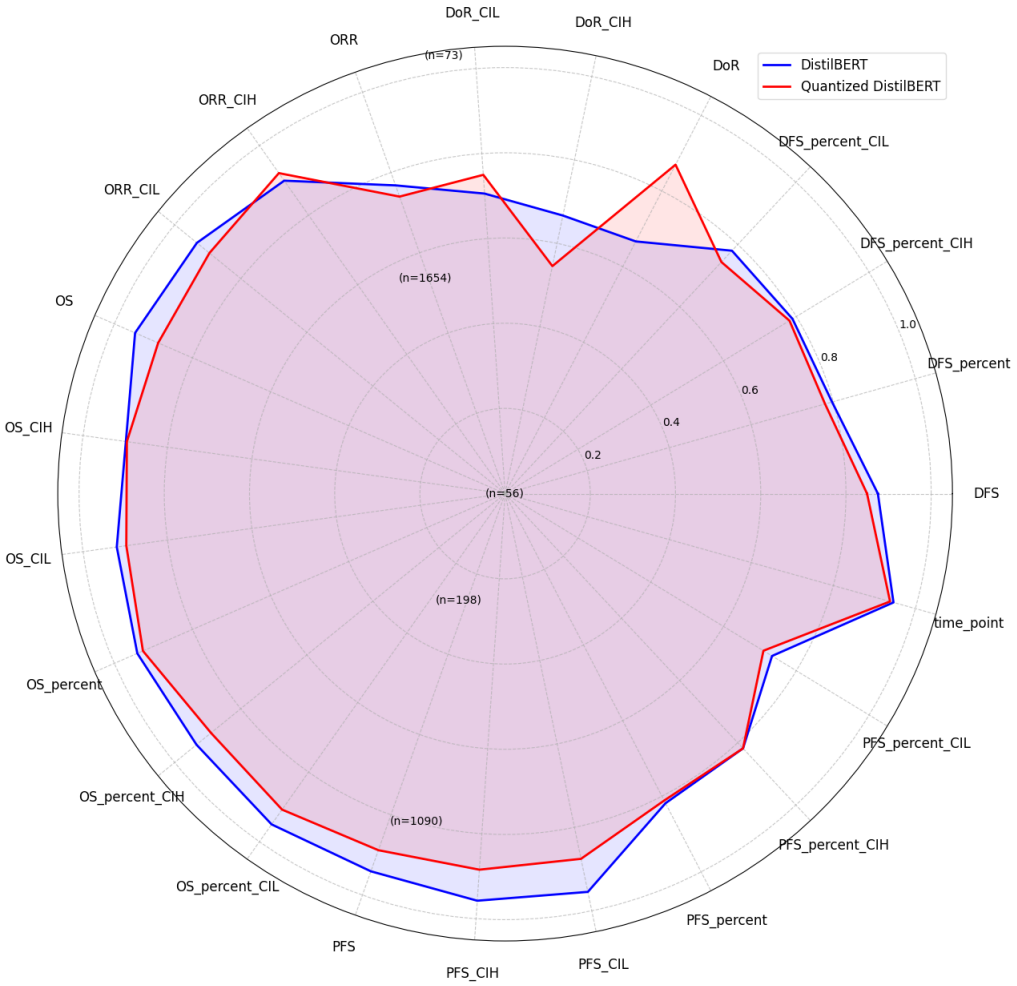

Figure 4.10: Radar charts comparing the F1 scores of quantized DistilBERT (green) and standard DistilBERT (blue) across all entities.

The overall performance metrics across all models before and after applying quantization are presented in Table 4.1. Regarding overall accuracy, negligible drops were observed after quantization; as can be seen, BioBERT experienced a decrease of 0.007 (from 0.985 to 0.978), BERT a drop of 0.003 (from 0.969 to 0.966), and DistilBERT a reduction of 0.007 (from 0.968 to 0.961). In terms of overall F1 scores, a minor decrease was noted for BERT (0.006, from 0.912 to 0.906), while BioBERT and DistilBERT experienced slightly larger reductions of 0.018 (from 0.959 to 0.941) and 0.020 (from 0.911 to 0.891), respectively.

Moreover, precision generally saw slight improvements for BioBERT (increasing by 0.011, from 0.952 to 0.963) and BERT (increasing by 0.013, from 0.891 to 0.904). However, DistilBERT experienced a minor decrease of 0.003 (from 0.903 to 0.900). Recall decreased more than other metrics across all models. BioBERT exhibited the largest reduction of 0.047 (from 0.967 to 0.920), followed by DistilBERT with a drop of 0.038 (from 0.920 to 0.882), and BERT with a decrease of 0.027 (from 0.934 to 0.907). The I-BERT model, included in Table 4.1 for comparison, consistently demonstrated lower overall performance metrics (Accuracy: 0.952, F1: 0.881,

Precision: 0.821, Recall: 0.950) compared to the standard (unquantized) BERT, BioBERT, and DistilBERT models. As a result, the decision was made not to use I-BERT due to its poor performance compared to other quantized models.

Model	**Accuracy**	**F1 Score**	**Precision**	**Recall**
Standard BioBERT	0.985	0.959	0.952	0.967
Quantized BioBERT	0.978	0.941	0.963	0.920
Standard BERT	0.969	0.912	0.891	0.934
Quantized BERT	0.966	0.906	0.904	0.907
Standard DistilBERT	0.968	0.911	0.903	0.920
Quantized DistilBERT	0.961	0.891	0.900	0.882
I-BERT	0.952	0.881	0.0.821	0.0.950

Table 4.1: Comparison of global performance metrics for I-BERT, standard and quantized BERT, BioBERT, and DistilBERT.

4.3 Pruning and Quantization

Having everything at hand from quantization and pruning, the combined impact of applying both techniques to the models is investigated to have a smaller and faster inference time. To experiment effects of quantization on the pruned model, dynamic quantization was applied on BERT and BioBERT models at 20% and 25% fixed sparsity. Additionally, Figure 4.11 demonstrates the impact of two compression stages on the BERT model in inference latency and throughput per sample. Removing 20% of the parameters already lowers the inference latency by 30.1 ms to 24.4 ms (-19%) and raises throughput by 33.2 to 41.1 ms per sample (+24%). With a quarter of the model removed, latency drops to 23.7 ms and throughput rises to 42.2 ms per sample. Applying dynamic quantization to either model halves the runtime to 12.41 ms/80.6 ms (20% mask) and 11.7 ms / 85.8 ms (25% mask). The 2.5 to 2.6-fold acceleration indicates that combining pruning and quantization gives a cumulative speed advantage over the dense model on CPU.

Figures 4.13 (for standard BERT) and 4.14 (for BioBERT) depict radar charts of F1 scores for all efficacy end-points: the dense model (blue), the 20% and 25% pruned models (green), and their quantized versions (red).
For standard BERT (Figure 4.13), the compressed polygons overlap the baseline on every high-count label (e.g., DFS, OS, PFS, ORR; n > 1000); the only noticeable decline appears on the rare DoR–CIH and DoR–CIL classes, where F1 falls from 0.83 to 0.58 at 25% sparsity plus quantization.
As in standard BERT, BioBERT (Figure 4.14) shows no degradation on high-count end-points up to 25% sparsity plus quantization, and even records noticeable gains on low-count labels such as DoR–CIH and DoR–CIL (+0.30 F1), confirming that the compression pipeline maintains or occasionally improves performance across the label set.

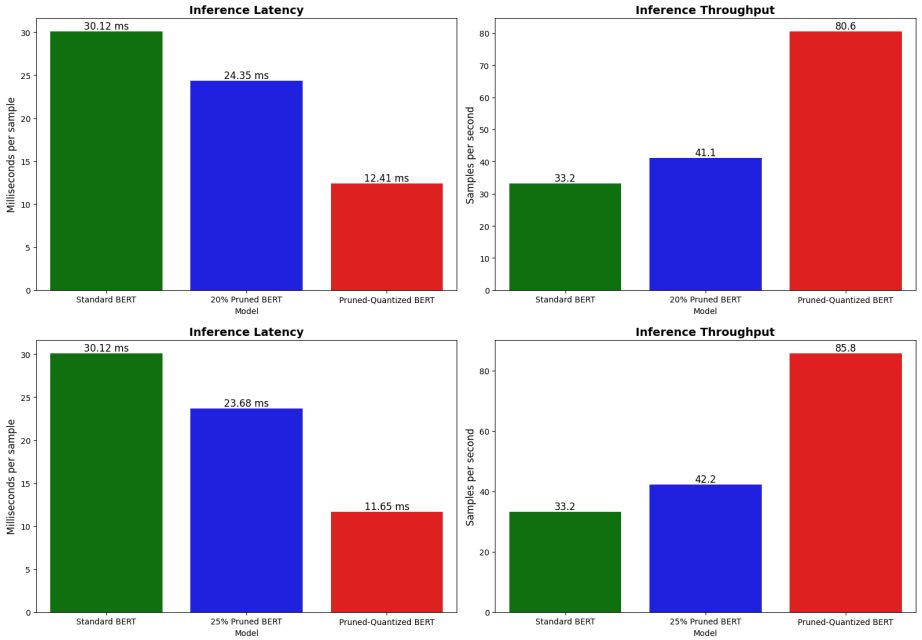

Figure 4.11: Inference latency and throughput per sample (ms) of standard BERT (green), 20% and 25% pruned BERT (blue), and 20% and 25% pruned and quantized BERT (red).

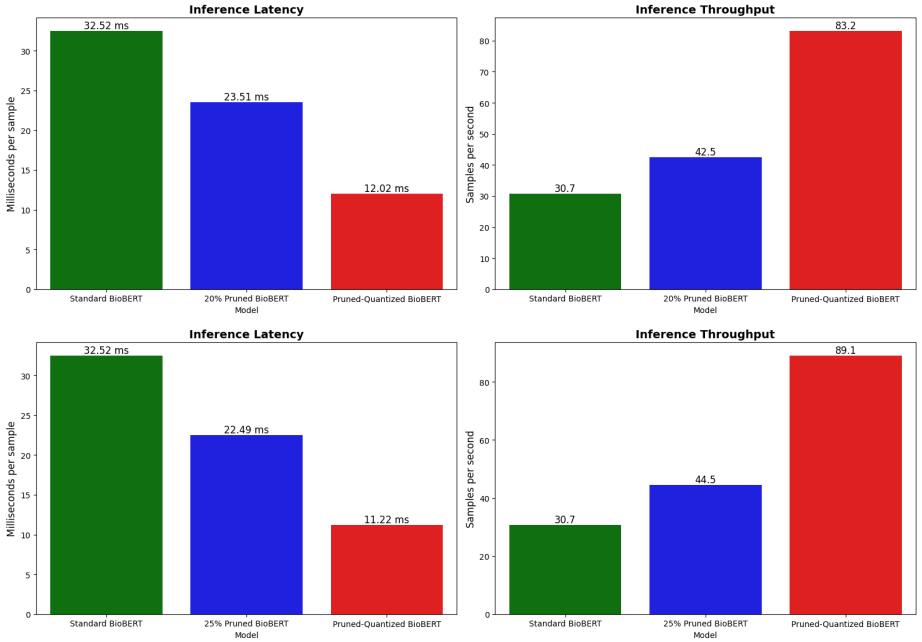

Figure 4.12: Inference latency and throughput per sample (ms) of standard BioBERT (green), 20% and 25% pruned BioBERT (blue), and 20% and 25% pruned and quantized BioBERT (red).

Figure 4.13: Radar charts comparing F1 score performance of the standard BERT (blue), 20% and 25% pruned BERT (green), and 20% and 25% pruned and quantized BERT (red) models across all defined entity endpoints

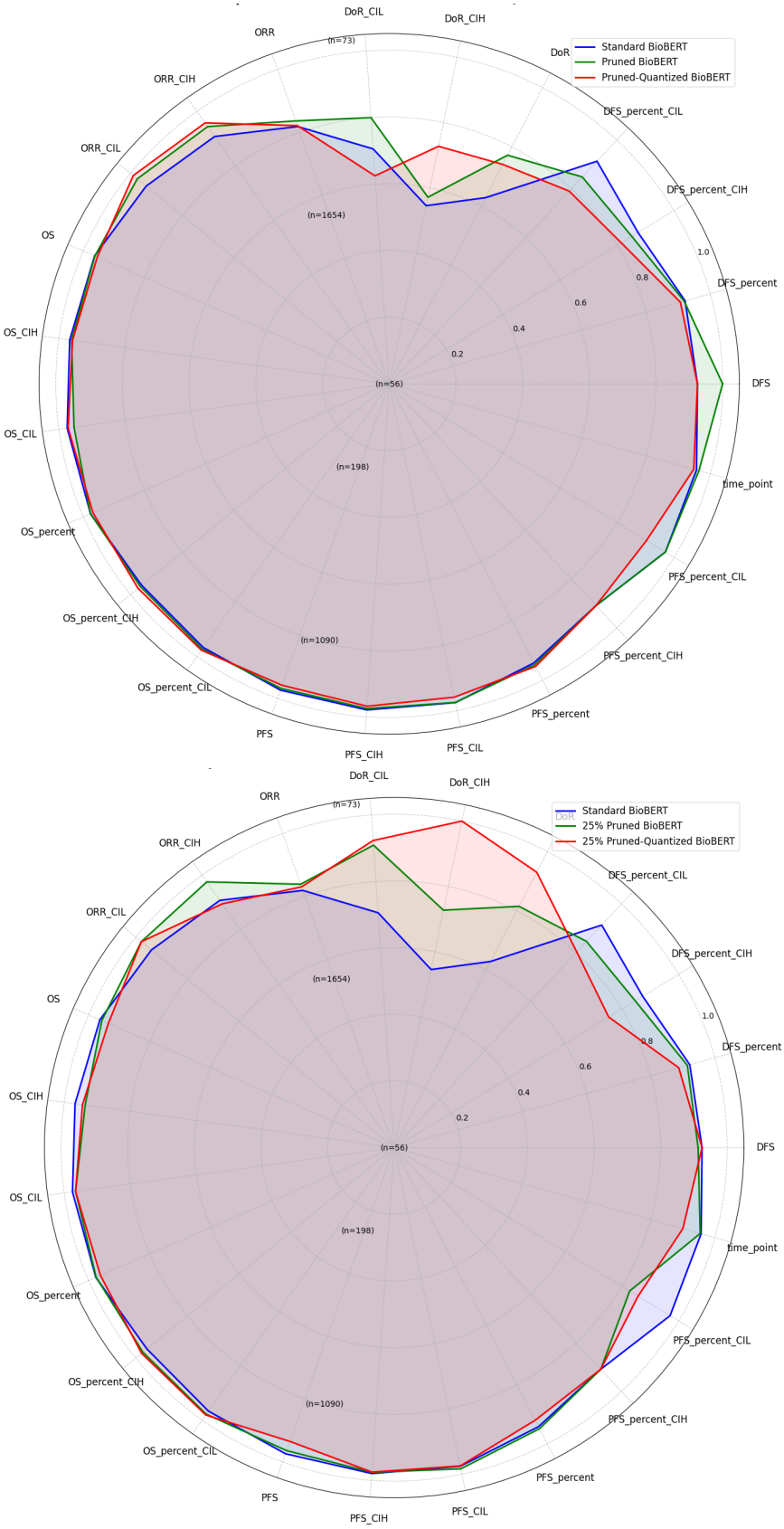

Figure 4.14: Radar charts comparing F1 score performance of the standard BioBERT (blue), 20% and 25% pruned BioBERT (green), and 20% and 25% pruned and quantized BioBERT (red) models across all defined entity endpoints

Figures 4.15 (for standard BERT) and 4.16 (for BioBERT) present the performance metrics such as accuracy, F1 score, precision and recall for the dense models (blue) together with models pruned by 20% and 25% (red) and the corresponding combined pruned model quantized to 8-big integers (green).

For standard BERT, after performing both compression methods, at 20% sparsity, Accuracy drops by 0.973 to 0.970 and F1 score by 0.928 to 0.916, while precision increases from 0.912 to 0.930, the loss is driven entirely by a 3% recall dip.
At the same sparsity level in BioBERT, pruning and quantization decrease accuracy (0.982 to 0.982) and F1 (0.954 to 0.948). As with standard BERT, precision increases by 1.5% and recall decreases by 2.7%. Pushing to 25% sparsity, BioBERT performance remains acceptable with (F1 0.932, accuracy 0.974), whereas the corresponding BERT model shows a performance drop.

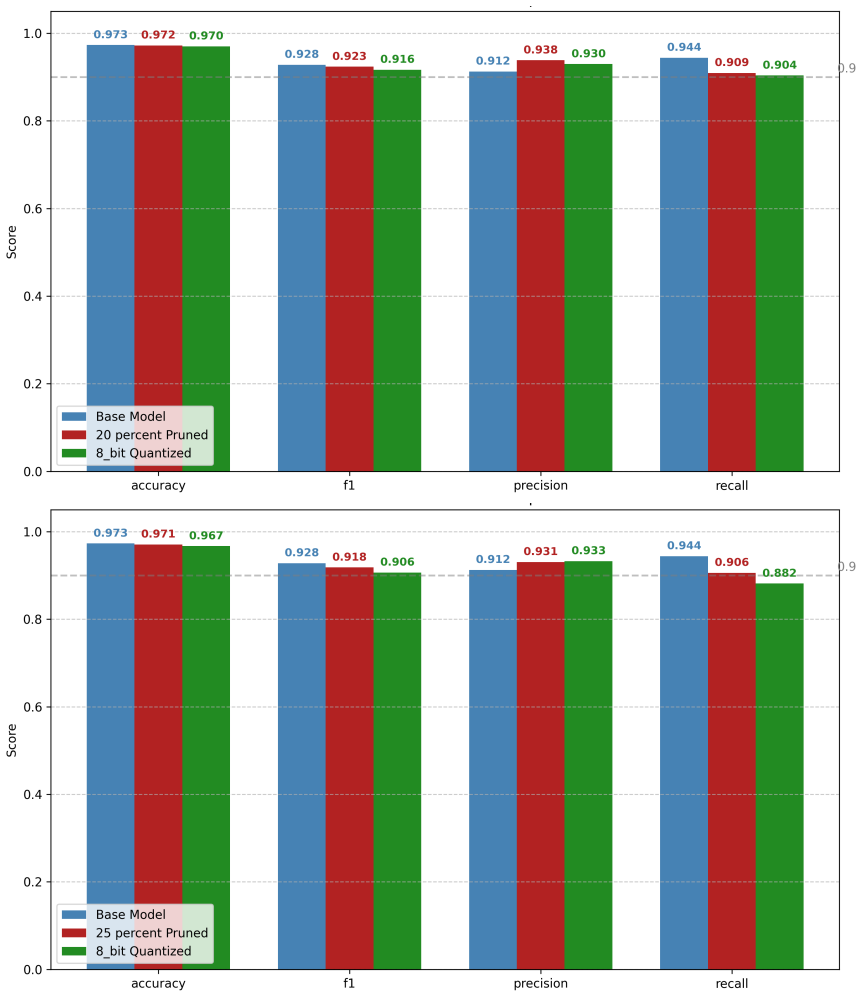

Figure 4.15: Global performance metrics of the standard BERT (blue), 20% and 25% pruned BERT (red), and 20% and 25% pruned and quantized BERT (green) models.

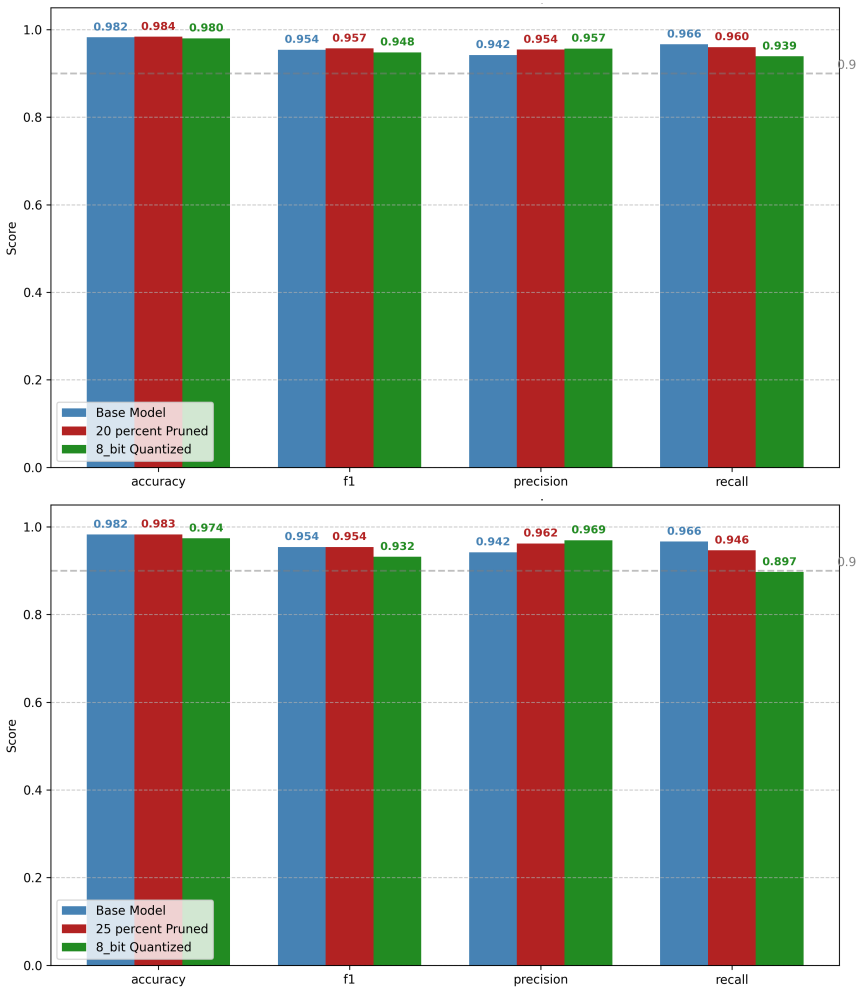

Figure 4.16: Global performance metrics of the standard BioBERT (blue), 20% and 25% pruned BioBERT (red), and 20% and 25% pruned and quantized BioBERT (green) models.

5

Discussion

This study investigated the dynamic quantization and structural pruning of three BERT variations: BERT, BioBERT, and DistilBERT, across three fixed sparsity levels. Subsequently, the combined effect of these quantization and pruning strategies was evaluated. The objective of this project was to reduce the memory footprint and inference time of these models without compromising their performance. Our analysis begins with insights that were derived from the structural pruning experiments on BERT and BioBERT and further examines their impact on performance metrics, inference time, and the observed patterns in pruned neurons and heads.

As demonstrated in the previous chapter in Figures 4.1 and 4.2, the performance of both models after 20% pruning generally remained robust. For pruned BERT, the results showed a minor impact on performance, with accuracy dropping by 0.001, F1 score by 0.005, and recall by 0.035. Notably, for BioBERT at 20% sparsity, the performance metrics for F1 score, accuracy, and precision demonstrated an improvement compared to the standard BioBERT before structural pruning, while its recall experienced a slight drop of only 0.006.

The enhanced performance observed in the pruned BioBERT model can be interpreted as follows: Given that BioBERT was pretrained on specialized data in the medical domain, the selective removal of less informative parameters (specifically, 0.39% neuron sparsity and 0.02% head sparsity in the 20% pruned BioBERT) likely enabled the remaining parameters, which presumably contain more important information, to contribute more effectively to generalization. These remaining parameters were thus able to perform better after pruning. Additionally, this improvement was critically influenced by the strategic nature of the pruning method. Instead of randomly removing the model's parameters, the Fisher Information Matrix was a key factor in achieving these improved results, as its objective is to identify and eliminate parameters that have the least impact on the model's output or loss function.

For the 25% sparsity level, the BERT model indicated acceptable performance. While recall experienced a 3.8% drop, other performance metrics, such as accuracy and F1 score, remained competent compared to the standard BERT and the 20% Pruned BERT. Since the accuracy drop for 25% sparsity is still less than 1% (at 0.002), and the F1 score experienced only a 1% reduction compared to the standard BERT model, it can be effectively used for deployment. Regarding 25% pruned BioBERT, the model indicated the same value for accuracy and F1 score compared to standard BioBERT. This result also shows that even after achieving 0.41% neuron

sparsity and 0.036% head sparsity of BioBERT, the performance, except for recall with a 2% drop, remained largely intact. One can analyze that for BioBERT, even after 25% of sparsity, since the model is pretrained on specific medical data, the informative neurons contributed more to the model when the redundant neurons and heads were pruned.

Focusing on the pruned models' inference time, for the 20% pruned BERT model, the inference time experienced a 19% acceleration. For the 25% pruned model, a 21% improvement in inference time was observed. At these sparsity levels, since the inference time in both the pruned models is so close, one might therefore opt for the latter, prioritizing performance over marginal inference gains.

Moreover, for BioBERT, the inference time after 20% pruning accelerated by 27.2%, and at 25% sparsity, it exhibited a 30.8% faster inference time. Due to the nearly identical performance between the 25% sparsity level and the standard BioBERT model (as discussed previously), one might consider this level of sparsity for BioBERT to leverage both lower inference time and higher pruning level.

In pruning, the experiments leveraged the benefits of pruning in terms of sparsity while maintaining the same performance metrics. Shifting our focus from structural pruning, we now turn to findings from our dynamic quantization experiments. In the quantized models, which were implemented in this project, the models exhibited notable inference latency time accelerations. Dynamic quantization was applied to all three BERT variations, such as BioBERT, BERT, and DistilBERT, with negligible performance metrics sacrifice.

The finding in previous chapters shows that across all three models, 2.3 to 2.6-fold acceleration was exhibited in inference time. Additionally, this inference acceleration can be attributed to a combination of ONNX Runtime optimizations and dynamic quantization techniques. The optimization step applies several graph-level transformations, such as constant folding, operator fusion, node elimination, and attention fusions. These techniques reduce computational overhead by computing constant values before inference, combining multiple smaller operations into one more efficient larger operation, and simplifying the computational graph, which results in inference speedup.

Furthermore, dynamic quantization, particularly the conversion of weights from 32-bit floating point to 8-bit integers, is arguably the most impactful component for the inference speedup, especially on edge devices like CPUs. Integer operations are more efficient than floating-point arithmetic on these platforms, which enables faster execution without affecting model accuracy. Moreover, dynamic quantization by using pre-quantized 8-bit integer weights and quantizing the activations on the fly helps accelerate the inference time. Additionally, 8-bit integer numbers occupy one-fourth of the memory of 32-bit floating point numbers, which means less data needs to be moved from memory to the CPU's processing units. This reduces memory bandwidth bottlenecks, which are often a major limitation in large model inference.

For the quantized model in terms of accuracy, F1, precision, and recall, in all three models, the accuracy and F1 drop were below 1% and 2% respectively and a small

increase in precision was found which suggested that quantization in this specific task, was increasing the number of False Positives (FPs). Although BioBERT after and before quantization exhibited the best performance among all models, its recall experienced a 4% drop, as earlier was mentioned due to FPs where the model predicts an entity, but it's incorrect.

Finally, we combined pruning and dynamic quantization techniques to leverage both the inference speedup from quantization and the efficiency of the pruned models, applying these to models with 20%, 25%, and 30% sparsity levels. Regarding performance metrics, the 20% pruned and quantized BioBERT model notably demonstrated better performance compared to its solely quantized models. For BERT, while the F1 score exhibited a 0.006 drop compared to the quantized model, the remaining performance metrics increased slightly after the combination of dynamic quantization and pruning. While quantization leads to the inference speedup and memory reduction, it can simultaneously introduce noise across the entire model. This noise can lead to slight shifts in weight values, distorted activations, and degradation in fine-grained, token-level embeddings. Since the pruning in this project was applied before dynamic quantization and after pruning, the less informative weights that contribute least to the model's outputs are removed. Due to the nature of structural pruning and the reduction in less informative neurons and heads, the noise introduced during quantization decreased, consequently, the performance metrics increased slightly after the combination of pruning and quantization compared to the solely quantized model.

Besides the performance metrics, the inference time of BioBERT after the combination of these two mentioned techniques for a 20% sparsity level slightly improved. It is essential to highlight that, for BERT and BioBERT, based on the results for F1 scores across all the entity endpoints, the F1 for popular entities in both 20% and 25% sparsity levels was competent. Also, in low-frequency entities such as DoR-CIL and CoR-CIH, the quantized BioBERT with 25% of sparsity experienced better F1 scores, and for entities like PFS-precent-CIL and PFS-precent-CIH, the quantized BERT model maintained its F1 scores in 25% pruning level and improved its F1 scores in 20% of pruning. This can be due to the regularization roles of pruning, which can improve generalization and lead to better performance on the test set, especially for less-represented categories. The interpretations for this improvement in F1 scores in low-frequency entities can be seen as follows: when pruning was applied before quantization, the model was already adjusted to rely on the most important weights. This combination of pruning and dynamic quantization acts as some fine-tuning or calibration, so the model's focus might go on a smaller but more meaningful parameter space, which improved the F1 scores for these challenging low-frequency entities.

In summary, this analysis demonstrates the capabilities of both structural pruning and dynamic quantization in optimizing large language models like BERT and BioBERT for deployment. While individual techniques yielded progress in efficiency, especially in reducing inference time and memory footprint, our findings show acceptable behavioral nuances in performance metrics. Notably, BioBERT consistently showed better performance across various compression methods. After the pruning

technique, it maintained or even enhanced key performance metrics and accelerated inference time in the quantization technique. The combination of these methods further indicates their ability to be used in real-world applications, even on edge hardware devices like CPUs. These results not only affirm the practical viability of model compression for clinical NLP tasks but also provide valuable insights into the differential impacts of pruning and quantization on model generalization.

5.1 Conclusion

The journey through this research has demonstrated the potential of model compression to make large language models like BERT more practical and accessible. This study aimed to examine optimization techniques on the oncology NLP pipeline in AstraZeneca to reduce memory footprint and inference time on CPU nodes without compromising performance metrics, especially accuracy and F1. The existing NLP pipeline, consisting of three main BERT variations, namely BERT, BioBERT, and DistilBERT, was trained and then prepared for subsequent optimization.

The objective was achieved by the implementation of structural pruning, dynamic quantization individually, and subsequently, a combination of structural pruning and dynamic quantization.

For structural pruning, Fisher Information Matrix was applied to BioBERT and BERT at three levels of neuron and head sparsity (20%, 25%, and 30%). For the 20% and 25% pruning levels, the models generally maintained strong performance in terms of accuracy and F1 scores. However, beyond 25% sparsity, the models' performance experienced a drastic drop.

In the case of quantization, a post training quantization (PTQ) approach, facilitated by ONNX Runtime dynamic quantization, was applied to BERT, BioBERT, and DistilBERT. Dynamic quantization, which involves converting 32-bit floating-point weights before inference and quantizing activations on-the-fly, proved its ability to decrease inference time effectively.

Additionally, BERT and BioBERT were optimized through a combination of dynamic quantization and structural pruning, utilizing the same three sparsity levels as the individual pruning experiments. The results from the combination of quantization and 20% of pruning for BERT and BioBERT achieved competent performance. During inference, the pruning method provided regularization and calibration effects, so compared to the individual quantization, F1 scores and the accuracy in this level of sparsity appeared better. Overall, BioBERT with the combination of dynamic quantization and 20% of structured pruning can be considered the best model with a 0.02% accuracy drop among all experiments.

Finally, it is worth mentioning that, although I-BERT was implemented in this project, it was later decided not to use it due to its poor performance and specific hardware requirements for inference. This research confirms that with careful application of these compression techniques, the NLP models can be used in real-world deployment.

Bibliography

[1] S. News, "Ancient myths reveal early fantasies of artificial life," 2019. [Online]. Available: https://news.stanford.edu/stories/2019/02/ancient-myths-reveal-early-fantasies-artificial-life.

[2] J. Devlin, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[3] Y. Liu, M. Ott, N. Goyal, *et al.*, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[4] E. Mueller, J. Hansjakob, D. Auge, and A. Knoll, "Minimizing inference time: Optimization methods for converted deep spiking neural networks," in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021, pp. 1–8.

[5] J. Chen and X. Ran, "Deep learning with edge computing: A review," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1655–1674, 2019.

[6] Z. Wang, "Sparsednn: Fast sparse deep learning inference on cpus," *CoRR*, vol. abs/2101.07948, 2021.

[7] IQVIA Ltd., *I2e: Information extraction platform*, I2E is developed and marketed by IQVIA Ltd. Further information can be obtained from http://www.linguamatics.com, n.d.

[8] J. Lee, W. Yoon, S. Kim, *et al.*, "Biobert: A pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, 2020.

[9] Y. Gu, R. Tinn, H. Cheng, *et al.*, "Domain-specific language model pretraining for biomedical natural language processing," *ACM Trans. Comput. Healthcare*, vol. 3, no. 1, Oct. 2021. DOI: 10.1145/3458754. [Online]. Available: https://doi.org/10.1145/3458754.

[10] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: Smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

[11] A. Gendrin-Brokmann, E. Harrison, J. Noveras, *et al.*, "Investigating deep-learning nlp for automating the extraction of oncology efficacy endpoints from scientific literature," *Information Based Medicine*, 2024, Available online 4 July 2024. DOI: 10.1016/j.ibmed.2024.100152.

[12] H. Bai, W. Zhang, L. Hou, *et al.*, "Binarybert: Pushing the limit of bert quantization," *arXiv preprint arXiv:2012.15701*, 2020.

[13] O. Zafrir, G. Boudoukh, P. Izsak, and M. Wasserblat, "Q8bert: Quantized 8bit bert," in *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing-NeurIPS Edition (EMC2-NIPS)*, IEEE, 2019, pp. 36–39.

[14] S. Kim, A. Gholami, Z. Yao, M. W. Mahoney, and K. Keutzer, "I-bert: Integer-only bert quantization," in *International conference on machine learning*, PMLR, 2021, pp. 5506–5518.

[15] H. Qin, Y. Ding, M. Zhang, *et al.*, "Bibert: Accurate fully binarized bert," *arXiv preprint arXiv:2203.06390*, 2022.

[16] X. Jiao, Y. Yin, L. Shang, *et al.*, "Tinybert: Distilling bert for natural language understanding," *arXiv preprint arXiv:1909.10351*, 2019.

[17] E. Kurtic, D. Campos, T. Nguyen, *et al.*, *The optimal bert surgeon: Scalable and accurate second-order pruning for large language models*, 2022. arXiv: 2202.09906 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2202.09906.

[18] Hugging Face, *Hugging Face Website and Hub*, https://huggingface.co/, Accessed: [Date of Access, e.g., May 20, 2025], 2024.

[19] Microsoft Azure, *Azure Databricks Documentation*, https://docs.microsoft.com/en-us/azure/databricks/, Accessed: [Date of Access, e.g., May 20, 2025], 2024.

[20] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[21] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[22] D. into Deep Learning, *Attention mechanisms and transformers*, Accessed: Month Day, Year. [Online]. Available: https://d2l.ai/chapter_attention-mechanisms-and-transformers/index.html#.

[23] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, *Dive into deep learning*, https://d2l.ai/chapter_attention-mechanisms-and-transformers/transformer.html, Accessed: 2025-04-28, 2021.

[24] G. Hinton, O. Vinyals, and J. Dean, *Distilling the knowledge in a neural network*, 2015. arXiv: 1503.02531 [stat.ML]. [Online]. Available: https://arxiv.org/abs/1503.02531.

[25] W. Wang, W. Chen, Y. Luo, *et al.*, "Model compression and efficient inference for large language models: A survey," *arXiv preprint arXiv:2402.09748*, 2024.

[26] W. Dai, H. Deng, M. Rong, *et al.*, *Flexible operator fusion for fast sparse transformer with diverse masking on gpu*, 2025. arXiv: 2506.06095 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2506.06095.

[27] A. Acharya, U. Bondhugula, and A. Cohen, "Effective loop fusion in polyhedral compilation using fusion conflict graphs," *ACM Transactions on Architecture and Code Optimization*, vol. 17, no. 4, pp. 1–26, Sep. 2020, ISSN: 1544-3973. DOI: 10.1145/3416510. [Online]. Available: http://dx.doi.org/10.1145/3416510.

[28] A. Phani, B. Rath, and M. Boehm, "Lima: Fine-grained lineage tracing and reuse in machine learning systems," in *Proceedings of the 2021 International Conference on Management of Data*, ser. SIGMOD '21, Virtual Event, China: Association for Computing Machinery, 2021, pp. 1426–1439, ISBN: 9781450383431. DOI: 10.1145/3448016.3452788. [Online]. Available: https://doi.org/10.1145/3448016.3452788.

[29] H. Zhang, Z. Yu, G. Dai, *et al.*, *Understanding gnn computational graph: A coordinated computation, io, and memory perspective*, 2021. arXiv: 2110.09524 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2110.09524.

[30] T. Theodoridis, M. Rigger, and Z. Su, "Finding missed optimizations through the lens of dead code elimination," in *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '22, Lausanne, Switzerland: Association for Computing Machinery, 2022, pp. 697–709, ISBN: 9781450392051. DOI: 10.1145/3503222.3507764. [Online]. Available: https://doi.org/10.1145/3503222.3507764.

[31] U.S. National Library of Medicine, *MEDLINE database*, https://www.nlm.nih.gov/medline/index.html, Accessed: 2025-05-21, 2023.

[32] Linguamatics, *I2e natural language processing platform*, https://www.linguamatics.com/products/i2e, Accessed: 2025-05-21, 2023.

[33] Heartex, *Label studio: Open source data labeling tool*, https://github.com/heartexlabs/label-studio, Accessed: 2025-05-21, 2020.

[34] W. Kwon, S. Kim, M. W. Mahoney, J. Hassoun, K. Keutzer, and A. Gholami, "A fast post-training pruning framework for transformers," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 101–24 116, 2022.

[35] Microsoft, *ONNX Runtime: Accelerating Machine Learning Inference*, https://onnxruntime.ai/, Accessed: 2025-05-21.

[36] O. R. Contributors, *Onnx runtime quantization tools*, https://github.com/microsoft/onnxruntime/tree/main/onnxruntime/python/tools/quantization, Accessed: 2025-05-21, 2023.

[37] A. Paszke, S. Gross, F. Massa, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., vol. 32, 2019.

[38] H. Nakayama, *Seqeval: A python framework for sequence labeling evaluation*, https://github.com/chakki-works/seqeval, Accessed: 2025-05-21, 2018.

A

Appendix 1

Figure A.1 shows the heat-map of BioBERT pruned to 30% overall sparsity. The top panel focuses on the attention heads, at this sparsity level the Fisher information pruning method has dropped 3 of the 144 heads outright, which appear in white. The remaining heads display different shades of blue. This color variation comes from the third pruning stage, rescaling, where the binary masks on the surviving heads were fine tuned and converted to float values to regain some of the loss introduced earlier. The bottom panel covers FFN. After the full three-stage routine, about 56.5% of the FFN neurons were removed, visible as the light-blue cells scattered across the layers.

Figure A.2 shows the radar charts for standard BERT (top panel) and standard BioBERT (bottom panel), together with their 30% sparsity pruned versions and the corresponding pruned quantized variants. At this higher sparsity, the F1 scores for most clinical-efficacy endpoints fall significantly compared with the lower sparsity of 20% and 25%. Even so, BioBERT has less performance drop than standard BERT.

Figure A.3 shows the inference time for standard BERT (top row) and standard BioBERT (bottom row) next to their 30% sparsity pruned models and the pruned + quantized versions. At 30% sparsity, the latency is lower than the latency at 20% and 25% sparsity, but this extra speed comes at a cost. As one can see in Figure 4.15 clearly, the global F1 score drops to 0.827 for BERT and 0.886 for BioBERT, along with similar declines in recall, while gaining some precision.

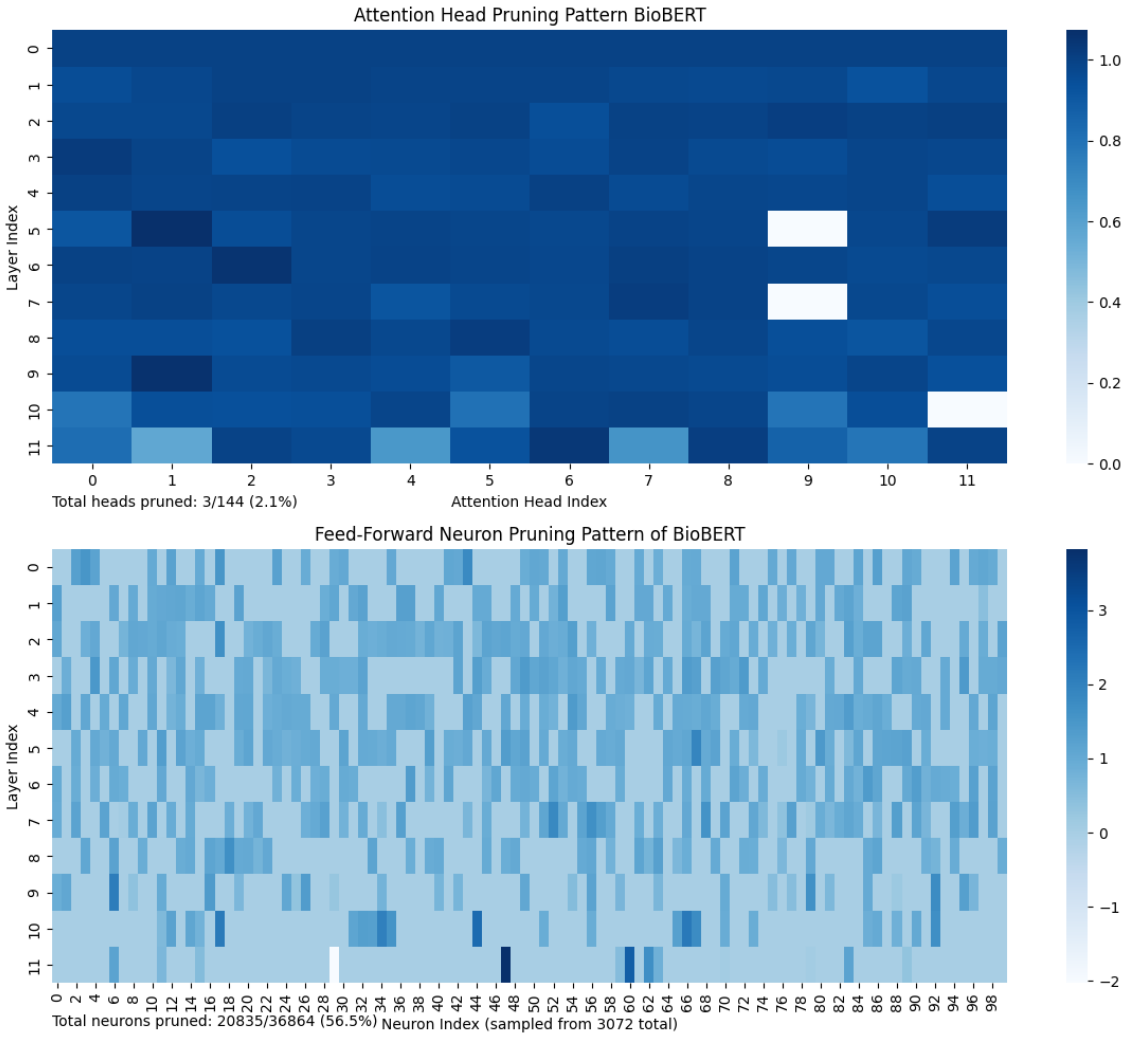

Figure A.1: Heat map of 30% of neurons and heads sparsity of pruned BioBERT

Figure A.2: Radar charts of BERT (above) and BioBERT (below). The standards model (grean), 30% pruned model (blue), and 30% pruned and quantized model (red).

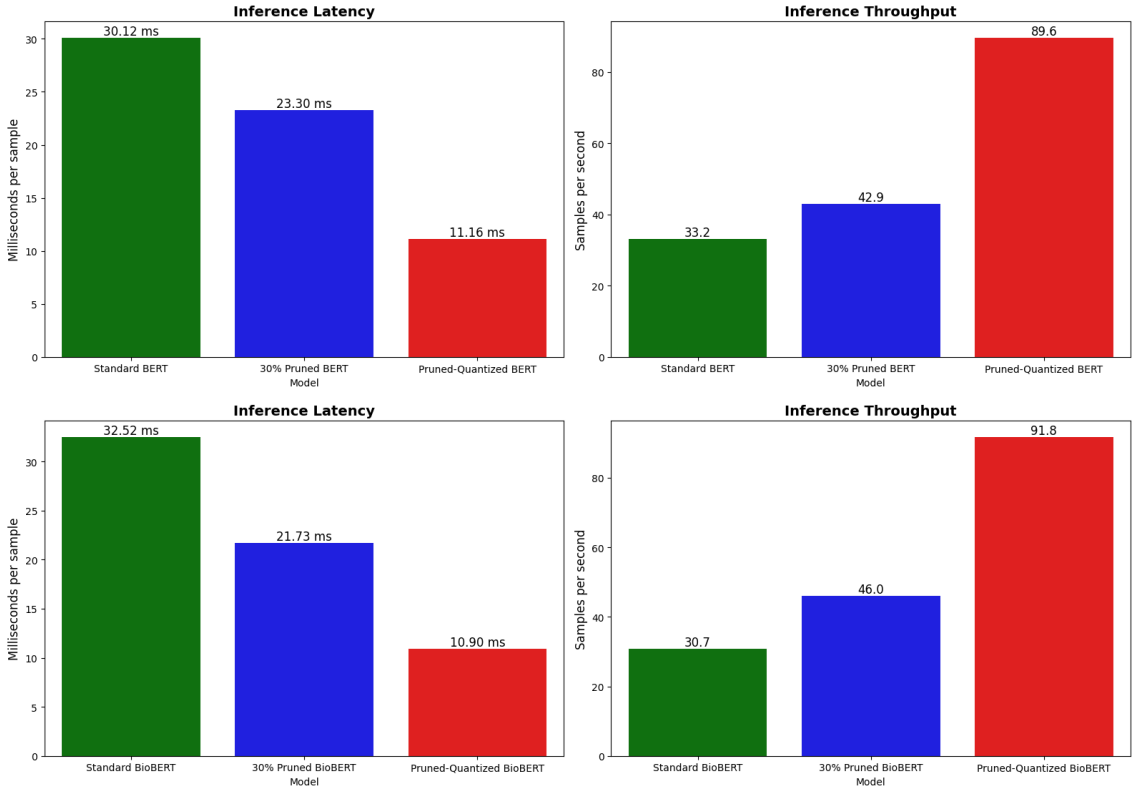

Figure A.3: Inference time of BERT (above) and BioBERT (below). The standards model (green), 30% pruned model (blue), and 30% pruned and quantized model (red).

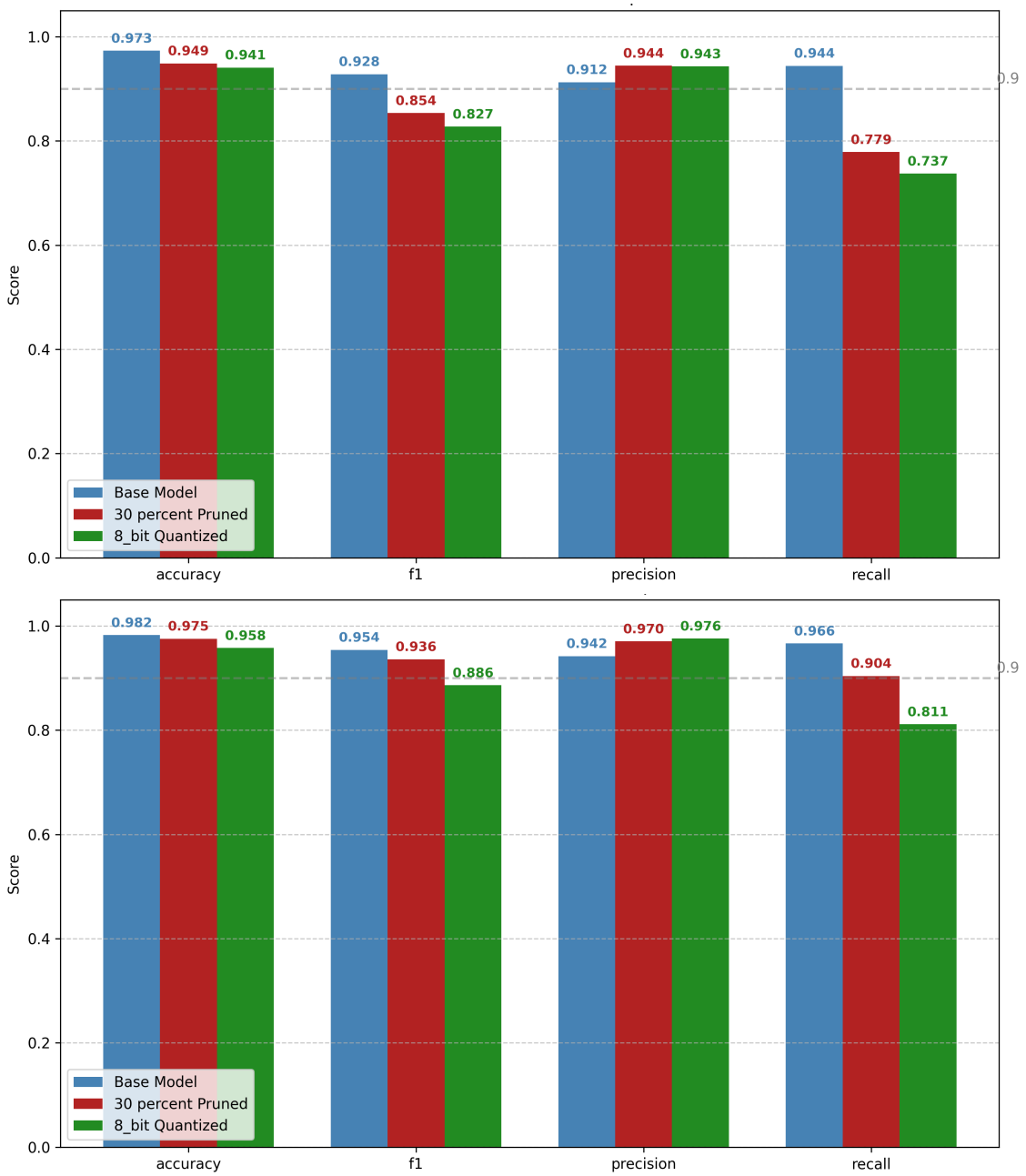

Figure A.4: Global performance metrics of the standard BERT (above) and BioBERT (below). The standard model (blue), 30% pruned model (red), and 30% pruned and quantized model (green).