LUT University

School of Engineering Science

Industrial Engineering and Management

BERT model optimization methods for inference – a comparative study of five alternative BERT-model implementations

Marko Buuri

1. Supervisor: Pasi Luukka
2. Supervisor: Jyrki Savolainen

1 Table of contents

TIIVISTELMÄ

Tekijä: Marko Buuri

Työn nimi: BERT-mallin optimointimenetelmät inferenssiin – vertaileva tutkimus viidestä vaihtoehtoisesta BERT-mallin toteutuksesta

Vuosi: 2022 **Paikka:** Espoo

Diplomityö. LUT-yliopisto, Tuotantotalous, Business Analytics

65 sivua, 5 taulukkoa, 5 kuvaa

Tarkastajat(t): Pasi Luukka, Jyrki Savolainen

Hakusanat: BERT, distillointi, kvantisointi, inferenssi

Aiempina vuosina on nähty Transfer Learning -lähestymistapojen nousu luonnollisen kielen käsittelyssä (NLP) ja laajamittaisista esikoulutetuista kielimalleista on tullut perustyökalu monissa NLP-tehtävissä. Vaikka suuret mallit johtavat yleensä merkittäviin parannuksiin, niissä on usein useita miljoonia parametreja, jotka voivat tuottaa haasteita.

BERT-mallit ovat Transformer-arkkitehtuurilla toteutettu kielenmallinnusjärjestelmiä, jotka ovat osoittautuneet tehokkaiksi kielimalleiksi. Tässä diplomityössä esitetään keinoja, jolla voidaan parantaa BERT-mallein nimetty kohteen tunnistamisen inferenssiä. Näillä menetelmillä voidaan hyödyntää pienetäkseen mallien kokoa ja parantaa niiden suoritustehoa pienentämättä merkittävästi mallin tarkkuutta. Kirjallisuuskatsauksessa selvitetään aikaisemmista tutkimuksista parhaita mahdollisia menetelmiä, joita on hyödynnetty mallin laskentatehokkuuden parantamiseen pienentämättä kuitenkaan niiden tarkkuutta.

Toteutuksessa viiden erilaisen BERT-mallin toimivuutta on testattuja CoNLL-2003 datalla. Mallien toimivuuksien tuloksia, etenkin F1-tulosta, joka mittaa mallin tarkkuutta, ja malleihin käytettyä aikaa on verrattu toisiinsa. Alkuperäisenä mallina käytettiin BERT-base mallia. Tulokset osoittavat testatuista BERT-malleista, että Distill RoBERTa onnistui suoriutumaan paremmin saavuttamalla F1-tuloksen 96.74 % ja puolet vähemmällä ajalla kuin alkuperäinen malli BERT-base F1-tuloksella 95.98 %. Parannuksia voidaan huomata tarkastelemalla F1-tuloksia ja käytettyä aikaa inferenssissä. mutta osa malleista, kuten DistillBERT, eivät tuottaneet parannuksia tarkkuudessa eikä ajassa lähtökohtaan kuten kirjallisuuskatsauksen perusteella oli odotettavissa.

ABSTRACT

Author: Marko Buuri

Title: BERT model optimization methods for inference – a comparative study of five alternative BERT-model implementations

Vuosi: 2022 **Paikka:** Espoo

Master's thesis. LUT University, Industrial Engineering and Management, Business Analytics
65 pages, 5 tables, 5 figures

Supervisors(s): Pasi Luukka, Jyrki Savolainen

Keywords: BERT, distillation, quantization, inference

Previous years have seen the rise of Transfer Learning approaches in Natural Language Processing (NLP) with large-scale pre-trained language models becoming a basic tool in many NLP tasks. Even though larger models generally lead to significant improvements, they often have several million parameters which can raise concerns.

BERT models are natural language processing models base on Transformer-architecture, which have been proven effective. The aim of this master thesis is to introduce different BERT base named entity recognition model inference optimization methods, which can be implemented to reduce the model's size and improve its throughput without compromising its accuracy. In literature review best possible methods from previous studies are find out, which have been used to increase models' computational efficiency and not reducing their accuracies.

In implementation five different BERT models were tested with CoNLL-2003 data and results show that part of these models were able to have better performance. Models we compared with each other especially concentrating on their F1-scores and total time used in inference. As a base model BERT-base was used. Results partly showed that the models' performance and accuracies for Distill RoBERTa achieved to perform better with F1-score of 96.74 % compared to the BERT-base with F1-score of 95.98 % and used half of time compared to the initial BERT-base model. Improvements can be seen by evaluating F1-scores and time used inference, but some of the models did not perform better like DistillBERT, compared to the base model as expected based on the literature.

ABBREVIATIONS

ABWR	Absolute Binary weight regularization
ACE	Automatic Content Extraction
ANN	Artificial Neural Network
BERT	Bidirectional Encoder of Representations from Transformers
CPU	Central processing unit
DNN	Deep neural networks
EDR	Entity Detection and Recognition Value
FLOP	Floating-point operations per second
GLUE	General Language Understanding Evaluation
GAN	Generative adversarial networks
GPU	Graphic processing units
ILF	Inverse Layer-wise Fine Tuning
MLM	Masked Language Modelling
NER	Named Entity Recognition
NLP	Natural language programming
NSP	Next sentence prediction
PT	Prioritized Training
SOP	Sentence order prediction
SQuAD	The Stanford Question Answering Dataset
TPU	Tensor processing unit

1. Introduction

In recent years, transfer learning approaches in natural language processing (NLP) have proliferated, with large-scale pre-trained language models becoming a key tool for many NLP tasks. . Although larger models generally result in significant improvements, they often have several million parameters that can raise various concerns. For example, the computational and storage requirements of these models can prevent widespread adoption. (Sanh et al., 2019, p. 1) Various model compression techniques have been developed to speed up model inference and reduce model size while maintaining accuracy. The most widely used techniques involve knowledge quantification and distillation. Many attempts have been made to distill heavy models into their lighter counterparts. (Liu et al., 2020, p. 1)

One idea in the Transformer architecture is to move away from sequential processing, where inputs are provided one at a time. Transformers intend to change this design by providing the entire sequence as a one-time input to the network, allowing the network to learn an entire sentence at a time. This enables parallel processing and enables parallel distribution of insights to other cores or graphic processing units (GPUs). The goal of the encoder layer is to convert all input sequences given to the model into a representation layer that captures the context in a way that also pays more attention to the words that matter most to them in a given context. (Jain, 2022, p. 21, 27)

The aim of this master thesis is to enhance bidirectional encoder representation from transformers named entity recognition (NER) model's inference by applying different optimization methods and comparing the effect on accuracy. By enhancement, the goal is to decrease the latency of the model's inference, decrease the model size in memory and increase the throughput on the central processing unit (CPU). (Kim et al., 2021, p. 1) NLP investigates the use of computers to process to understand human languages to perform useful tasks. It is an interdisciplinary field that combines computational linguistics, computing science, cognitive science, and artificial intelligence. (Jain, 2022, p. 2) Despite the most recent results on various NLP tasks, pre-trained Transformer models are generally an order of magnitude larger than previous models. For example, the large BERT model contains 340 million parameters, and in recent years larger transformer models have been introduced to contain even more parameters. Efficient implementation of these models has become a major challenge even in data centers due to limited resources such as performance, storage, space and computing power. In addition, these modelsrequire real-time inference. These challenges are more advanced devices where computing and power resources are more limited. (Kim et al., 2021, p. 1) Recent studies conducted by Sahn et. al. (2019) has also shown that pre-trained language models have redundancy, and therefore, it is crucial and feasible to reduce the

computational overhead and model storage while retaining performance. (Sanh et al., 2019, p. 1) Training large models from scratch typically takes four days on 4 to 16 Cloud tensor processing units (TPU)s, and even fine-tuning and pre-trained models with task-specific datasets may take several hours to finish one epoch. For these reasons, reducing computational costs is crucial for their applications in practice, where resources are limited. (Sun et al., 2019, p. 1)

Knowledge distillation aims to transfer the knowledge embedded in a large teacher network where the student network is trained to reproduce the behaviors of the teacher network (Jiao et al., 2019, p. 1). Quantization is a technique that compresses models into smaller ones by representing a parameter and/or enabling low bit precision, reducing memory consumption. floating point arithmetic. These methods are used, for example, in integer-only quantization approaches. Additionally, it should be noted that approaches using floating-point arithmetic are inferior in terms of latency and power efficiency compared to integer-only inference. (Kim et al., 2021, p. 1-2)

1.1 Motivation

The aim of this thesis is to produce information related to different BERT models' relative functionalities and compare 5 different BERT-based models named entity recognition models and develop my competencies related to natural language processing. From a professional point of view, this study gives me knowledge related to improving natural language models.

1.2 Research objectives and limitations

The purpose of this master thesis is to investigate different techniques for enhancing bidirectional encoder representation from transformers model's inference. The subject group of this thesis is people interested to know different techniques for improving BERT models inference cost-efficiency and decreasing the latency of the model. The research questions are as follows:

1. Which techniques can be used for decreasing BERT models' production costs inference recourses?
2. Which methods can be used for increasing the inference of the BERT model's throughput by not severely decreasing the accuracy of the model?

1.3 Data and Methodology

Methodologies used in this master thesis include investigating different possible techniques through previous studies made in this area. Sources included especially regarding distillation and quantization techniques are reached and described. Also, found possible techniques are used in practice to make the model more efficient. The data used in this study is called CoNLL-2003. (Tjong et al., 2003, p. 1-2)

1.4 Structure of the Thesis

The first part of the thesis consists of the theoretical background, where the Bidirectional Encoder of Representations from Transformers (BERT) is introduced. Next different model efficiency methods concentrating on distillations and quantization methods are described. After the theoretical background, different methods found for model inference optimization are presented in different academical articles.

In the empirical part data source used in the study is described and methods used for testing different models. Results are analyzed and conclusions are derived from the literature review and empirical results.

2 Theoretical Background

2.1 BERT - Bidirectional Encoder Representations from Transformers

Bidirectional Encoder Representation from Transformers (BERT) is applied to language modeling. It is a model introduced by researchers by Google and it is popular in a variety of NLP tasks like question answering. The model consists of two encoders for encoding sequences and it takes two sequences for encoding. One is the normal sequence and the other one is the reverse of it. The model makes it different from previous models where sequences are taken in one direction only, from left to right to right to left. (Sabharwal and Agrawal, 2021, p. 60)

The encoder consists of input embeddings, tokenization, vectorization, and positional encoding. One way to think of a word embedding layer is as a lookup table, which allows for the acquisition of a learned vector representation of each word, after which each word of a sentence is tokenized. After tokenization the tokens are vectorized, where each word is represented as a vector. Finally, positional encoding is done, which is based on the position of a specific word. Some information related to the positions in the input embeddings needs to be provided since the transformer encoder does not have recurrence as recurrent neural networks do. (Jain, 2022, p. 24-25)

The responsibility of a decoder is to produce text sequences. It is similar to encoders in having the layers like multi-headed attention layers, adds and norm layers, and feed-forward layers. In addition, it has a linear layer with a SoftMax classifier to emit probabilities of an output. The decoder takes the starting tokenized word and then previous outputs if any and combines them with the output of the encoder. The beginning of a decoder is like an encoder to large extent. The input is first placed via an embedding layer and then a positional encoding layer. The positional embeddings are sent through to the first multi-head attention layer. (Jain, 2022, p. 35)

The first Multi-headed Attention layer uses a lookahead mask to restrict the decoder from looking at tokens that are yet to come. The mask is included both before and after the SoftMax calculation. The idea of the mask is to calculate the attention score for the current word based on previous words and not for future words in the sentence. The second layer of Multi-headed Attention takes the output from the first layer of the decoder and combines it with the output of the encoder, which allows the decoder to understand better as to which components of the encoder output to attend to the output of this layer are passed via a feed-forward network. In the last step, the output of the previous layer and feed-forward network is again

normalized and passed to a linear layer with a SoftMax component for emitting probabilities. As an example, the probability would be the probability of what could be the next word in the sentence. (Jain, 2022, p. 35)

Pretrained BERT does not require any architectural change and can be used for different tasks by modifying the output layer. For grasping the relationship between a token or a word in the text it uses a transformer, which consists of an encoder and a decoder. BERT requires only the encoder mechanism. The encoder reads the input texts and can read the entire sequence of words at once instead of reading them sequentially from right to left, which makes the model bidirectional. The sequences of tokens are embedded into vectors, and they are used as input to the transformer. Then the vectors are processed in the neural network. As an output neural network gives a sequence of vectors corresponding to the input tokens, and it is dependent on the context in which it occurs. To surpass unidirectional constraints, BERT used two strategies: masked language modeling and next-sentence prediction. (Sabharwal and Agrawal, 2021, p. 60-61, 65)

Input embedding in BERT is a combination of three types of embedding: position embedding, segment embedding, and token embedding. Since order-related information is lost in transformers, position embeddings are used to learn the order information. BERT learns a unique position embedding for each position in the input stream, allowing BERT to express the position of words in a stream. In segment keying, BERT learns keystones unique to the first and second keystone to help the model distinguish between them. With segment embedding, BERT can use sentence pairs as input for tasks such as answering questions. In token embeddings, token embeddings are learned by using tokens in WordPiece token vocabulary. (Sabharwal and Agrawal, 2021, p. 66–67)

The vocabulary is initialized with individual characters in the language, then the most frequent combinations of symbols in the vocabulary are iteratively added to the vocabulary. The vocabulary inventory is initialized with all the characters in the text and the most frequent combinations of symbols in the vocabulary are added iteratively to the vocabulary. (Graves, 2012, p. 2) The vocabulary contains subworlds of words in their corpus. Summing the token, segment, and position embeddings of the input representation of a token given make it a comprehensive embedding scheme containing empirical useful information for the model. It was observed that WordPiece embeddings are designed to learn context-independent representations, whereas the hidden layer embeddings are designed to learn context-dependent representations. (Sabharwal and Agrawal, 2021, p. 66–67, 86)

Sequence transduction, sequence transformation is meant, where the input sequence is transformed into an output sequence for example in speech recognition, machine translation, and text-to-speech translation. (Graves, 2012, p. 1) One of the most competitive neural sequence transduction models has an encode-decoder structure. In BERT architecture encoder maps the input sequence of symbol representations to a sequence of continuous representations. Next, the sequence of continuous representation is passed to the decoder, which generates an output sequence of elements at a time. At each step, the model is autoregressive, which means that it uses the previously generated symbols as additional input information when generating the next. The transformer is using self-attention and pointwise, fully connected layers for the encoder and decoder. In the model, the architecture encoder is composed of 6 identical layers, which have two sublayers. (Vaswani et al., 2017, p. 2)

An Artificial Neural Network (ANN) is a biologically inspired computational model, which consists of processing elements called neurons, and connections between them with coefficients called weights, bound to connections. The feed-forward network is one of the most common neural network architectures in which connections between neurons are directed and going only in a global forward direction, avoiding the formation of feedback loops. (Shanmuganathan, 2016, p. 48) To perform multiclass classifications with reasonable results, the SoftMax activation function is used. The SoftMax function transforms a vector into another vector for real values, each between 0 and 1, that sum up to 1. The activation function works like a probability since the sum over vector values is 1 and its elements are all less than 1. Given real values z_k for $i = 1, \ldots, k$ the $z = (z_1, \ldots, z_k)$ the SoftMax vector is defined as follows. (Michelucci, 2022, p. 68)

$$S(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{e_j}} \quad (1)$$

The encoder is composed of a stack of six identical layers, each layer having two identical sub-layers. The first is a multi-head self-attention mechanism and the second is a position-wise fully connected feed-forward network. (Vaswani et al., 2017, p. 3) In the multi-head attention mechanism, the attention value represents the contribution to the classification results, to further optimize the final output of the model. It uses multiple parallel queries to extract multiple groups of different subspaces from features to obtain the relevant information, and self-attention feedbacks the internal dependence between the data and captures the key information of the sequence from different aspects. (Michelucci, 2022, p. 2)

Residual connection is hired round every of the two sublayers, accompanied with the aid of using layer normalization. The decoder is also composed of a stack of six same layers. The decoder inserts a third

sub-layer similarly to the two sub-layers in every encoder performing multi-head interest over the output of the encoder stack. Like the encoder, residual connections round every of the sub-layers are hired., accompanied with the aid of using layer normalization. Self-attention sublayer in the decoder stack is modified to prevent positions from attending to subsequent positions. This masking ensures that the prediction for position *i* can depend only on the known output at positions less than *i*. (Vaswani et al., 2017, p. 3)

An attention function can be described as mapping a query and a set of key-value pairs to an output. Here the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values. The weights assigned to each value are computed by a compatibility function of the query with the corresponding key. In Scaled Dot-Product Attention the input consists of queries and keys of dimensions d_k, and values of dimension d_v. The dot products of the query with all keys are computed, divided by d_k, and the SoftMax function is applied to obtain the weight of the values. (Vaswani et al., 2017, p. 3)

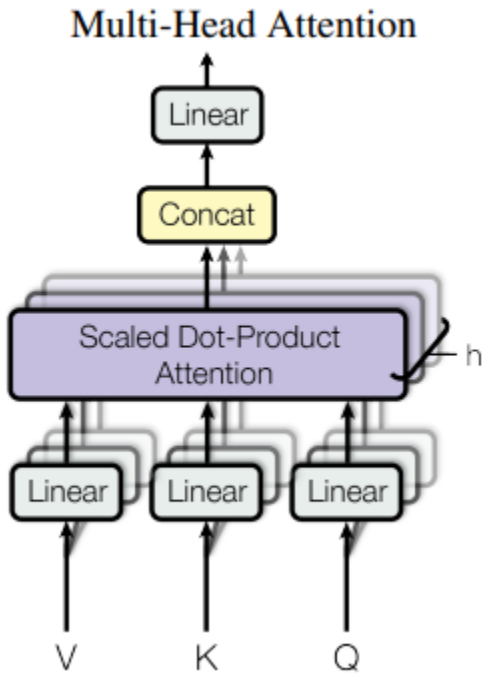

Figure 1. Multi-Head Attention (Vaswani et al., 2017, p. 4)

The attention function on a set of queries is computed simultaneously, packed into a matrix *Q*. The keys and values are also packed together into matrices *K* and *V*, and the matrix of outputs is computed with the following equation. (Vaswani et al., 2017, p. 4)

$$Attention(Q, K, V) = \frac{QK^T}{\sqrt{d_k}}V \quad (2)$$

The most used attentions are additive and dot-product attention. Dot-product is identical to the above-described algorithm, except for the scaling factor. Even though the two are similar in theoretical complexity, dot-product attention is much faster and more space-efficient since it can be implemented using highly optimized matrix multiplication code. (Vaswani et al., 2017, p. 4) Multi-Head attention is visualized in Figure 1.

It was found beneficial by Vaswani et. al. (2017) to linearly project the queries, keys, and values hidden state times with different, learned linear projections to d_k, d_k and d_v dimensions, respectively. On each of the projected entities, the attention function is performed in parallel, resulting in the final values. With Multi-head attention, it was found that the attention allows the model to jointly attend to information from different representation subspaces at different positions. (Vaswani et al., 2017, p. 4-5)

The Transformer uses multi-headed attention in three different ways. In the encoder-decoder attention layer, queries come from the decoder layer above, and storage keys and values come from the encoder output. The encoder contains self-service layers where all keys, values and queries come from the same place, while the self-service layers in the decoder allow any position in the decoder to serve all positions in the decoder up to and including that position. Each of the layers contains a fully connected feedback network that is applied to each position separately and identically. The learned embeddings are used to convert the input and output tokens to vectors. The decoder also outputs the learned transform and the SoftMax function for converting the probabilities of the next prediction token. Since neither recursion nor convolution is used, some information about the relative or absolute positions of the tokens in the sequence must be included in order to take advantage of the sequence order. Positional encodings are added to the input embeddings at the end of the encoder and decoder stacks. (Vaswani et al., 2017, p. 5-6) Transformer model architecture is pictured in Figure 2.

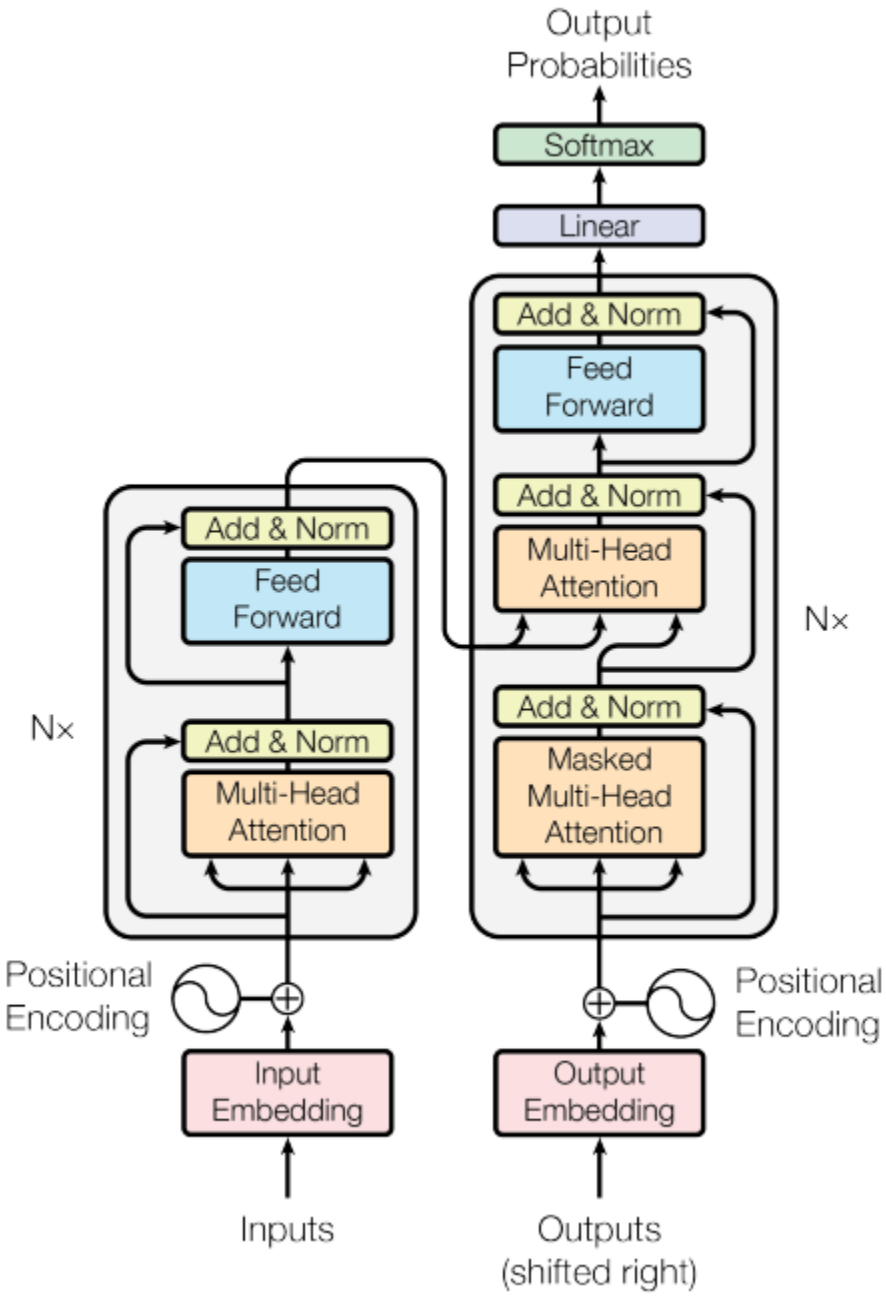

Figure 2. Transformer model architecture (Vaswani et. al., 2017, p. 3)

Two BERT models have been implemented; the BERT base model and the BERT large model. The BERT base model is a pretrained BERT model that has 12 layers of transformer block, 768 hidden units in each layer, and 110 million parameters. Model can be further classified as BERT base-cased and BERT base-uncased depending on the text. BERT large model has 24 layers, 1024 hidden units in each layer, and 340 million parameters. It can be also further classified as BERT large-cased and BERT large-uncased. (Sabharwal and Agrawal, 2021, p. 62-63)

2.2 Masked language modeling

Masked language modeling is used to surpass the BERT model's unidirectional constraint. The purpose is to assist the bidirectional transformer by masking randomly tokens from the input test while the next sentence prediction task jointly pre-trains text pair representations. The goal is to minimize the combined loss function for both tasks during training. (Sabharwal and Agrawal, 2021, p. 61)

To predict a masked word, words surrounding it are used to predict the masked word. 15% of words are masked when sequences are fed into BERT. Three different mask strategies are used. 80% of the masked words are replaced with a [MASK] token, 10% are replaced with random words, and 10% of the time the words are unchanged. The purpose is to bias the representation of the actual observed words is done because if 100% of the masked words were used then the model would not necessarily produce good token representations for non-masked words. This improves the model's performance since too much focus on a particular token or position has been prevented. Three steps need to be followed to generate a word embedding using BERT: adding a classification layer on top of the encoder output, multiplication of the output vectors by the embedding matrix, and the calculation of the probability of each word in the vocabulary with SoftMax. Masked values are only considered by the loss function in the prediction, and the non-masked words are ignored. (Sabharwal and Agrawal, 2021, p. 68)

2.3 Next Sentence prediction

Next sentence prediction is applied in order BERT model to understand how different sentences in a text corpus are related to each other. For the training, the sentence pairs are taken as input. The goal is to predict if the second sentence in the pair is the previous sentence in the original input text. In 50% of inputs second sentence is the subsequent sentence as in the original text and in the other 50% of the pairs, the second sentence is chosen randomly from the text. The model assumes that the random second sentence is disconnected from the first sentence. (Sabharwal and Agrawal, 2021, p. 69)

Prior to training, inputs are processed. The process consists of three phases. The first two tokens are inserted into a sentence pair. One is at the beginning of a sentence, and one is at the end of a sentence. Both sets are tokenized, and the use of delimiters separates them. They are then entered into the model as a single input. In the second phase, an embedding is added for each symbolic sentence, indicating whether the sentence is the first or the second sentence. In the last phase before training, position embeddings are added to each token, which help the model to indicate the token's position in the sequence. (Sabharwal and Agrawal, 2021, p. 70)

To predict the correct class for sentence pairs three steps are performed. The input sequence is passed through the transformer model. Next using a classification layer, the output of the first sentence's token is transformed into a 2x1-shaped vector and finally, the probability is computed with SoftMax. (Sabharwal and Agrawal, 2021, p 70-71)

2.4 Benchmarks for BERT models

The performance and accuracy the of BERT model have been evaluated several times over different types of datasets for various NLP tasks. This is being done to check if BERT can achieve benchmark values already set up for these datasets. These datasets evaluate the working of specific aspects of a model and the most common benchmarks are discussed next. (Sabharwal and Agrawal, 2021, p. 83)

2.4.1 GLUE Benchmark

General Language Understanding Evaluation (GLUE) is a collection of datasets that can be used to train, test, and analyze NLP models. These different models are compared with each other by the GLUE dataset. The GLUE benchmark includes nine different datasets. To evaluate a model, it is first trained over a dataset provided by GLUE, and then it is scored for the nine tasks. The final performance score is the average of all the nine tasks. (Sabharwal and Agrawal, 2021, p. 83)

$$Final\ GLUE\ Score = \sum Individual\ Task\ score \quad (3)$$

There is no need to change the input layer because the layer accommodates all of the GLUE tasks. However, the pretraining classification layer has to be removed. The BERT model scores a state-of-the-art result on the GLUE benchmark, with a score of 80.5%. (Sabharwal and Agrawal, 2021, p. 83)

2.4.2 SQuAD dataset

The Stanford Question Answering Dataset (SQuAD) is a reading comprehension dataset, consisting of questions asked on a series of Wikipedia articles. The answer to each of the questions is either a text segment or a span from the passage. There are two versions of the SQuAD dataset 1.1 and 2.0. SQuAD 1.1 consists of 50 000 unanswered questions and SQuAD 2.0 of 100 000 answered questions. The questions are similar in both datasets. The BERT model can achieve an F1-score of 93.2 and 83.1 for SQuAD 1.1 and SQuAD 2.0 over the test dataset.(Sabharwal and Agrawal, 2021, p. 84)

2.4.3 IMBD Reviews dataset

The IMBD dataset is a film review dataset that was used to classify viewers' opinions of films. The dataset consists of 25,000 reviews for testing. In addition to the training and testing data, there is unlabeled data, and the dataset was also used to assess BERT in a sentiment ranking task. (Sabharwal and Agrawal, 2021, p. 84)

2.4.4 RACE Benchmark

RACE is a large reading comprehension dataset from the examination. he RACE dataset is used to evaluate models in a reading comprehension task. The data set comes from the English tests of Chinese students. Dataset consists of nearly 28 000 passages and 100 000 questions generated by human experts. BERT large model achieves a score of 73.8 % on the RACE benchmark dataset. (Sabharwal and Agrawal, 2021, p. 85)

2.5 Types of BERT models

Various models have been developed based. Different variants have been developed to cater to different types of NLP systems. In this section, four different variants are introduced. The variants covered in this section are:

- ALBERT
- RoBERT
- DistillBERT
- StructBERT

2.5.1 ALBERT

The ALBERT model was developed jointly by Google Research and the Toyota Technologies Institute. It is a smaller and smarter "lite" version of BERT that can be used with less processing power compared to BERT, but at the expense of some accuracy. Both BERT and ALBERT share a similar core architecture. ALBERT has a Transformer encoder and a vocabulary of 30,000 words equal to BERT's. However, some architectural improvements have been made. Whereas in BERT the embed size of the WordPiece is bound to the size of the hidden layer, in ALBERT the two parameters are not bound and the embed parameters are split into two smaller arrays. In ALBERT, the one-hot vectors are not projected directly onto the hidden layer, but are instead projected onto a smaller, lower-dimensional matrix, which is then projected onto the hidden layers. (Sabharwal and Agrawal, 2021, p. 86)

Parameter efficiency is improved by sharing all the parameters across all layers. The feed-forward and attention parameters are all shared, which helps stabilize network parameters. Also compared to BERT, ALBERT does not use Next sentence prediction (NSP). Instead, it uses its developed training method called sentence order prediction (SOP). It is used to model inter-sentence coherence loss, whereas BERT combines topic prediction with coherence prediction. In benchmarks, ALBERT has outperformed BERT. (Sabharwal and Agrawal, 2021, p. 87)

2.5.2 RoBERT

Facebook's artificial intelligence team has developed robust optimized BERT (RoBERT), and it is a streamlined method for pre-training NLP systems. The method re-implements the neural network architecture with additional pre-training improvements. While BERT has about 30,000 subwords, Robert has about 50,000 subwords. Compared to BERT, RoBERT uses more training data and more iterations. Static masking is applied in BERT, escaping words from the sentence during pre-processing. RoBERT applies dynamic masking, generating a masking pattern each time a sentence is entered into the training. The training is duplicated ten times and the data is masked differently. (Sabharwal and Agrawal, 2021, p. 88)

This has improved the performance of BERT-based models giving better results than static masking. Also, the training objective differs from BERT. In BERT the relationship between the sentences is captured by training on NSP. Experiments have shown that models trained without NSP performed better on several BERT benchmarks. Training on longer sequences has achieved better results. (Sabharwal and Agrawal, 2021, p. 90)

2.5.3 DistillBERT

DistillBERT was introduced for the knowledge distillation required to solve the problem of calculating a large number of parameters. Some NLP models can reach up to ten billion parameters. While this ensures optimal performance, it prevents training and maintaining the model with limited computational resources. In knowledge distillation, a larger model acts as a teacher to a smaller one that seeks to replicate its findings and underlayer activation, also known as teacher-student learning. To generalize the student model, the teacher's performance distribution can be used for all possible goals. (Sabharwal and Agrawal, 2021, p. 90)

Distillation loss considers the combination of the output probabilities of the teacher (t) and the student (s), and the teacher probabilities are calculated through temperature SoftMax. Compared to SoftMax, temperature SoftMax gives a smoother output distribution, where the size of larger probabilities is decreased, and the smaller ones are increased. (Sabharwal and Agrawal, 2021, p. 91)

To build a better model, the cosine embedding loss is used as a measure of the distance between the hidden representation for the teacher and the student. In DistillBERT, the loss is the same as that used in the BERT

model to predict the correct token value for the masked token in the stream. Similar to the BERT model, the DistillBERT network architecture is a transformative encoder model as a BERT base. However, with 66 million parameters, DistillBERT has half the number of layers compared to BERT's,110 million parameters, which helps reduce computational complexity when the computing environment is limited. On the GLUE benchmark, DistillBERT can achieve 97% of the BERT-base score. (Sabharwal and Agrawal, 2021, p. 91-92)

2.5.4 StructBERT

StructBERT integrates language structures into BERT pretraining with two linearization strategies, namely word-level order and sentence-level order. By including structural pre-training, StructBERT achieves better generalizability and adaptability. In StructBERT, the ability of the MLM task is increased by shuffling a certain number of tiles after masking words and predicting the right order. By randomly changing the order of sentences, StructBERT can also better understand the relationship between sentences. After pre-training StructBERT, the model can be referenced to task-specific data for a variety of downstream tasks, such as the summary of documents. (Sabharwal and Agrawal, 2021)

The BERT base model does not explicitly model sequential order or higher-order word dependency in natural language, while StructBERT can implement this by supplementing BERT training targets with new structural targets of words. This is done in conjunction with Masked Language Modeling (MLM) and gives the model the ability to restructure the sentence to get the correct order of randomly shuffled word tiles. In addition, the goal of the original NSP-BERT model is extended by not only predicting the set as well as the previous set, which allowed StructBERT to learn the sequential order of the set in a bi-directional way. (Sabharwal and Agrawal, 2021, p. 93-94)

2.6 Named entity recognition

Named entity recognition is a technique for identifying and classifying named entities in text. The result is the identification of different categories of words. These categories can be, for example, people, nationalities, religious or political groups, buildings, companies, and countries. Agrawal and Sabharwal (2021) states that Named Entity Recognition (NER) plays an important role in search engines and conversational systems. Search engines are used to identify documents relevant to a query made by a user, giving more importance to documents containing entities used in a search. Entities are used in

conversational systems to make a question asked by the user unique when the question relates to general problems, but for different entities. (Sabharwal and Agrawal, 2021, p. 482)

Marrero et. al. (2013) state that NER is widely used for example in semantic annotation, question answering, ontology population, and opinion mining. The term was first used at the 6th Message Understanding conference in 1996. NER is one the areas which purports to identify the semantics of interest in unstructured text to add a structure, which is one of the goals serving as the basis for other areas to manage information. (Marrero et al., 2013, p. 482)

Semantic annotation goes beyond textual annotations about the concept of the documents to the formal identification of concepts and their relations. These annotations bring two main benefits enhanced information retrieval and improved interoperability. For example, semantic annotation might relate a city name to the concept a of *City* linking the instance to a specific country name of the abstract concept of a *Country*. NER techniques are widely used in question-answering systems as means to facilitate the selection of answers. Ontologies play a key role in the semantic web and all the applications it supports depend on technology to make information interoperable. One of the cornerstones is the proliferation of ontologies, which aims to incorporate instances into existing ontologies. One of the pre-processing techniques for opinion mining is the recognition of named entities of interest. From these entities opinions can be identified and assessed as positive or negative. (Marrero et al., 2013, p. 483)

Testing five different NER tools by Marrero et. al. (2013) all seem implicit to recognize the categories of people, organization, and localization as types of named entities. Less frequently recognized entities were food products, natural elements, and names of events like wars. Only one system considered categories such as currency, dates, and measurements, while other tools do not recognize them or classify them in a category named miscellaneous, unknown, or other different definitions are given for named entities. Marrero et. al. (2013) categorizes them in terms of the following four criteria: grammatical category, rigid designation, unique identification, and domain of application. (Marrero et al., 2013, p. 486)

NER tasks can be divided into two subtasks: entity detection and entity classification. Entity detection aims to detect whether a word string in a given text is an entity and in entity classification tasks the aim is to judge a category of the detected entities. (Zong et al., 2021, p. 230)

Early research in NER mostly focused on rule-based methods and was commonly used among regular expressions. For example, in English, person names usually start with capital letters followed by titles such as Mr., Dr., or Prof. For example, databases have been constructed of location and organization names for

NER purposes. Nevertheless, with large databases, rule-based NER methods still face many challenges. On the one hand, a phrase might lead to different types of entities. As an example, *Washington* can be either a person's name or a location name. Also, common words can be a type of entity. For example, *Bill* could be an ordinary word or a name. Challenges might also occur with abbreviations. This means that rule-based methods enhanced with entity databases are difficult to handle and cannot obtain high recognition accuracy. The rule-based approach faces the problem of system maintenance. It requires constant modification or addition of new rules that may conflict with existing ones. (Zong et al., 2021, p. 230-231)

Supervised NER systems attempt to design machine learning methods that learn automatic prediction models on correctly labeled training data. This method is usually regarded in research as a sequence labeling problem. In sequence labeling, the first step is to determine the label set and language granularity for labeling. For example, BIO is a widely used label set. "B" denotes the beginning of an entity, "I" indicates the middle or end of an entity, and "O" denoted the outside of an entity. (Zong et al., 2021, p. 232)

Supervised NER methods can achieve acceptable performance given large amounts of annotated data. In practice, however, the corpus of annotations on named entities is limited. Some areas may not be covered, e.g., financial areas. Also, the training sets contain only about 100,000 sets, which leads to limitations in NER performance, especially in domain matching. There are also massive untagged corpora in different languages and regions. In view of these shortcomings, research resorts to semi-supervised NER methods. In semi-supervised methods, unlabeled data can be used in a variety of ways. Unlabeled data can be used to extract more features based on the similarity of speech units, or they can be used to extract different contextual patterns. The similarity of distributed language units can be used to exploit more features in large unlabeled data to discover effective features. For example, the words *say* and *tell* can be grouped into a group provided that the group can be used as a feature word, say appears in the annotated corpus and tell in unannotated data, tell and its context can be used to correctly predict the name Identities. (Zong et al., 2021, p. 239-240)

Mining the diversity of context patterns is another purpose for using NER. Representative samples with high confidence and low redundancy can be selected from the unlabeled data and treated as labeled samples to enlarge the supervised training data. This method cannot effectively improve performance because samples with high confidence base on the same context pattern as the training samples. For NER, this results in the newly added instances failing to enrich the context features. (Zong et al., 2021, p. 240)

2.7 NER model evaluation

NER methods are usually evaluated objectively. First step is to select the test set, which not overlaps with the training data. The test set is selected and manually labeled with entities. These entities include entities such as person, location, and organization names according to the specification used for the training data. If one method automatically recognizes the named entities in the test data obtaining the system output, the performance can be calculated by comparing the output to the reference. (Zong et al., 2021, p. 241)

The calculation process includes three variables: count(*correct*), count(*spurious*) and count(*missing*). Count(*correct*) is the number of entities correctly recognized in the system output, that is the overlap between system output and the reference. Count(*spurious*) refers to the number of entities recognized in the system output, but not considered as named entities in the reference. Count(*Missing*) refers to the number of named entities that exist in the reference that are not recognized by the system in the output. Based on these three variables, the precision, recall and F1 can be calculated. (Zong et al., 2021, 241)

$$precision = \frac{count(correct)}{count(correct)+count(spurious)} * 100\% \quad (4)$$

$$recall = \frac{count(correct)}{count(correct)+count(missing)} * 100\% \quad (5)$$

$$FI = \frac{2*precision*recall}{precision+recall} * 100\% \quad (6)$$

2.8 BERT model evaluation

In NER, systems are usually evaluated based on how their output compares with the output of human linguists 17. NER system can have different error types like system miss hypothesized an entity where there is none, entity is completely missed, a wrong label was given to an entity, system noticed there is an entity but got its boundaries wrong and a combination of two previous error types. (Sekine & Ranchhod, 2009, p. 18)

2.8.1 MUC score

In MUC a system is evaluated on two axes; its ability to find the correct type (TYPE) and its ability to find exact text (TEXT). When an entity is assigned the correct TYPE, a correct TYPE is credited and correct TEXT is credited if entity boundaries are correct, regardless of the TYPE. Or type and text three measures are kept: the number of correct answers (COR), the number of actual system guesses (ACT) and the number of possible entities in the solution (POS). Final MUC score is a micro-averaged f-measure (MAF), which is the harmonic mean of precision and recall calculated over all entity slots on both axes. The harmonic mean of two numbers is never higher than the geometrical mean. It tends toward the least number, minimizing the impact of large outliers and maximizing the impact of small ones. Precision is calculated by dividing COR with ACT, COR / ACT, and recall is calculated by dividing COR with POS, COR / POS. (Sekine & Ranchhod, 2009, p. 19)

2.8.2 Exact-match evaluation

Models are compared based on the MAF with the precision being the percentage of named entities found by the system that are correct and the recall being the percentage of named entities present in the solution that are found by the system. If named entity is an exact match with the corresponding entity in the solution, it is considered as correct. For some application the constraint of exact match is unnecessarily stringent. (Sekine & Ranchhod, 2009, p. 19-20)

2.8.3 Automatic Content Extraction - ACE evaluation

Automatic Content Extraction (ACE) evaluation includes mechanisms for dealing various evaluation issues like partial match and wrong type. It is more elaborated compared to previous tasks at level of named entity “subtypes”, “class” as well as entity mentions. In ACE evaluation each entity has a parametrized weight and contributes up to maximal proportion (MAXVAL) of the final score. For example, if each person is worth one point and each organization is worth 0.5 point then it takes two organizations to counterbalance one person in final score. In addition, for false alarms, missed entities and type errors customizable costs (COST) are used. Partial matches of textual spans are only allowed if named entity head matches on at least given proportion of characters, and temporal expressions are not treated in ACE. (Sekine & Ranchhod, 2009, p. 20)

Final score in particular evaluation method is called Entity Detection and Recognition Value (EDR), which is 100 % minus the penalties. ACE evaluation scheme cost of error is very customizable, and it covers wide range of problems. However, scheme is problematic because the final scores are only comparable when parameters are fixed, and complex methods are not intuitive and make error analysis difficult. (Sekine & Ranchhod, 2009, p. 21)

2.9 Cost efficiency of implemented models

The power of Transformers has been pushed to new levels with recent success in language models, leading to new achievements for example in natural language processing. It also has been observed that with more floating-point operations per second (FLOP)s the performance of pretrained Transformer models keeps consistently improving. On the other hand, it is extremely expensive to pretrain and finetunes the state-of-the-art models as they require more compared to traditional models. Due to these challenges, there has been a number of efforts to reduce the cost of pretraining and finetuning self-attention models. Typical approaches to tackle these issues are distillation, pruning, and quantization, where the aim is to derive a lighter model from a well-pre-trained model by learning to remove less important operations and taking the advantage of richer signals in the larger model. There are also aims to design an architecture that not only has a lower resources-to-performance ratio but also scales as well as the Transformer in certain domains. (Dai et al., 2020, p. 1)

For many sequence-level NLP tasks, the most common use case is to extract a single vector from the entire sequence, which does not necessarily preserve all information down to the token-level granularity. For example, in a study conducted by Galindez Olascoaga et. al. (2021) authors propose a sequential resolution of the hidden representation in self-attention models to reduce sequence length which can lead to saving in FLOPs and in memory, which can be re-invested in constructing a deeper model to boost the model capacity without additional computational burden.

Size and computation constraints also limit the amount of available memory and compute power, and these capabilities are in stark contrast with the state-of-the-art machine learning implementations, the inference stage requiring a vast number of computations per second and gigabytes of storage space. These demanding workloads are not attainable at the extreme, where devices are equipped with embedded CPUs that can perform at ranges lower than ten Giga operations per second and include very small memory. Consequently, the most extreme-edge realization of machine learning is currently located in the realm of deep neural

networks (DNN)s, both from an algorithmic and a hardware point of view. One can identify the following trends in pursuit of efficient inference on DNNs:

- A Strategy for modifying and pruning the model topology, with the goal of making the model as compact as possible by removing redundant connections and weight and exploiting weight and structural scarcity.
- Parameter quantization, parameter reduction, which improves the saving of computing resources.
- Specialized Hardware where DNNs lend themselves to parallelization because of their layered and uniform structures. For example, the use of graphics processing units (GPUs) has exploited parallelization capabilities and other properties.
- Memory-based strategies where reduction of such transactions aims to exploit the properties of memory cells to perform local computations before performing unnecessary transactions. (Galindez Olascoaga et al., 2021, p. 10-11)

The impressive performance of BERT comes with a heavy computing and memory cost, which makes on-device inference prohibitive. Most significantly, the BERT base model consumes a staggering 432 MB of memory in a 32-bit floating-point (FP32). Therefore, with limited resources is challenging and requires tight co-design of the BERT model optimizations with dedicated hardware acceleration and memory system design. For example, an early exit mechanism has been proposed to reduce the average energy and latency. The early existing entropy is probabilistic of the classification confidence, which is evaluated at the output of each computed Transform layer, and the inference exits when the entropy value falls below a pre-defined threshold. While this approach can appreciably reduce computation and energy costs, the achieved latency can vary drastically from one input sentence to another, potentially violating the strict real-time latency constraint of the application. (Tambe et al., 2021, p. 831)

3 Methods considered in this study

3.1.1 Distillation

Distillation is a process through which a large and accurate model transfers its knowledge to a smaller model with less representational power. The process consists of two steps. In the first step, a large teacher is trained on gold labels and in the second step a smaller student is trained on the labels produced by the teacher, also known as soft labels. The idea in knowledge distillation is that a larger model acts as a teacher for a smaller model. The smaller model tries to replicate the teacher's outputs and sublayer activation for a given set of inputs, which is also known as teacher-student learning. Applying the distribution information from the teacher's model helps in the creation of student models for a different purpose. Since soft labels carry additional information distillation outperforms standard training. (Sabharwal and Agrawal, 2021, p. 90-91)

Due to the drawbacks of a large number of parameters compression techniques like knowledge distillation were introduced in DistillBERT model. Recently developed NLP models show an increase in parameter count, which prevents model training and serving due to limited computational resources. (Sabharwal and Agrawal, 2021, p. 90-91)

Go et. al. (2021) discusses three different categories of knowledge distillation: response-based knowledge, feature base knowledge, and relation base knowledge. In response knowledge, the idea is to mimic the prediction of the teacher model by referring to the neural response of the last output layer of the teacher model. Soft targets are known as the most popular response-based knowledge for image classification. As a defect, this distillation category fails to address the intermediate-level supervision from the teacher model. (Gou et al., 2021, p. 4-5)

A good extension of response-based knowledge and feature-based knowledge from the intermediate layers. So-called feature maps, the output of the last layer, and the output of the intermediate layers can be used as knowledge to supervise the training of the student model. The main idea is to directly match the feature activations of the teacher model and the student. (Gou et al., 2021, p. 5)

In relation-based knowledge distillation, different layers of data samples are further explored. For example, a flow of solution process is proposed, which is defined by a Gram matrix between two layers. The matrix summarizes the relations between pairs of feature maps, which are calculated using the inner products between features from two layers. Via singular value decompositions using the correlations between feature

maps as the distilled knowledge was proposed to extract information in the feature maps. (Gou et al., 2021, p .6)

Distillation schemes, so-called training schemes can be divided into three main categories: offline distillation, online distillation, and self-distillation. In offline distillation the training process consists of two stages: the teacher model is trained on a set of training samples and in the form of logits the intermediate features teacher model is used to extract the knowledge. The scheme usually employs one-way knowledge transfer and two-phase training procedures. (Gou et al., 2021, p. 8)

To improve the performance of the student model online distillation is proposed, when a large-capacity high-performance teacher model is not available. In online distillation, the teacher model and the student model are updated simultaneously. To reduce computational cost multi-branch architecture was proposed, in which each branch indicates a student model and different branches share the same backbone network. In a distillation method co-distillation, multiple models with the same architectures are trained in parallel and any other model is trained by transferring the knowledge from one to another. An online scheme is a one-phase end-to-end training scheme with efficient parallel computing. However, usually existing online methods fail to address the high-capacity teacher online setting making it more interesting to explore. (Gou et al., 2021, p. 8-9)

Self-distillation is a special case of online distillation that uses the same networks for the teacher and student models. A new method of self-distillation has been proposed, in which knowledge is distilled from the deeper sections of the lattice into its shallow sections. A special variant of auto-distillation is snapshot distillation, in which findings from earlier epochs (teacher) are transferred to later epochs in another network (student). This is done to support's supervised training process within the same network. To further reduce the inference time using distillation-based early output training, it has been proposed that the early output layer attempts to mimic the output of the later output layer during training. (Gou et al., 2021, p. 9)

In short online distillation means the teacher teaches the student, in online distillation both teacher and the student study together with each other, and in self-distillation student learns knowledge by himself. Different distillation schemes can be combined to complement each other. As an example, both self-distillation and online distillation are integrated via the so-called multiple knowledge transfer framework. (Gou et al., 2021, p. 9)

Next different distillation algorithms are presented. Algorithms, which are discussed are adversarial, cross-modal, graph-based, attention-based, data-free, quantized, lifelong, and NAS-based distillations.

Many competing methods of knowledge distillation have been proposed to provide teacher and student networks with a better understanding of data distribution. In generative antagonistic networks (GANs),

the discriminator estimates the probability that a sample comes from the training data distribution, while the generator tries to fool the discriminator with generated data samples. Methods using GAN can be divided into three main categories. In the first category, the synthetic data is generated with an adversarial generator used as a training data set or to augment the training data set. In the second category, a discriminator is introduced to distinguish the student and teacher model samples through the use of logits or features. In the last category, the distillation of contradictory knowledge is performed online, with the student and the teacher optimizing together in each iteration. (Gou et al., 2021, p. 11)

The use of logit averaging, and functional representation has been shown to be effective in training students across multiple teacher networks. Two networks of teachers can be used to use logit and intermediate functions, in which one teacher transmits response-based knowledge to the student and the other teacher transmits function-based knowledge to the student. Various methods have been proposed to simulate multiple teachers by adding types of sounds to a specific teacher to realize knowledge transfer and explore the power of teachers by using multiple teachers, the distillation of knowledge from multiple teachers can be rich Deliver knowledge and adapt a versatile learning model more efficiently. (Gou et al., 2021, p. 12)

The reason that data or labels are not available in some modalities it is, important to transfer knowledge between different modalities using cross-modal distillation algorithms. The methods can rely on unlabeled paired samples involving different modalities, for example, RGB images and depth images. Knowledge distillation performs well in visual recognition tasks in cross-modal scenarios. (Gou et al., 2021, p. 13)

Graph-based distillation algorithms examine relationships between data, while most knowledge distillation algorithms focus on transferring knowledge from individual instances from teacher to student. There are two main ideas behind these methods; Using the chart as a carrier of the teacher's knowledge or using the chart to control the delivery of the teacher's knowledge message. In a study, a distillation diagram is introduced to examine the relationships between different modalities. The vertices represent a modality and the edges indicate the strength of the connection between one modality and another. This kind of algorithms can transfer the knowledge about the informative structure of the data. (Gou et al., 2021, p. 14)

In data-free distillation algorithms as the name implies no training data exists and the data is newly or synthetically generated. In some studies, the transferred data is generated by a GAN. In one of the proposed methods, the transferred data is reconstructed by using the layer activations or layer spectral activations of the teacher network. In one of the studies zero-shot, knowledge distillation was proposed that does not use existing data. Using the parameters of the teacher model the transferred data is produced modeling the SoftMax space. Data-free distillation has shown potential under the condition of unavailable data when the

data can be generated from the feature representations from the pre-trained teacher model. (Gou et al., 2021, p. 15)

In some knowledge distillation methods, quantification processes in the teacher-student framework have been proposed. For example, a quantified distillation method, in which knowledge is transferred to a quantified network of higher-weight students, has been proposed. In a study, knowledge is transferred from a high-precision network of teachers to a small, low-precision network. The master in the feature maps is first quantized and knowledge is transferred from the quantized master to a quantized student network. In recent studies, a self-distillation training scheme was developed to improve the performance of quantized depth models. In these models, the teacher shares the parameters of the student model. (Gou et al., 2021, p. 15)

The distillation method of lifelong learning includes three different learning methods: continual learning, continuous learning, and meta-learning. The aim is to accumulate previously learned knowledge and transfer the knowledge to future learning. 15 For example meta-transfer networks have been proposed that can determine what and where to transfer in the teacher-student architecture. Different lifelong methods have been developed to extract the learned knowledge and teach a student network new tasks to address the problem of forgetting in lifelong learning. (Gou et al., 2021, p. 16)

In neural architecture search, the idea is to identify deep neural models and adaptively learn appropriate deep neural structures. Here in distillation knowledge transfer depends not only on the teacher but also on the architecture of the student model. Issues related to student learning from the teacher might make it difficult due to the capacity gap between the teacher and a student model. (Gou et al., 2021, p. 16)

Models like BERT are very time and resource consuming with complex structures, so knowledge distillation in natural language processing is widely studied, and many different methods have been proposed to solve resource consumption in NLP tasks. Examples of these tasks include neural machine translation, text generation, question-answer systems, event detection, document retrieval, and text recognition. A so-called patient knowledge distillation was proposed for the compression of the BERT model, in which the feature representation of the token [CLS] of the track layers is transferred from the teacher to the student. To speed things up, a speech inference model called TinyBERT was introduced. It is a two-step transformative knowledge distillation that includes both cross-domain and task-specific knowledge distillation. (Gou et al., 2021, p. 20)

3.1.2 Quantization

Quantization involves improving the efficiency of deep learning computations through smaller representations of model weights. By applying different quantization methods, the aim is to reduce the number of bits to represent a single scalar parameter. The definition of quantization is the division of a quantity into a discrete number of small parts. Often the parts are assumed to be integral multiples of a common quantity. The oldest example of quantization is rounding off. Generally, a can be defined a quantizer as consisting of a set of intervals or cells, where the index set is ordinarily a collection of consecutive integers beginning with 0 or 1, together with a set of reproduction values or points or levels. (Gray and Neuhoff, 1998, p.1)

In a uniform quantizer, the levels are evenly distributed, and the thresholds are midway between adjacent levels. In an infinite number of levels, all cells have a width equal to the delta, the distance between levels. If only a finite number of levels are allowed, then all but 2 cells are delta-wide and the outermost cells are half-infinity. Given a cell width delta uniform quantizer, the region of the input space within delta/2 of a certain quantizer level is called the granular or support region. The outer range where the quantization error is unbounded is called the overload or saturation range. In general, the supporting or granular region of a non-uniform quantizer is the region of the input space within a relatively small distance of a given level. The overload region is the complement of the granular region. (Gray and Neuhoff, 1998, p. 1)

By measuring the rendering that results from a quantizer on the original, the quality of a quantizer can be measured. One way is to define a bias measure that quantifies the cost or bias that results from reproducing x and x and consider the average bias as a measure of a system's quality. Here, a smaller average distortion means higher quality. One of the most distorting errors is the squared error. In practice, the average is a sample average when the quantifier is applied to a sequence of real data. The theory assumes that the data share a probability density function that corresponds to a generic random variable and the mean bias becomes an expectation. (Gray and Neuhoff, 1998, p. 1)

It is desirable to have the average distortion as small as possible. Negligible average distortion is achievable by letting the cells become numerous and tiny. For describing the quantizer output to a decoder there is a cost in terms of the number of bits. The goal of quantization is to encode the data from a source, characterized by its probability density function, into as few bits as possible. This is done in a way that

reproduction may be recovered from the bits with as high quality as possible. Quantization has a trade-off between the two-performance measure: average distortion and rate. (Gray and Neuhoff, 1998, p. 2)

Most generally speaking, quantization complexity has two aspects: arithmetic and memory complexity. Arithmetic complexity represents the number of arithmetic operations per sample that must be performed when encoding or decoding. Memory complexity represents the complexity of the amount of auxiliary memory required for encoding or decoding. Keeping them on separate paths is being considered as the associated costs vary by location and in some locations, storage is so inexpensive that one is tempted to ignore it. Some techniques benefit from more memory, although the cost per unit is low. Memory usage should be increased until the ratio of the marginal gain to the cost of additional increases is small. At this point, the total cost of memory can be significant. These complexity measures require a number of qualifications. A decision must be made whether to count the encoding and decoding complexities separately, add them up, or just one of them is important. (Gray and Neuhoff, 1998, p. 2361)

When evaluating the complexity of a quantization technique, it is interesting to compare the complexity invested in the lossy encoder/decoder versus that of the lossless encoder/decoder. A quantizer is considered low complexity if both coders are low complexity. Some fixed-rate techniques, such as lattice quantization and scalar vector quantization, have so-called indexing problems. In an indexing problem, it's easy to find the cell that the source vector is in, but the cells map to a set of N indices, which are not simply the integers from 1 to N, where N is the number of cells. The non-trivial part is converting the cell identity to a sequence of $\log N$ bits. Some vector quantization techniques that are not prohibitively complex to implement but have many codevectors that are overly complex in design or require an excessive amount of training data. Another problem is that in some applications it is desirable that the encoder's output be progressively decodable in the sense that the first bits it receives can be roughly reproduced. Also, improved reproductions are made as more bits are received. These quantizers a are referred to as progressive or embedded. In some applications it is desirable that the encoding is also progressive. As it turns out, several vector quantization approaches address these last two problems with reduced complexity, meaning they're easy to design and advanced. (Gray and Neuhoff, 1998, p. 2362)

Lattice quantization can be viewed as a vector generalization of uniform scalar quantization that contains the playback codebook as a subset of a regular lattice. The network is the set of all vectors. The result is a Voronoi partition where all cells have the same shape, size, and orientation. The quantization technique was proposed by Gersho because of its near optimum for high-resolution, variable-rate quantization of evenly distributed sources. Although

low complexity algorithms were found for the lossy encoder, issues impact the performance and complexity of the lattice quantizers. For variable rate coding, the network must be scaled to get the desired rate and distortion. Also, an algorithm must be implemented to map integers to variable-length binary keywords. For rate *R* fixed rate coding, the network must be scaled and a subset of *2kR* network points must be identified as the code vectors. This includes a support region. The grid quantizer is usually chosen to have the same support when the source has finite support. If not, then the scaling factor and grid subset are usually chosen such that the resulting quantizer support range has a high probability. In both cases, for the assignment of binary sequences to the chosen code vectors i.e., for indexing, a low-complexity technique is required. (Gray and Neuhoff, 1998, p. 2363)

A product quantizer uses a reproduction codebook that is the Cartesian product of lower dimensional reproduction codebooks. For example, the application of a scalar quantizer to k successive samples can be viewed as a product quantizer operating on the k-dimensional vector. It makes searching easier and, unlike the special case of a sequence of scalar quantizers, the search needs to be comprised-independent searches. Products of vector quantizers are also possible. Typically, the product quantizer is applied to some functions or features extracted from the vector. (Gray and Neuhoff, 1998, p. 2364)

An example of a product quantizer is a shape-gain vector quantizer that uses a product reproduction codebook consisting of a positive scalar gain codebook and a positive scalar codebook in the form of k-dimensional vectors of a uniform norm. One of the advantages of these systems is that by separating these two features, scalar quantization can be used for the gain feature and a lower rate codebook for the shape feature. This special feature can have a larger dimension with the same search complexity. Problems occur in an example of the rate allocation problem; What is the best way to split the bits between two codebooks in an overall rate limit? In a shape-gain vector quantizer, the optimal lossy encoder will generally not see only one coordinate at a time. Low complexity can be achieved by separate and independent quantization of the components, but usually a suboptimal encoder. With this particular quantizer, the optimal lossy coder is a simple sequential operation. The gain quantizer is scalar, but the choice of one of its quantization levels depends on the result of another quantizer, the shape quantizer. The Fischer pyramid vector quantizer is also a type of shape gain quantizer. Here the codevectors of the shape codebook are constrained to lie on the surface of a pyramid of dimension *k*, that is, the set of all vectors whose components have magnitudes that sum to one. (Gray and Neuhoff, 1998, p. 2364)

Usually called polar quantizers are two-dimensional shape-gain product quantizers. In the basic scheme, the codebook consists of the Cartesian product of a nonuniform scalar codebook for the phase, and a two-dimensional source vector is represented in polar coordinates. The first versions used independent quantization of the magnitude and phase information. In later versions, better versions used a method described above, and some allowed the phase quantizers to have a resolution that depends on the outcome of the magnitude quantizer. With high-resolution analysis, the rate-distortion performance of these quantizers can be studied. These analyses allow for finding the optimal point density for the magnitude quantizer and the optimal bit allocation between magnitude and phase. (Gray and Neuhoff, 1998, p. 2365)

Scalar-Vector Quantization attempts to match the performance of an optimal entropy-constrained scalar quantizer with a low-complexity fixed-rate structure vector quantizer. A technique called block-constrained quantizer is easier to describe. In block-constrained quantizer, the reproduction codebook is a subset of the k-fold product of some scalar codebook. The scalar levels are associated with the variable-length binary codewords. Given some target rate R, the k-dimensional codebook contains only those sequences of k quantization levels for which the sum of the lengths of the binary codewords associated with the levels is at most kR. Using dynamic programming the minimum distortion codevector can be found. Using a knapsack packing or Lagrangian approach an essentially optimal search can be performed with low complexity. (Gray and Neuhoff, 1998, p. 2366)

In tree-structured quantization, a k-dimensional tree-structured vector quantizer is a fixed-rate quantizer with rate R. Its encoding is guided by the balanced binary tree of depth kR. With each of its $2kR$ terminal nodes, also called leaves, there is a codevector associated. K-dimensional test vector associated with each of its $2kR$ -1 internal nodes. Quantization of the source vector proceeds in a tree-structured search by finding which of the two nodes stemming from the root node has the closer test vector to the source vector. Then find which of the two nodes stemming from this node has the closer testvector until a terminal node and codevector are found. The binary encoding of this codevector consists of the sequence of kR binary decisions that leads to it. Using a table lookup decoding is done as in unstructured vector quantization. Using this method encoding requires storing the tree of testvectors and codevectors, demanding approximately twice the storage of an unstructured codebook. However, encoding requires only $2kR$ distortion calculations, which is a tremendous decrease over the $2kR$ required by a full search of an unstructured codebook. (Gray and Neuhoff, 1998, p. 2366)

Arithmetic complexity can be roughly halved to kR operations per sample and $2kR$ vectors in the case of quadratic error distortion. This can be achieved by storing the normal to the hyperplane at each interior

node, bisecting the test vectors at the two nodes derived from them, and determining on which side of the hyperplane x lies by taking an inner product of x with the normal to a threshold is compared, which is also stored. The greedy method of designing vector quantization of tree structures is to first design the test vectors derived from the root node. This can be done by applying Lloyd's algorithm to a training set. A tree-structured quantifier is analogous to a classification or regression tree. As such, unbalanced, tree-structured quantizers can be developed using the grow-and-prune gardening metaphor, and the best known is the CART algorithm. With CART, a balanced or imbalanced tree with more leaves than necessary is first cultivated and then pruned. By splitting all the nodes at each level of the tree or splitting one node at a time or greedily to maximize the reduction in distortion by increasing the rate, a balanced tree can be grown. The tree can then be pruned by removing all descendants of all internal nodes, making it a leaf. The method increases average distortion but decreases speed. In addition, it is likely that in cases of moderate to severe distortion, pruning will remove the leaves corresponding to the elongated cells, like cubes cut in half, leaving mainly cubic cells. (Gray and Neuhoff, 1998, p. 2367)

As a form of tree-structured quantization with reduced arithmetic complexity and storage, multistage vector quantization was introduced. A single codebook could be used for all branches of a common length instead of having a separate reproduction codebook for each branch in the tree. This is achieved by coding the residual error accumulated to that point instead of coding the input vector directly. In other words, by the following stage, the quantization error from the previous stage is quantized in the usual way. Reproduction is formed by summing the previous reproduction and the newly quantized residual. A multistage vector quantization contains all codevectors formed by summing codevectors from the reproduction codebooks used at each stage. In a sense, the multistage quantizer has a direct sum reproduction codebook and it can be viewed that the reproduction codebook is determined by the Cartesian product of the stage codebooks. Multistage structuring leads to a suboptimal vector quantizer for its given dimension. Usually, the direct sum for the codebook is not optimal. The greedy search algorithm, in which the residual from one stage is quantized by the next does not find the closest codevector in the direct sum codebook. In general, the usual greedy design method, which uses a Lloyd algorithm to design the first stage in the usual way and then to design the second stage to minimize distortion when operating on the errors of the first does not design an optimal multistage vector quantizer. (Gray and Neuhoff, 1998, 2367)

Implementing multistage-conditioned vector quantization calls for the storing of a scale issue and a rotation for every first degree. The multistage operates at the first-degree residual earlier than quantization via way of means of the second one degree. Since the first-degree cells are almost spherical, the rotation profits most effective a small quantity and can be omitted.

Complexity may be decreased via way of means of the usage of a lattice vector quantizer as the second one degree because the best-recognized lattice tessellations are so near the best-recognized tessellations. As a way of circumventing the reality that optimum vector quantizers cannot be applied with commanders, multistage-conditioned two-degree quantizers may be considered as having a piecewise-regular factor density. (Gray and Neuhoff, 1998, p. 2368)

Implementing cell-conditioned vector quantization requires the storing of a scale factor and a rotation for each first stage. The cell operates on the first-stage residual before quantization by the second stage. Since the first-stage cells are so nearly spherical, the rotation gains only a small amount and may be omitted. Complexity can be reduced by using a lattice vector quantizer as the second stage since the best-known lattice tessellations are so close to the best-known tessellations. As a means of circumventing the fact that optimal vector quantizers cannot be implemented with commanders, cell-conditioned two-stage quantizers can be viewed as having a piecewise-constant point density. (Gray and Neuhoff, 1998, p. 2368)

Codebook sharing is another scheme to adapt each phase to the previous one. Here each stage has a finite set of reproduction codebooks, one of which is used to quantize the rest based on the sequence of results from the previous stage, and each codebook is shared by a subset of the possible sequences of the previous stages. The method lies between the conventional multi-level vector quantizer and the tree-structured vector quantizer. First, each stage has a codebook shared by all result sequences from previous stages, and in the second, a different codebook is actually used for each result sequence from previous stages. (Gray and Neuhoff, 1998, p. 2368)

Feedback vector quantization allows the encoder and decoder to share a finite set of states and a custom quantizer for each state. The state must be determinable from knowledge of an initial state in combination with the binary code words transmitted to the decoder. Here both the encoder and the decoder must be able to track the state without channel errors. The result is a finite state version of a predictive quantifier. It is also known as an affine vector quantizer. In this particular quantization technique, given the binary keyword and state, the optimal playback decoder gives us a conditional expectation of the input vector. The optimal lossy encoder is not easy to describe. The next stage should be chosen in a way that ensures good future performance, rather than in a greedy way that minimizes the current squared error. Both final state and predictive vector quantizers generally use memory in the lossy encoder but use a memoryless lossless code that is applied independently to each

subsequent binary codeword. One can make dependent on previous binary code word or make lossless code state dependent. (Gray and Neuhoff, 1998, p. 2369)

One way to introduce memory into the lossy encoder of a vector quantizer to attain higher dimensional performance with low dimensional complexity is called address-vector quantization. In this method, in addition to the usual reproduction codebook, there is an address codebook containing permissible sequences of indices of code vectors in the codebook. In a contained channel code outer code has the same role as the address codebook, which limits the allowable sequences of codewords from the inner code. The method allows address-vector quantization to exploit the property that certain sequences of codevectors are much more probable than others. (Gray and Neuhoff, 1998, p. 2369)

In tree/trellis encoded quantization, unlike vector quantizers on tree structures, these systems enforced the tree structure on the sequence of symbols and not on a single vector of symbols. In the case of channel coding, where a good channel code can be converted into a good source code by reversing the order of encoder and decoder, the encoder is a convolutional code, the input symbols are shifted into a shift register as output symbols. formed by a linear combination of the shift register contents, shifted times. The sequence of output symbols could be represented by a tree structure. In the tree structure, each tree node corresponding to the state of the shift register and the branches connecting the nodes has been determined by the symbol last entered the shift register and identified by the corresponding output, the resulting output symbol if it is a branch taken out. Tree and trellis encoded quantizers can be viewed as vector quantizers with large block lengths and a reproduction codebook constrained to be small the possible outputs of a nonlinear filter or an affine state quantizer or a plus-dimensional vector quantizer. Both generate long codewords with a lattice structure. (Gray and Neuhoff, 1998, p. 2369)

Traditional trellis-encoded systems can be improved with trellis-coded quantization by labelling the trellis branches with entire sub codebooks rather than with individual reproduction levels. The gain is a reduction in encoder complexity for a given level of performance. Combinations of trellis-coding quantization have achieved excellent performance for example in image coding applications. (Gray and Neuhoff, 1998, p. 2370)

Gaussian quantization showed that a Gaussian source has the worst velocity warp function of any source with the same variance, indicating that the Gaussian source was extreme in terms of source encoding. This has provided a robust approach to quantization in the sense that there are vector quantizers designed for the

Gaussian source with a given process domain distortion that will not yield worse distortion when applied to any source with the same variance. The method has provided a approach to robust vector quantization. Robust vector quantization is code that may not be optimal for the font and would perform no worse than the Gaussian source it was designed for. The external properties of the Speedwarp function to a source with memory. In another study, it was shown that code developed for a Gaussian source would perform essentially the same if applied to another process with the same covariance structure. One approach uses the central limit theorem and then the known structure of an optimal scalar quantifier for a Gaussian random variable to encode a general process. The Gaussian variable is first filtered to produce an approximate Gaussian density, the scalar result is quantized, and then back filtered to retrieve the original. (Gray and Neuhoff, 1998, p. 2370)

In the min-max average sense, the Gaussian quantizers were described as being robust. A vector quantizer designed for a Gaussian source will yield no worse average distortion for any source in the class of all sources with the same second-order properties. An alternative formulation is to place a maximum distortion requirement the on quantizer design. If a quantizer bounds the maximum distortion for a class of sources a quantizer is considered robust. (Gray and Neuhoff, 1998, p. 2370)

A quantizer is no worse than a fixed distortion value for all fonts in some collections. An alternative approach is to be greedy and try to design code that delivers near-optimal performance. This occurs regardless of what source is encoded within any collection, which is the idea of the universal quantization approach. The idea is to have a lossless encoder that works well for different sources by running multiple lossless codes in parallel and choosing the that produces the fewest bits over a period of time. The lossless encoder would work for different sources. It does this by running multiple lossless codes in parallel and choosing the one that produces the fewest bits over a given period of time, sending a small overhead to let the decoder know what code the encoder used. The existence of a universal fixed-rate lossy code has been proved under certain assumptions about the source statistics and the source and codebook alphabet, and the idea has been used, for example, to extend a coding theorem to a non-stationary source. The idea was used using the ergodic decomposition to interpret a non-ergodic source as a universal coding problem for a family of ergodic codes. A universal code is theoretically more complicated than an ordinary code. Instead, in practice it may mean that codes with smaller dimensions can be more efficient, as separate codebooks can be used for different behavior in the short term. Previously, universal quantization was considered more of a theory development method than a practical code design algorithm. Works assumed that the encoder and decoder had copies of the codebook used. A system has been considered where the codebooks are designed in the encoder. The codebooks were also encoded and

transmitted to the decoder, as is usually done with codebook supplementation. (Gray and Neuhoff, 1998, p. 2370–2371)

Performance tradeoffs can be improved by allowing both rate and distortion to vary. The universal coding problem was formulated as an entropy-constrained vector quantization problem for a family of sources and Lloyd-style design algorithms for the collection of codebooks subject to Lagrangian distortion measure. Measure yields a fixed rate-distortion slope optimization rather than fixed distortion or fixed rate. To study the rate of convergence with block length to the optimal performance high resolution quantization theory was used. It yielded results consistent with earlier convergence results developed by other means. The fixed-slope universal quantizer approach was further developed with other code structures and design algorithms. Another approach resembling traditional adaptive and codebook replenishment- did not involve training but created and removed code vectors according to the data received and an auxiliary random process in a way that could be tracked by a decoder without side information. (Gray and Neuhoff, 1998, p. 2371)

Interpolated quantization randomizes the effect of uniform quantization to minimize visual artifacts. The goal of dithering is to make the reconstruction error more like signal-independent additive white noise. Dithering, rather than directly quantizing an input signal, quantizes a signal that consists of a random process and a separate signal called the dithering process. Subtractive dithering achieves well behaved quantization noise as well as quantization error. However, is impractical for two main reasons. First, the receiver generally does not have a perfect analog connection to the transmitter, and therefore a pseudo-random deterministic sequence must be used in both the transmitter and the receiver. However, there is no guarantee that the quantization error and noise will have the properties that apply to true random dithering. Second, subtractive interpolation, which looks like a sampling function of a random process with no memory, is complicated to implement. Requires state RAM dither, high precision arithmetic, and perfect timing. (Gray and Neuhoff, 1998, p. 2371–2372)

Non-subtractive dither is not capable of making the reconstruction error independent of the input signal, but the proper choice of dithering function can make the conditional moments of the reproduction error independent of the input signal. As an example, the dithering function can make the perceived quantization noise energy constant as an input signal fades from high intensity to low intensity. Otherwise, it can exhibit strongly signal-independent behavior. In addition, in whitening quantization noise and making the noise or its moments independent of the input, dithering has a role in the proof of universal quantization. It has been shown that even without high-resolution theory, uniform scalar quantization combined with dithering and vector lossless coding could yield performance within 0.75 bit/symbol of the rate-distortion function. Also, extensions to lattice quantization and variations of this result have been developed. (Gray and Neuhoff, 1998, p. 2372)

Near optimal communication of an information source over a noisy channel can be achieved by quantization or coding of the source separately from source and channel coding or error control coding of the resulting scrambled source for reliable transmission of a noisy channel. This was stated in the separation theorem of information theory. Common source and channel codes, codes that together account for quantization and reliable communication, must be considered if is to be implemented near the Shannon limit for moderate delays or block lengths. There are a variety of code structures and design methods that have been considered for designing source and channel code together. (Gray and Neuhoff, 1998, p. 2372)

When designing a quantizer to use noisy channels, one approach is to replace the distortion measure that a quantizer is optimized for with the expected distortion over the noisy channel, so that the channel's statistics are included an optimal quantizer design formulation. The method was called optimized channel quantization. The method was applied to Shannon's source coding theorem and the Lloyd-style layout algorithm was also provided. The method has also been applied to the quantization of tree-structured vectors. (Gray and Neuhoff, 1998, p. 2372)

To join source and channel coding based on a quantizer structure and not explicitly involving typical channel-coding techniques, another approach is to design a scalar or vector quantizer for the source without regard to the channel. Here the resulting indices are coded in a way that ensures that the small Hamming distance of the channel codewords corresponds to a small distortion between the resulting reproduction codewords. This is done to correspond to that of the resulting reproduction codewords. The odes usually doing this are called index assignments. For example, index assignment has been introduced in an iterative search algorithm for designing index assignments for scalar quantizers, which has been extended to vector quantization. (Gray and Neuhoff, 1998, p. 2372)

Determining the quantization rate to use when keeping the total number of channel symbols per source symbol fixed is a key issue when considering source and channel codes together. For example, as the quantization rate increases, the quantization noise decreases, but the channel induced noise increases because the channel code's ability to protect bits is reduced. Also, determining the rate at which overall distortion decreases in an optimal system as the total of channels used per source symbol increases. In addition, there are other approaches to joint source and channel coding, including using codes with a source-optimized channel coder structure or with a special source-adapted decoder. Unequal error protection is used to better protect the most important reproduction indices, common combinations of channel-optimized quantizers with source-optimized channel codes. (Gray and Neuhoff, 1998, p. 2372-2373)

Quantization for a noisy channel a parallel problem is quantization for a noisy source. Attempting to compress a dirty source into a clean rendition or estimating the source based on a quantized version of a noise-impaired version can be seen as a problem. This can be treated as a quantization problem with a modified distortion measure if the problem can be viewed as trying to compress a dirty source into a clean rendition or making a source estimate based on a quantized version of a noisy version. The modified distortion measure for a quadratic error distortion was used to prove that the optimal quantizer for the modified distortion could be decomposed into the cascade of a least mean squared error estimator followed by an optimal quantizer for the estimated source. The result has been extended to a more general class of distortion measures, including input-weighted quadratic distortion. A generalized Lloyd algorithm for layout is presented here. (Gray and Neuhoff, 1998, p. 2373)

In several descriptions, the quantization of noisy channels, where the problem is usually formulated as a problem of source coding or quantization over a network, is a closely related topic to the quantization of noisy channels. Multi-description quantization is most easily described in terms of packet communications. For example, suppose that two packets of information, each at rate R, are transmitted to describe a reproduction of a single random vector. The encoder may receive one or all of the other packets, or both together, and wants to provide the best possible reconstruction for the received bit rate. This can be viewed as a network problem where one receiver sees only one channel, another receiver sees the second channel, and a third receiver sees both channels. The goal here is that everyone has an optimal reconstruction for the total received bitrate. It can be made better that each packet alone results in a distorted display close to the Shannon distortion rate function, while at the same time the two packets produce a distorted display. Unfortunately, this positive overall performance is not possible (Gray and Neuhoff, 1998, p. 2373) In Table 1 all discussed distillation and quantization methods are listed.

Table 1: Table of above discussed distillation and quantization methods

Distillation methods	Quantization methods
• Response-based knowledge distillation • Feature-based knowledge distillation • Relation-based knowledge distillation • Online-based knowledge distillation • Co-distillation • Self-distillation • Adversarial distillation • Cross-modal distillation • Graph-distillation • Attention-based distillation • Data-free distillation • Quantized distillation • NAS-based distillation	• Uniform quantization • Lattice quantization • Scalar vector quantization • Progressive/embedded quantization • Product quantization • Polar quantization • Scalar-vector quantization • Block-constrained quantization • Tree-structured quantization • Multistage quantization • Cell-conditioned vector quantization • Feedback vector quantization • Tree/trellis-encoded quantization • Gaussian quantization • Dithered quantization • Noisy channel quantization • Multiple description quantization

4 Literature Review

Next, a literature review was conducted in order to find different quantization and distillation methods applied to NKLP and BERT models. Also, different modeling cases were also considered. Different academical articles were gone through to find possible methods for decreasing model size and improving their inference speed. First, different studies related to quantization methods are introduced after which distillation and other findings are presented. Information regarding different studies was searched using LUT Primo using keywords like *BERT, named entity recognition, distillation, and quantization.*

4.1 Quantization methods

Piao et. al. (2022) proposed a quantization-based BERT compression model improving the model's efficiency by three aspects: model size, accuracy, and inference speed. The size is decreased by sensitivity-aware mixed precision quantization. This improves the quantization approach by choosing target compression ratios based on the sensitivity of modules in BERT, and it is demonstrated that the encoders close to the input layer are more sensitive than those near the output layer in BERT, and the Self-Attention layer is more sensitive than the feed-forward network in an encoder. These sensitive parts are quantized to 8-bit and the remaining parts to 1-bit. 8-bit quantization 8-bit index quantization is introduced to reduce the model size while retraining the accuracy by using 8-bit indices, minimum weight, and maximum weight to efficiently represent all weights of each layer. To achieve fast inference speed proposed model applies FP16 general matrix multiplication (GEMM) to the 8-bit parts of the model and XNOR-Count GEMM to the 1-bit parts. Experiments were conducted on four GLUE downstream tasks. Experiments showed that the model compresses BERT 8 times in terms of models' size and gives 5 times faster inference speed. The study also showed that three conducted 1-bit training methods improved the average accuracy by 1.1-1.4 %. (Piao et al., 2022, p 2-3)

The accuracy loss of the model is minimized with three training methods: Absolute Binary weight regularization (ABWR), Prioritized Training (PT), and Inverse Layer-wise Fine Tuning (ILF). ABWR is a regularization method to reduce the precision loss by learning new weight distribution that fits the 1-bit value quantization. The intuition of ABWR is to train the absolute value of the weights to become close to 1 in the first place, thereby minimizing the drop in accuracy when 1-bit quantization is applied. PT is proposed to overcome the difficulty to train the model with binary weights. In conventional 1-bit quantization methods binarize both input and weights from the beginning to the training. PT keeps the input

binarized as in the conventional methods, but trains the weights with FP32 precision first, and then applies 1-bit quantization. ILF is introduced to overcome the difficulty in training a model that applies 1-bit value quantization to a large portion of the model at once. (Piao et al., 2022, p 12-13)

Experiments conducted by Cho et. al. achieved similar accuracy (90.4) to the BERT-base model, but up to 6 times smaller model size. Also proposed model showed better accuracy compared to the pruning method with a similar model size. Compared to Q8BERT with a model size of 25 % from the BERT-base and average accuracy of 90.2 %, the proposed model reduces the model size to 7.7 %, while achieving higher accuracy. While BERT-base inference time takes 480 seconds, for the proposed model takes only 95 seconds to make inferences on all the tasks' training sets on average. (Piao et al., 2022, p. 1, 17)

When converting data representation in higher precision to low-precision ones, which can only represent a smaller number of different values, how to handle outliers is believed to be a key issue. Outliers occupying a limited percentage of the values are decided to be kept and mapped into the range of low precision, a drawback is that a large-scale factor must be selected. As a result, the other element, which is in the majority and has contributed most to the accuracy, would be concentrated on a small part of the targeted range and become numerically close to each other leading to a degradation in accuracy. To avoid this issue clipping is expected to offer a solution. To implement clipping, a threshold must be decided in advance. Outliers that are beyond the set threshold, would be saturated to the threshold. In research conducted by Zhang et. al. (2022) clipping positions for weights belonging to each layer of BERT were integrated. The effectiveness of clipping was illustrated by extracting weights belonging to the same layer in BERT, the distribution of those weights was analyzed, and MSE was calculated, which showed that clipping effectively shortens the intervals between sample points, indicating an improved resolution and reducing MSE. (Zhang et al., 2022, p. 3-4)

In two-piece-wise quantization, data points are divided into multiple classes, while for different classes, linear quantization with a unique parameter of the scale factor is adopted. In this quantization method splitting all the data points into two segments with data belonging to each segment quantized to a 7-bit integer. Data is split by selecting a split point, which in this case is called *thresh*. The precision of data is changing oppositely through selecting a different split point. (Zhang et al., 2022, p. 4)

In experiments conducted by Zhang et. al. (2022) code developed on the hugging face library the quantization part of Q8BERT was overridden by clipping and two-piece wise quantization. With weight data in 8-bit integer, a model performance over 98 % of the full precision, 32-bit floating points, the baseline

is maintained for different tasks. With a 4-bit weight, 96.8 % to 98.8 % of the baseline for GLUE can be achieved. With 7-bit activation, there was no performance loss in some tasks, and with 4-bit activation and full quantization, the proposed method still has over 90 % performance in most benchmarks but can gain over 70 % or 50 % improvement in hardware implementation. (Zhang et al., 2022, p. 6)

A study conducted by Qiu et. al. (2022) proposed an end-to-end contrastive product quantization model to jointly refine the original BERT embeddings and quantize the refined embeddings into codewords. The study was motivated by two main problems. Firstly, methods for Approximate Nearest Neighbor, used in document search, are mostly on top of outdated TFID features, which do not contain various kinds of important information about documents, like word order and contextual information. Instead, in recent years pre-trained BERT has achieved success in various downstream tasks. However, it has been reported that BERT embeddings are not suitable for semantic similarity-related tasks. Secondly, to guarantee the efficiency of retrieval, most existing methods quantize every document to a binary code via semantic hashing. (Qiu et al., 2022, p. 1-2)

First, the original BERT embeddings are transformed via a learnable mapping and feed the transformed embedding into a probabilistic product quantization module to output a quantized representation. A probabilistic contrastive loss is designed and trained in an end-to-end manner, simultaneously achieving the optimization of refining and quantizing modules. To improve the retrieval performance a mutual information maximization base method is developed to increase the representativeness of learned codewords. This enables the cluster structure hidden in a dataset of documents to be kept soundly, making the documents quantized more accurately. The proposed method outperformed benchmarks by more than 4 %. It was also observed that the retrieval performance of the proposed method consistently improves as the code length increases. (Qiu et al., 2022, p. 2)

Qiu et. al. further evaluated the retrieval performance of two variants of their method. At first, the model removes the mutual-information term in each codebook and only optimizes the quantized contrastive loss to learn semantics preserving quantized representation, and the second method does not inject Gumbel noise but utilizes the sole SoftMax operation to produce the deterministic codeword index. When compared to the originally proposed method, the model improves the retrieval performance averaged over all code lengths by 1.51 % and 0.94 % respectively, demonstrating the effectiveness of mutual-information terms inside each codebook. (Qiu et al., 2022, p. 7)

ZeroQuant is an end-to-end post-training quantization and inference pipeline proposed by Yao et. al. (2022) to address challenges targeting both INT8 and INT4/INT8 mixed-precision quantization. In their study fine-grained hardware-friendly quantization schemes for both weight and activations, group-wise quantization for weight, and token-wise quantization for activations. Both quantization schemes can significantly reduce the quantization error and retain hardware acceleration properties. A novel layer-by-layer knowledge distillation method for INT/INT8 mixed-precision quantization, where the neural network is quantized layer-by-layer through distillation with minimal iterations and without access to original training data. At any given moment, the device memory is primarily populated only with a single extra layer's footprint, making billion-scale model distillation feasible with a limited training budget and GPU devices. Also, a highly optimized inference backend was developed eliminating the expensive computation cost of quantization /dequantization operators, which enables latency speedups on INT8 Tensor cores on modern GPU hardware. (Yao et al., 2022, p. 1)

Empirical studies showed that ZeroQuant enables BERT into INT8 weight and activations to retain accuracy without incurring any retraining cost. INT8 model achieves up to 519 times speedup on BERT-base on A100 GPUs. ZeroQuant with layer-by-layer knowledge distillation can do INT4/INT8 mixed-precision quantization for the BERT style model, which results in 3x memory footprint reduction with marginal accuracy loss as compared to the FP16 model. Laos it was demonstrated that the scalability of ZeroQuant on two of the largest open-source language models with INT8 quantization, where ZeroQuant can achieve 3.67X speedup over the FP116 model and reduce the GPU requirement for inference from 2 to 1. (Yao et al., 2022, p. 1)

In a study conducted by Qin et. al. (2022) was found that the performance drop in BERT with binarized 1-bit weight, activation, and embedding comes from the information degradation of the attention mechanism in the forward propagation and the optimization direction mismatch of the distillation in the backward propagation. The analysis also showed that direct binarization leads to the almost complete degradation of the information of attention weight, which results in the invalidation of the selection ability for the attention mechanism. It was also shown that severe optimization direction mismatch is caused by utilizing the attention score, the direct binding product of two binarized activations, since the non-neglectable error between the de facto and expected optimization direction. (Qin et al., 2022, p. 2)

The BiBERT model is proposed to turn the full-precision BERT into a strong fully binarized model. Also, the Bi-Attention mechanism is introduced to tackle information degradation of the attention mechanism. Bi-Attention applies binarized representations with maximized information entropy, allowing the binarized

model to restore the perception of input contents. With the Direction-Matching Distillation scheme, the direction mismatch is eliminated. The scheme takes appropriate activation and utilizes knowledge from constructed similarity matrices in distillation to optimize accurately. (Qin et al., 2022, p. 2)

Experiments were done on the GLUE benchmark show that BiBERT outperforms existing quantized BERT models with low-bit activation. For example, the average accuracy of BiBERT exceeds 1-1-1 bit-width BinaryBERT (1-bit weight, 1-bit embedding, and 1-bit quantization) by 20.4 % accuracy on average, and better than 2-8-8 bit-width Q2BERT by 13.3 %. In model size BiBERT is compressed by 31.2 times, which shows the advantage and potential of fully binarized BERT in terms of fast inference and flexible deployment in real-world resource-constrained scenarios. 2 With data augmentation BiBERT achieves comparable performance with full-precision BERT on several tasks with 90.9 % accuracy, which also indicates that BiBERT makes full use of the limited representation capabilities by the well-designed structure and training scheme. Also, it was shown that BiBERT on TinyBERT compact architectures still outperforms existing quantization methods on BERT-base. (Qin et al., 2022, p. 8)

For reducing memory footprint by storing parameters and/or activations with low bit precision quantization method is used by Kim et. al. (2021, p. 1). The proposed model I-BERT is an integer-only quantization scheme for transformers. The entire inference is performed with integer arithmetic and key elements include approximation methods for nonlinear operations such as GELU, SoftMax, and LayerNorm. The model has been evaluated with RoBERTa-Base/Large, where the quantization method improves the average GLUE score by 0.3/0.5 points compared to the baseline. The model has been also on end-to-end inference latency, showing that the quantization scheme can achieve 4 times faster speedup compared to the floating-point baseline. (Kim et al., 2021, p. 9)

A study conducted by Fan et. al. (2020) introduces a model by quantizing only a subset of weights instead of the entire network during training, which is more stable for high-compression schemes. By quantizing only, a random fraction of the network at each forward, most of the weights is updated with unbiased gradients. Employing a simpler quantization scheme during the training makes it useful for quantizers with trainable parameters, such as Product Quantizers (PQ) for which quantization proxy is not parametrized. The approach applies a quantization noise, called Quant-Noise, to a random subset of weights, which makes a network more resilient to various types of discretization of weights. The approach reached 82.5% accuracy on MNLI by compressing RoBERTa to 14 MB. (Fan et al., 2020, p. 2)

The study proposed by Zafrir et. al. (2019, p. 1) authors point out that real-time NLP applications that integrate BERT must meet low latency requirements to achieve a high-quality customer experience, which poses a challenge to the deployment of these models to production. Models a have height impact on how the way business organizations consume computing resources since computing resources will have to the handle loading of large models and heave feed-forward calculations, shifting workload focus from lower-level training to more application-specific fine-tuning and inference.

Zafrir et. al. (2019) in their study implemented the quantization method for the BERT model by quantizing fully connected layers and embedding layers using linear quantization. The higher requirement for operations was kept in the original Int32 values. These included operations like SoftMax, Layer Normalization, and GELU. In total 99 % of the model's weight was compromised to 8bit. This achieved reducing the memory by 4 times compared to the original model. Model maintained 99% accuracy in comparison to FP32 which refers to BERT-Base, which has 110M parameters in 32bit floating point representation. (Zafrir et al., 2019)

Jacob et al. (2017) in their study provide a quantization scheme that quantizes weights activations as 8-bit integers and a few parameters as 32-bit integers. Also, they introduce a quantized training framework that is implementable on integer/arithmetic/only hardware., and a quantized training framework to minimize the loss of accuracy from quantization on models. The quantization scheme is implemented using integer-only arithmetic during inference and floating-point arithmetic during training. (Jacob et al., 2017, p. 2)

4.2 Distillation and other methods

Avram et. al. (2022) introduces three compressed versions of compressed BERT models that were obtained through the distillation process. Distil-BERT-base-ro was obtained by distilling the knowledge of BERT-base-ro using its original training corpus and tokenizer. Distil-RoBERT-base was created from RoBERT-base using both original training corpus and tokenizer, and DistillMulti-BERT-base-ro considered the distillation of the knowledge from an ensemble consisting of BERT-base-ro and RoBERT-base while relying on the combined corpus and coupled with the tokenizer of the former model. (Avram et al., 2021, p. 1-2)

Models were evaluated on five datasets and the results showed that they maintained most of the performance of the original models, while being approximately twice as fast when run on a GPU. Both DistilBERT-base-ro and DistilMulti-BERT-base-ro have a size of 312 MB and contain 81 million parameters, reducing

the size of BERT-base-ro by roughly 35 %. Distill-ROBERT-base is smaller, with 72 million parameters and a size of 232 MB, compressing ROBERT-base by the same amount. Named entity recognition evaluation showed that the Distil-BERT-base-ro obtained an F1-score of 79.42 %, which is almost identical to the F1-score obtained by the distilled ensemble of 79.43 %. Both models outperformed Distil-RoBERT-base by 0,3 % on the same metric, and the compressed model lagged the base models by more than 3 % on the F1 metric. (Avram et al., 2021, p. 2, 4)

In a study conducted by Zhou et. al. (2021) authors present an efficient knowledge distillation scheme that trains a light model called BiLSTM, which can retain the accuracy of its heavier counterparts, such as BERT, while significantly reducing costs. Solution exploits so-called soft surrogates, the most probable label sequences under the teacher model, to inform the student learner. Authors explore the Viterbi algorithm to expedite computation to efficiently identify the most likely label sequences and determine their relative likelihood. For sequence labeling, multi-grained knowledge distillation is used. (Zhou et al., 2021, p. 5705)

The idea in the Viterbi algorithm for sequence outputs is to extract information from the teacher model via drawing a set of most probable sequences. Then the sequences are presented to the student model during its training, to pass on the knowledge from, the teacher through various loss functions. (Zhou et al., 2021, p. 5706)

Experiments, to validate the proposed solution and elaborate its gains, were done with TensorFlow and executed on a single NVIDIA P100 GPU. The teacher model is constructed by a BERT model followed by the CRF layer. A dropout layer is concatenated to the BERT, followed by a fully connected layer. For the student model, BiLSTM+CRF architecture is used, which exploits Bidirectional LSTM to map input sequences into a sequence of feature vectors. The learned word embedding is reused from the teacher model and kept frozen during training. (Zhou et al., 2021, p. 5709)

In the results, the authors have found out that the teacher model outperforms baselines. In terms of the teacher is that directly copying the teacher embedding to the student model can be most helpful. Regarding distillation, it achieved cross-the-board performance gains relative to the no-distillation use of fixed pre-trained teacher embeddings baseline. Also, it was noticed that inducing data augmentation consistently improves student learning, and in all cases, sequence-level distillation outperforms token-level distillation, especially in the absence of data augmentation. (Zhou et al., 2021, p. 5710)

In many industry scenarios needed student models to require different widths and depths to meet various latency and memory requirements. Chen et. al. (2021, 571- 572) proposed an Extract Then Distill (ETD) method to reuse teacher's parameters for distillation purposes. ETD is a flexible and effective method to reuse. It firstly allows the student to have a narrower width than the teacher model. Models allow width-wise extract the teacher's parameters randomly and depending on the importance scores. This method applies to students with different models' architecture sizes. In their test, they manage to save 70 % of computation cost. Moreover, when using the same computing resources, ETD outperforms the baseline.

The model consists of three steps. First, parameters are extracted from a teacher model to a model called the thin teacher. The first step is called width-wide extraction. In the uniform layer selection step, thin teacher layers are selected with a so-called uniform strategy, and these parameters are used to initialize the student. The last step is called transformer distillation which performs last-layer distillation. (Chen et. al. 2021, p. 572-573)

To extract the teacher's parameters two different approaches were introduced. In the first approach, neurons are randomly extracted, and corresponding weight parameters are assigned to the student model. In the other approach, a score-based pruning method is used to extract the relatively important weights from the teacher model. (Chen et. al. 2021, p. 574) Compared with the baselines in the study ETD can achieve similar results with less than 28% computation cost. In their conclusions, it was pointed out that fine-grained extraction is needed to achieve better results and reusing the teacher's parameters is beneficial for most of the tasks. (Chen et. al. 2021, p. 578)

Bai et. al. (2021, p. 1, 8) in their study proposed *ternary weight splitting*, which takes the ternary model as a proxy to bridge the gap between the binary and full-precision models. The model converts both the quantized and latent full-precision weights in a well-trained ternary model to initialize BinaryBERT. BinaryBERT supports also *adaptive splitting*, which means that it can adaptively perform splitting on most important ternary modules while leaving the rest as binary, based on efficiency constraints such as model size or floating-point operations. On the GLUE and SQuAD benchmarks, BinaryBERT has less than a 0.5 % performance drop compared to the full-precision BERT-base model while being 24 times smaller. (Bai et al., 2020, p. 1, 8)

Lin et. al. (2015, p. 1-2, 8) et al. proposed an approach for quantizing neural networks, which consists of 2 components. In the forward pass, weights are stochastically binarized using so-called *binary connect* or *ternary connect*. 1 Binary connect allows eliminating multiplications in the feed-forward process by

stochastically sampling weights to be -1 or 1. Ternary connect allows weights to be also 0. 2 The idea is to eliminate most of the floating-point multiplications used during training feedforward neural networks. For backpropagation of errors method called quantized backpropagation is used, which converts multiplication into bit-shifts. For example, testing the model with the MNIST dataset by applying ternary connect and quantized backpropagation, an error of 1.15 % error rate was achieved while full precision training yielded a 1.33% error rate.

In a study conducted by Romero et. al. (2014, p. 1-3) team aimed to address the neural network compression problem by taking advantage of depth. The method originated from Knowledge distillation in which the authors propose to train thin and deep neural networks. In their study authors explore a proposed framework in which a student network is trained from the softened output of an ensemble of wider networks so-called teacher network. The idea is to allow the student network to capture information from true labels and finer structures learned by the teacher network. Knowledge distillation is designed such that student networks mimic teachers' architecture of similar depth. Also, so-called hints were introduced to student networks from teachers' hidden layers to guide the training process of the student.

In their implemented approach student network contained only 33.3% of the teacher's parameters achieving 91.61 % accuracy, which is higher than the teacher's network accuracy, of 85.8%. One of their implemented student networks with 36 times less capacity compared to the teacher network witnessed a minor performance decrease of 1.3%. Another student network outperforms the teacher by 0.9 % while being faster by 4.64 factor. In this study using information from teacher networks or student networks models with fewer parameters can run faster and/or generalize better than their teachers. Also, it was found with empirical evidence -that hinting at the inner layers of a thin and deep network with information from the teacher's hidden state generalizes better than hinting at classification targets. (Romero et al., 2014, p. 5, 9)

Luo et. al. (2020, p. 2) proposed a solution and optimization scheme from a light pre-trained model to downstream tasks. Authors point out that models often contain a large number of parameters which poses challenges for fine-tuning and online services for latency and capacity limitations in real-world applications even though large parameter models can achieve better performance. In their study authors propose a DistillBERT model in which the original 12 layers are reduced to 6 and soft label and hidden layer parameters are used from a teacher's model to train the student model. Compared to the benchmark BERT model size is reduced by 40% and the inference speed is increased by 60% decreasing the performance only by 3%. In their approach output distributions of the teacher, the model is transferred to the student model

to achieve the purpose of improving the effectiveness of the student model on the target tasks. In their experiments, it was found that fine-tuning is more effective for small models. (Luo et al., 2020, p. 4, 8)

In the literature review, it was found that different quantization methods like ZeroQuant and I-BERT have similar accuracies and much faster inference times compared to their base models. Some issues were found regarding the accuracy decrease in models like BiBERT. Regarding different distillation methods, it was found that the size of the models was decreased substantially like in Distill-BERT-base-ro model, and achieved better results compared to its base model. Also, the ETD model's computational savings were significant and ternary weight splitting gave good results regarding performance and model size. The summary of the works reviewed within the scope of this thesis is shown in Tables 2 and 3.

Table 2. Table of discussed quantization methods, use cases, and main results

Author	Year	Use case	Model	Key results
Piao et. al.	2022	Test with GLUE downstream tasks.	Quantization-based BERT compression	Compression by 8 times and 5 times faster inference speed. Improved average accuracy by 1.1-1.4 %.
Zhang. al.	2022	Evaluation of BERT with GLUE benchmark	Clipping and two-piece wise quantization	The 8-bit weight integer model has 98 % accuracy of the full precision 32-bit floating point. 4-bit weight 96.8-98.8 % accuracy of the baseline was achieved.
Qiu et. al.	2022	Document search	End-to-end contrastive product quantization	Method outperformed benchmarks by more than 4 % and retrieval performance averaged over all code lengths by 1.51 % and 0.94 %.
Yao et. al.	2022	Conversion of EBRT into INT8a and INT4/INT mixed-precision quantization	ZeroQuant	INT8 model achieves up to 519 times speedup on BERT-base on A100 GPUs.
Qin et. al.	2022	GLUE benchmark	BiBERT	BiBERT achieves performance with full-precision BERT on several tasks with 90.9 % accuracy. The average accuracy of BiBERT exceeds BinaryBERT by 20.4 % accuracy on average, and better than 2-8-8 bit-width Q2BERT by 13.3 %.
Kim et. al.	2021	Evaluation with RoBERTa	I-BERT	Improved GLUE score by 0.3-0.5 %. Achieves 4 times faster speedup compared to baseline.
Fan et. al.	2020	MNLI dataset compressing RoBERTa	Quantization of subset of weights	82.5% accuracy on MNLI by compressing the model to 14 MB.
Zafir et. al.	2019	Real-time NLP applications.	Quantization of fully connected layers and embedding layers using linear quantization.	99 % of the model's weight was compromised to 8bit. The model maintained 99% accuracy.
Jacob et. al.	2017	Quantization training to ResNets and ImageNet dataset	Quantization of weights to 8-bit	Accuracy within 2 % of their floating-point counterparts.

Table 3. Table of discussed distillation methods, use cases, and main results

Author	Year	Use case	Model	Key results
Avram et. al.	2022	Named entity recognition	Distill-BERT-base-ro, DistillMulti-BERT-base-ro	Reduction in size of BERT-base model by roughly 35 %. Outperforming Distil-RoBERT-base by 0.3 %
Zhou et. al.	2021	Named entity recognition	Knowledge distillation scheme training light model called BiLSTM	Cross-the-board performance gains are achieved relative to the no-distillation use of fixed pre-trained teacher embeddings baseline.
Chen et. al.	2021	Evaluation of BERT with GLUE and SQuAD benchmarks	Extract Then Distill	Saving computation costs by 70 %.
Bai et. al.	2021	Evaluation of BERT with GLUE and SQuAD benchmarks	Ternary weight splitting	Less than a 0.5 % performance drop and model size decreased by 24 times.
Lin et. al.	2015	Test with MNIST dataset	Ternary connect	Elimination of most of the floating-point multiplication.
Romero et. al.	2014	Neural network compression	Knowledge distillation-based model	Containing only 33.3 % of the teacher's parameters achieving 91.61 % accuracy, which is higher than the teachers' network accuracy, of 85.8 %

5 Results

In the implementation part, different Hugging face model inferences were tested, and results were analyzed compared to the findings from the literature. Python programming language was used for model inference testing using the *task_evaluator* function provided by the *evaluation* library.

5.1 Data

Data used in the implementation part contains CoNLL-2003 which is a named entity recognition dataset released as a part of the CoNLL-2003 shared task for language-named entity recognition. The data consists of eight files covering two languages: English and German. For each of the languages, the dataset consists of a training file, a development file, a test file, and a large file with annotated data. The English data, which is used was taken from Reuters Corpus. The Corpus consists of stories between August 1996 and August 1997. (Tjong et al., 2003, p. 1-2)

Table 4 shows the dataset structure consisting of English dataset training, development, and test set. In the Table 4, LOC indicates the number of tokens related to location, MISC to miscellaneous names, ORG to organizations, and PER to persons. In the implementation, 1 000 sentences of the validation set are used to derive the inference values analyzed in this part. In the NER implementation, these four tokens are taken into account in the results analysis of the overall inference results.

Table 4. Structure of the ConLL-2003 dataset

English data	Articles	Sentences	Tokens	LOC	MISC	ORG	PER
Training set	946	12 987	203 621	7 140	3 438	6 321	6 600
Development set	216	3 466	51 362	1 837	922	1 341	1 842
Test set	231	3 684	46 435	1 668	702	1 661	1 617

5.2 Tested models

Five different HuggingFace models were tested in the implementation since they could already be used, and their inference could be tested with the evaluator. The first is called the bert-base-NER model, which is a fine-tuned BERT model that is ready to use for NER models, fine-tuned for the CoNLL-2003 dataset- The model is limited by its training dataset of entity-annotated news articles from a specific span of time, which may not generalize for all use cases in different domains. The second model is bert-base-multilingual-case-ner-hrl is a NER model for 10 high-resourced languages based on a fine-tuned mBERT base model. Which is also limited by its dataset of entity-annotated news articles from a specific span of time. (HUggingFace 2022c, HuggingFace, 2022e)

The third model is called the distilbert-base-uncased-finetuned-ner-model, which is based on the distillbert-base-uncased-model. DistillBERT base model is a distilled version of the BERT base model based on the paper of Sanh et al. (2019). The following model is a transformers model, smaller and faster than BERT. It was trained on the same corpus in a self-supervised fashion, using the BERT base model of a teacher. Particularly model was trained with three objectives. Using distillation loss, the model was trained to return the same probabilities as the BERT base model. MLM to randomly mask words in the input and then run the entire masked sentence through the model to predict the masked words and cosine embedding loss, where the model was trained to generate hidden states as close as possible to the BERT base model. (HuggingFace, 2022a, HuggingFace, 2022b)

The fourth model is an XML-RoBERTa model proposed by Conneau et al., which is based on Facebook's Roberta model released in 2019. XML-RoBERTa model is fine-tuned with the CoNLL2003 dataset in English. Potential downstream use cases include NER and Part-of-Speech (PoS) tagging. The fifth and last model is a BERT-large-cased-finetuned NER model fine-tuned with the CoNLL2003 dataset in English. (HuggingFace, 2022f)

5.3 Inference Results

All models were run using Google Collaboratory. Several inference evaluation results were gathered after each model's inference. Only overall evaluation metrics are analyzed, and token-specific evaluation metrics are not taken into the scope of this thesis because the aim is to compare overall results to their inference time and other evaluation results regarding the time used in inference. The gathered evaluation metrics are

overall accuracy, overall precision, overall recall, overall F1, total time in seconds, samples per second, and latency in seconds. The results of each model's metrics are described in the following Table 5.

Table 5. Inference results of five different BERT models. The best performance is bolded

Model	Overall accuracy	Overall precision	Overall recall	Overall F1-score	Total time in seconds	Samples per second	Latency in seconds
BERT base	99.22 %	96.60%	96.36%	95.98%	126.17	7.92	0.13
DistillBERT	98.37 %	90.41%	94.85%	92.1%	174.66	5.72	0.17
Distill RoBERTa	**99.33%**	**96.44%**	**97.04%**	**96.74%**	**67.66**	**14.77**	**0.07**
XML-RoBERTa	87.26%	96.96 %	98.06%	97.5%	451.42	2.21	0.45
BERT base multilingual	97.54%	97.08 %	86.97%	91.74%	132.29	7.55	0.13

As can be seen from the inference results from Table 5, best accuracies were achieved by BERT-base and Distill RoBERTa models with an accuracy of 99.33 %, while the best F1-score was achieved with XML RoBERTa-model with an F1-score of 98 %. Looking at the accuracies of all the models four of them achieved above 97 % except the XML RoBERTa -model with an accuracy of 87.26 %. When comparing the overall f1-score all models' results vary between 91.74 % and 97.74 %. The lowest score was achieved by BERT-base multilingual while the best score was achieved with the XML-RoBERTa model with an F1-score of 97.74 %.

When comparing the inference times between these models three evaluation metrics were gathered as described above. From the results can be seen that the lowest total time used for inference was achieved with Distill RoBERTa model with a total time of 67.66 seconds, which is almost two times less than the second fastest model. The slowest model of these five models in time was the XML-RoBERTa model with a total time of 451.42 seconds. Even though the fastest samples per second were achieved with the same model. When comparing the latency in seconds between different models best-achieving model was Distill RoBERTa model with a latency of 0.07 seconds and the worst-achieving model was the XML-RoBERTa model with 0.45 seconds in latency.

In Figure 3, the overall F1-score is compared to the samples per second evaluation metric from which can be seen that the XML-RoBERTa overall F1-score is among the highest and the samples per second are lowest comparing the other models. In Figure 5, due to the high latency of XML-RoBERTa model, Distill

RoBERTa, BERT-base are most prominent options of these models. From the Figure 5 it can be seen, the latency in seconds can be seen in a bar chart format from which can be seen the high latency time of the XML-RoBERTa model.

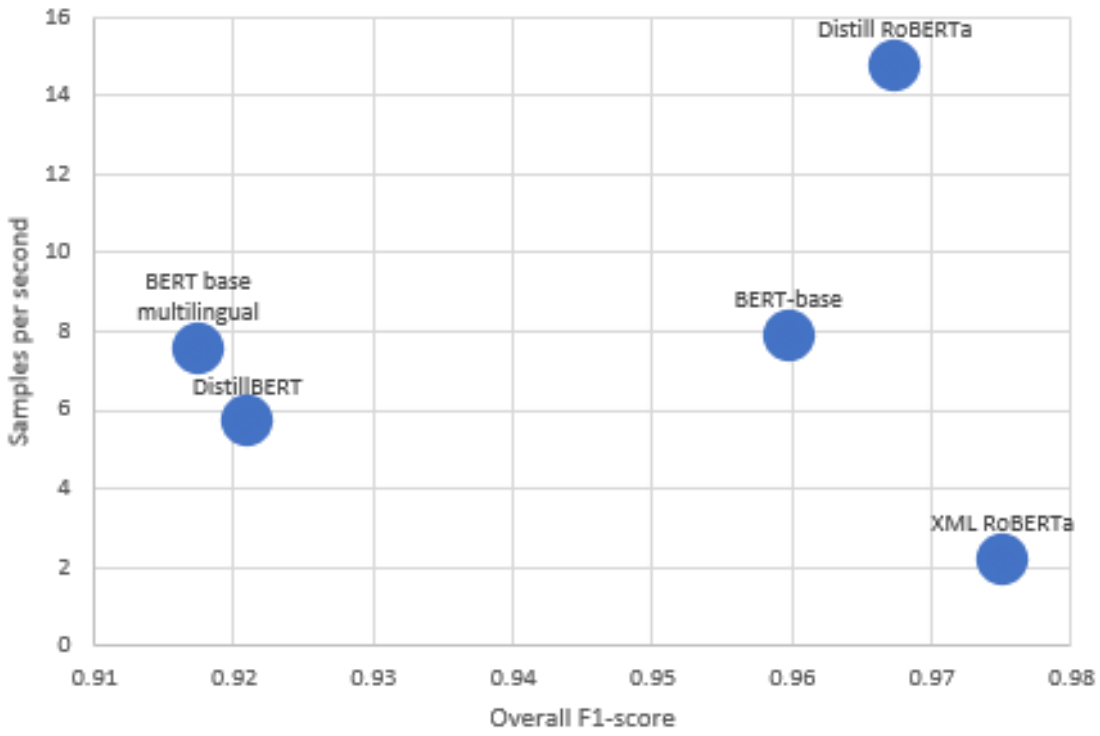

Figure 3. Overall F1-score compared to samples per second

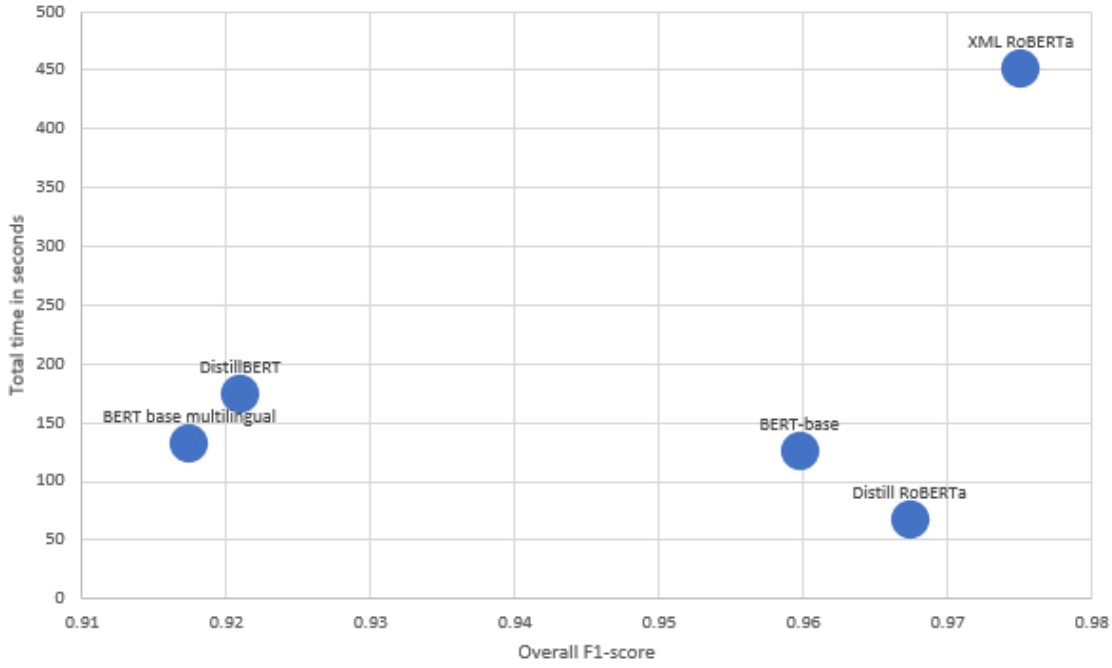

Figure 4. Overall F1-score compared to total time in seconds

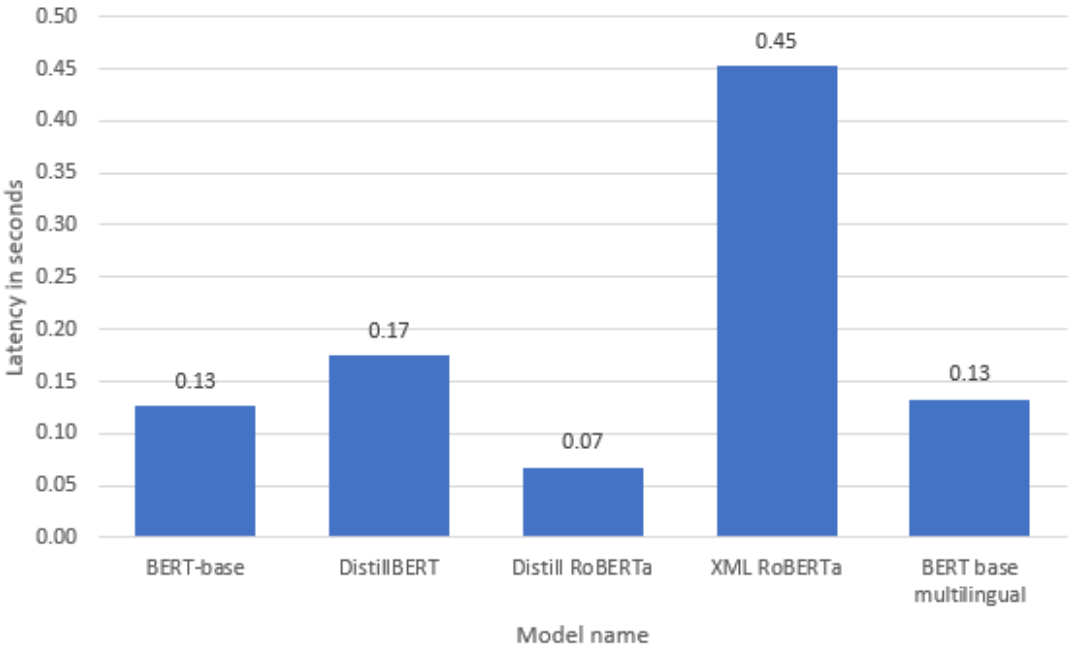

Figure 5. Models' latency in seconds

5.4 Result analysis

From Table 5 we can see accuracies are similar regarding models with accuracies around 97-98 % except the XML-RoBERTa model with accuracy of 87.26 % which is more than 10 % less compared to other models even though the F1-score of this model is 97.5 %, which is highest compared to the other models. The highest accuracy result was achieved with Distill RoBERTa model with the accuracy values of 99.33 %. When comparing the other models to best performing accuracy their accuracy is not less than 2 % except the accuracy of XML-RoBERT.

When comparing F1-scores it can be seen that high for BERT-base, Distill RoBERTa and XML-RoBERTa models and lower for DistillBERT and BERT base multilingual model. Highest F1-score is achieved with XML-RoBERTa model with F1-score of 97.5 % and regarding overall it can be seen that the F1-score values are distributed 91.74 % - 97.5 %.

When taking account time parameters, it can see that Distill RoBERTa has the lowest total time used in seconds with time of 67.66 seconds, which is around 67,66 seconds less compared to second less time used model and 58.51 seconds less compared to BERT base model. The worst performing model compared in total time in seconds was XML-RoBERTa, which used time 6.7 times more time compared to Distill RoBERTa model and 3.6 times more time than the BERT-base model. Even though XML-RoBERTa sample per second time is the lowest its latency in time is very high compared to other models with 0.45 seconds.

To take the performance values and time used in time from Figures 3 and 4 it can be seen that Distill RoBERTa and XML-RoBERTa performance is higher compared to BERT-base model, with F1-scores of 96.74 % and 97.5 %, but the time used for inference for both models are 67.66 and 451.42 seconds from which can be seen that the Distill RoBERTa performs better in both of the aspects.

Base on the result it can be stated the most feasible model is Distill RoBERTa due to the reason that its F1-score is among the highest and its total time used in inference the lowest and for those reasons it could be used among tested the models.

6 Conclusion and discussion

In this study, different methods for improving BERT model were reviewed and tested. Models' inference performance regarding time and cost and different purposes for it have been described, like efficient employment challenges and limited resources needed for these models to operate. Also, improvements can be seen cost-wise reducing the computational, but still achieving similar performance results. Despite the state-of-the-art results Transformer models like BERT are generally larger than previous models with a large number of parameters used.

From the literature review, it can be seen that different methods like Distill-BERT-base-ro by Avram et. al. (2022) and quantization-based BERT compression method by Piao et. al. (2022) was applied for reducing the size of these models and methods like knowledge distillation methods by Romero et. al. (2014) have decreased the number of parameters needed in order to use those models with constrained resources by reducing the computational cost of those models. As it was found out from the extant literature there are several different opportunities available for model inference optimization methods, which can improve throughput and with low latency without decreasing models' accuracy performance.

When reflecting on implementation results to theoretical background and literature review similarities regarding inference efficiency can be found even though several models were close to the base model like models based on RoBERTa where in study conducted by Kim et. al. (2021) has shown faster speedup compared to its base model and implementation of Distill RoBERTa has shown 0,06 second lower latency and using 58,51 seconds less time compared to BERT-base model, and DistillBERT model's performance was significantly lower compared to others with F1-score of 92.1 %. Next research questions are answered.

6.1 Answering research questions

The RQ1 considered production inference and it was formalized as "Which techniques can be used for decreasing BERT models' production costs inference recourses?"

From the theoretical background, different quantization and distillation methods are developed to reduce the model production cost of its inference resources by decreasing the size of the model, which can affect the throughput of the implemented model and reduce the total time and latency in the production without compromising its accuracy.

As seen from the literature review quantization models like clipping and two-piece wise quantization methods, ZeroQuant and I-BERT can achieve similar accuracy metrics as their baselines with much faster inference. For example, as discussed in the literature review section quantized models like ZeroQuant were able to achieve up to 519 times speedup compared to the BERT-base model. This is partly explained by the reason for reducing the number of parameters that the quantized model has, which implies the possibility of using these methods in order to decrease one model size for improving model inference performance without decreasing the accuracy of the model.

Regarding distillation methods like Distill-BERT-base-or ternary weight splitting can their base model size even outperforms their base model. Also, it was found that knowledge distillation for example in neural networks can significantly decrease the number of teacher's parameters and still be able the student to gain better accuracy compared to the teacher model, which implies that knowledge distillation might outperform the teacher model and reduce parameter amount decreasing the computation need and improving inference speed. Also, it was found out from the literature that models like ETD can save a significant amount of computation costs.

To answer RQ2 "Which methods can be used for increasing the inference of the BERT model's throughput by not severely decreasing the accuracy of the model?" literature review was conducted, and different BERT models were tested

First, different literature resources like studies conducted by Qiu et. al. (2022), Kim et. al. (2021) and Yao et. al. (2022) was used to find out possible methods to increase the throughput of a model and not decrease its accuracy significantly. From the literature's theoretical background different trends have been identified like parameter quantization and pruning and ways like model topology modifications and memory-based strategies. From the literature review, it was found that models like quantization-base BERT compression, ZeroQuant, and I-BERT were able to increase the BERT model's throughput. Regarding distillation models from the literature review reductions in size were achieved with for example Distill-BERT-base, Ternary weight splitting, and knowledge distillation. On the other hand, the BERT-base model in the implementation part gave better results compared to the DistillBERT model even though in the literature review it was found that distill model could improve the model's accuracy, while also its size was significantly less compared with the baseline model.

On the other hand, the implementation of Distill RoBERTa was able to increase the throughput and improve the F1-score compared to the BERT-base model. It was also found that this model was the only one whose

latency in seconds was lower compared to the BERT-base model, while the other models' latency time was equal or more, which can explain why other models did not perform better timewise compared to the BERT-base model. The performance of the RoBERTa model was also backed by the theoretical background where it was described that it is an optimized method of pretraining NLP systems. Also, Avram et. al (2022) in their study, applied distillation to the BERT-base model, as described in the literature review were able to decrease the model size and increase accuracy compared to the BERT-base model, which might indicate the benefit of using distilled models for gaining performance benefits in inference.

6.2 Further research and development

There are different research and development options to conduct regarding model inference optimization. For example, production-wise implementation can be researched and find out if a certain type or certain type of models could be implemented in production and how they will behave from point of view of accuracy efficiency.

Another development opportunity is to find out if is it possible to reduce production costs money-wise using this kind of model's inference optimization methods. For example, it can be researched whether is it possible to reduce maintenance costs in production with these above-discussed models.

7 References

Avram, A.-M., Catrina, D., Cercel, D.-C., Dascălu, M., Rebedea, T., Păiş, V., Tufiş, D., 2021. Distilling the Knowledge of Romanian BERTs Using Multiple Teachers.

Bai, H., Zhang, W., Hou, L., Shang, L., Jin, J., Jiang, X., Liu, Q., Lyu, M., King, I., 2020. BinaryBERT: Pushing the Limit of BERT Quantization.

Chen C., Chen X., Jiangm Z., Liu O., Shang L., Xin Z., Yin Y., 2021. Extract the distill: Efficient and effective task-agnostic BERT distillation. Artificial neural networks and machine learning – ICANN 2021. Vol.12893, 570-581.

Dai, Z., Lai, G., Yang, Y., Le, Q. v., 2020. Funnel-Transformer: Filtering out Sequential Redundancy for Efficient Language Processing.

Fan, A., Stock, P., Graham, B., Grave, E., Gribonval, R., Jegou, H., Joulin, A., 2020. Training with Quantization Noise for Extreme Model Compression.

Galindez Olascoaga, L.I., Meert, W., Verhelst, M., 2021. Hardware-Aware Probabilistic Machine Learning Models, Hardware-Aware Probabilistic Machine Learning Models. Springer International Publishing.

Gou, J., Yu, B., Maybank, S.J., Tao, D., 2021. Knowledge Distillation: A Survey. Int J Comput Vis 129, 1789–1819.

Graves, A., 2012. Sequence Transduction with Recurrent Neural Networks.

Gray, R.M., Neuhoff, D.L., 1998. Quantization. IEEE Trans Inf Theory 44, 2325–2384.

HuggingFace, 2022a, distilbert-base-uncased, Available at: https://huggingface.co/distilbert-base-uncased.

HuggingFace, 2022b, distilbert-base-uncased-finetuned-ner, Available at: https://huggingface.co/Rocketknight1/distilbert-base-uncased-finetuned-ner.

HuggingFace, 2022c, bert-base-multilingual-cased-ner-hrl, Available at: https://huggingface.co/Davlan/bert-base-multilingual-cased-ner-hrl.

HuggingFace, 2022d, bert-large-cased-finetuned-conll03-english, Available at: https://huggingface.co/dbmdz/bert-large-cased-finetuned-conll03-english.

HuggingFace, 2022e, bert-base-NER, Available at: https://huggingface.co/dslim/bert-base-NER.

HuggingFace, 2022f, xml-roberta-large-finetuned-conll03-english, Available at: https://huggingface.co/xlm-roberta-large-finetuned-conll03-english.

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., Kalenichenko, D., 2017. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference.

Jain, S.M., 2022. Introduction to Transformers for NLP, Introduction to Transformers for NLP. Apress.

Jiao, X., Yin, Y., Shang, L., Jiang, X., Chen, X., Li, L., Wang, F., Liu, Q., 2019. TinyBERT: Distilling BERT for Natural Language Understanding.

Kim, S., Gholami, A., Yao, Z., Mahoney, M.W., Keutzer, K., 2021. I-BERT: Integer-only BERT Quantization.

Li H., Luo H., Wang. X., and Zhang Y., 2020. Knowledge distillation and data augmentation for NLP light pre-trained models.

Lin, Z., Courbariaux, M., Memisevic, R., Bengio, Y., 2015. Neural Networks with Few Multiplications.

Liu, W., Zhou, P., Zhao, Z., Wang, Z., Deng, H., Ju, Q., 2020. FastBERT: a Self-distilling BERT with Adaptive Inference Time.

Luo, H., Li, Y., Wang, X., Zhang, Y., 2020. Knowledge distillation and data augmentation for NLP light pre-trained models, in: Journal of Physics: Conference Series. IOP Publishing Ltd.

Marrero, M., Urbano, J., Sánchez-Cuadrado, S., Morato, J., Gómez-Berbís, J.M., 2013. Named Entity Recognition: Fallacies, challenges and opportunities. Comput Stand Interfaces 35, 482–489.

Michelucci, U., 2022. Applied Deep Learning with TensorFlow 2, Applied Deep Learning with TensorFlow 2. Apress.

Piao, T., Cho, I., Kang, U., 2022. SensiMix: Sensitivity-Aware 8-bit index & 1-bit value mixed precision quantization for BERT compression. PLoS One 17.

Qin, H., Ding, Y., Zhang, M., Yan, Q., Liu, A., Dang, Q., Liu, Z., Liu, X., 2022. BiBERT: Accurate Fully Binarized BERT.

Qiu, Z., Su, Q., Yu, J., Si, S., 2022. Efficient Document Retrieval by End-to-End Refining and Quantizing BERT Embedding with Contrastive Product Quantization.

Romero, A., Ballas, N., Kahou, S.E., Chassang, A., Gatta, C., Bengio, Y., 2014. FitNets: Hints for Thin Deep Nets.

Sabharwal, N., Agrawal, A., 2021. Hands-on Question Answering Systems with BERT, Hands-on Question Answering Systems with BERT. Apress.

Sanh, V., Debut, L., Chaumond, J., Wolf, T., 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter.

Sekine S., Ranchhod E., 2009. Named entities. Recognition classification and use.

Shanmuganathan, S., 2016. Studies in Computational Intelligence 628 Artificial Neural Network Modelling.

Sun, S., Cheng, Y., Gan, Z., Liu, J., 2019. Patient Knowledge Distillation for BERT Model Compression.

Tambe, T., Hooper, C., Pentecost, L., Jia, T., Yang, E.Y., Donato, M., Sanh, V., Whatmough, P.N., Rush, A.M., Brooks, D., Wei, G.Y., 2021. EdgeBERT: Sentence-level energy optimizations for latency-aware multi-task NLP inference, in: Proceedings of the Annual International Symposium on Microarchitecture, MICRO. IEEE Computer Society, pp. 830–844.

Tjong, E.F., Sang, K., de Meulder, F., 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I., 2017. Attention Is All You Need.

Yao, Z., Aminabadi, R.Y., Zhang, M., Wu, X., Li, C., He, Y., 2022. ZeroQuant: Efficient and Affordable Post-Training Quantization for Large-Scale Transformers.

Zafrir, O., Boudoukh, G., Izsak, P., Wasserblat, M., 2019. Q8BERT: Quantized 8Bit BERT.

Zhang, X., Ding, Y., Yu, M., O'Uchi, S.I., Fujita, M., 2022. Low-Precision Quantization Techniques for Hardware-Implementation-Friendly BERT Models, in: Proceedings - International Symposium on Quality Electronic Design, ISQED. IEEE Computer Society.

Zhou, X., Zhang, X., Tao, C., Chen, J., Xu, B., Wang, W., Xiao, J., 2021. Multi-Grained Knowledge Distillation for Named Entity Recognition.

Zong, C., Xia, R., Zhang, J., 2021. Text Data Mining.