Optimizing the Performance of Text Classification Models by Improving the Isotropy of the Embeddings using a Joint Loss Function

Joseph Attieh

School of Science

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 28.7.2022

Supervisors

Prof. Tom Bäckström
Prof. Vladimir Vlassov

Advisors

Ph.D. Abraham Woubie
Ph.D. Adrian Flanagan

Aalto University
School of Science

Aalto University
School of Science

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the master's thesis

Author Joseph Attieh

Title Optimizing the Performance of Text Classification Models by Improving the Isotropy of the Embeddings using a Joint Loss Function

Degree programme Master's Programme in Security and Cloud Computing

Major Security and Cloud Computing **Code of major** SCI3113

Supervisors Prof. Tom Bäckström
Prof. Vladimir Vlassov

Advisors Ph.D. Abraham Woubie
Ph.D. Adrian Flanagan

Date 28.7.2022 **Number of pages** 53 **Language** English

Abstract

Recent studies show that the spatial distribution of the sentence representations generated from pre-trained language models is highly anisotropic, meaning that the representations are not uniformly distributed among the directions of the embedding space. Thus, the expressiveness of the embedding space is limited, as the embeddings are less distinguishable and less diverse. This results in a degradation in the performance of the models on the downstream task. Most methods that define the state-of-the-art in this area proceed by improving the isotropy of the sentence embeddings by refining the corresponding contextual word representations, then deriving the sentence embeddings from these refined representations. In this thesis, we propose to improve the quality and distribution of the sentence embeddings extracted from the [CLS] token of the pre-trained language models by improving the isotropy of the embeddings. We add one feed-forward layer, referred to as the Isotropy Layer, between the model and the downstream task layers. We train this layer using a novel joint loss function that optimizes an isotropy quality measure and the downstream task loss. This joint loss pushes the embeddings outputted by the Isotropy Layer to be more isotropic, and it also retains the semantics needed to perform the downstream task. The proposed approach results in transformed embeddings with better isotropy, that generalize better on the downstream task. Furthermore, the approach requires training one feed-forward layer, instead of retraining the whole network. We quantify and evaluate the isotropy through multiple metrics, mainly the Explained Variance and the IsoScore. Experimental results on 3 GLUE datasets with classification as the downstream task show that our proposed method is on par with the state-of-the-art, as it achieves performance gains of around 2-3% on the downstream tasks compared to the baseline. We also present a small case study on one language abuse detection dataset, then interpret some of the findings in light of the results.

Keywords Text Classification, Isotropy, Embeddings, BERT, IsoScore

Aalto University
School of Science

Aalto-universitetet, PB 11000, 00076 AALTO
www.aalto.fi
Sammandrag av diplomarbetet

Författare	Joseph Attieh
Titel	Improving the isotropy of the BERT embeddings using an Isotropy Layer and a Joint Loss Function
Utbildningsprogram	Masterprogram i Säkerhet och Molntjänster
Huvudämne	Säkerhet och Molntjänster — **Huvudämnets kod** SCI3113
Övervakare	Prof. Tom Bäckström Prof. Vladimir Vlassov
Handledare	PhD Abraham Woubie PhD Adrian Flanagan
Datum	28.7.2022 — **Sidantal** 53 — **Språk** Engelska

Sammandrag
Nya studier visar att den rumsliga fördelningen av de meningsrepresentationer som genereras från förtränade språkmodeller är mycket anisotropisk, vilket innebär att representationerna mellan riktningarna i inbäddningsutrymmet inte är jämnt fördelade. Inbäddningsutrymmets uttrycksförmåga är således begränsad, eftersom inbäddningarna är mindre särskiljbara och mindre varierande. Detta leder till att modellernas prestanda försämras i nedströmsuppgiften. De flesta metoder som definierar den senaste tekniken på detta område går ut på att förbättra isotropin hos inbäddningarna av meningar genom att förädla motsvarande kontextuella ordrepresentationer och sedan härleda inbäddningarna av meningar från dessa förädlade representationer. I den här avhandlingen föreslår vi att kvaliteten och fördelningen av de inbäddningar av meningar som utvinns från [CLS]-tokenet i de förtränade språkmodellerna förbättras genom inbäddningarnas isotropi. Vi lägger till ett feed-forward-skikt, kallat det isotropa skiktet, mellan modellen och de nedströms liggande uppgiftsskikten. Detta lager tränas med hjälp av en ny gemensam förlustfunktion som optimerar ett kvalitetsmått för isotropi och förlusten av nedströmsuppgiften. Den gemensamma förlusten resulterar i att de inbäddningar som produceras av det isotropa lagret blir mer isotropa, samtidigt som den semantik som behövs för att utföra den nedströms liggande uppgiften bibehålls. Det föreslagna tillvägagångssättet resulterar i transformerade inbäddningar med bättre isotropi, som generaliseras bättre för den efterföljande uppgiften. Dessutom kräver tillvägagångssättet träning av ett feed-forward-skikt, i stället för omskolning av hela nätverket. Vi kvantifierar och utvärderar isotropin med hjälp av flera mått, främst Förklarad Varians och IsoScore. Experimentella resultat på tre GLUE-dataset visar att vår föreslagna metod är likvärdig med den senaste tekniken, eftersom den uppnår prestandaökningar på cirka 2-3 % på nedströmsuppgifterna jämfört med baslinjen. Vi presenterar även en liten fallstudie på ett dataset för upptäckt av språkmissbruk och tolkar sedan några av resultaten mot bakgrund av dessa.

Nyckelord Klassificering av Text, Isotropi, Inbäddningar, BERT, IsoScore

Preface

This thesis work was conducted as part of the Erasmus Mundus Master's degree programme in Security and Cloud Computing (SECCLO). I am firstly grateful to the **SECCLO Consortium** for the scholarship that provided financial support during my studies, as well as the flexibility in my course choices. I also want to thank both **Eija Kujanpää** and **Anne Kiviharju** for the countless email exchanges and the administrative support throughout the degree.

I would like to express my deepest appreciation to my supervisors and advisors for their valuable guidance, timely suggestions, and continuous encouragement at all stages of my work. I want to specifically thank my advisor **Abraham Woubie** and my supervisor **Vladimir Vlassov** for the constant support and motivation, as well as for the help with proofreading and giving suggestions on how to improve the thesis. I also want to thank my supervisor **Tom Bäckström** for guiding me through the entire process of the thesis and encouraging me to question the findings on a more fundamental level.

I would also like to mention that this thesis would not have been achievable without the support that I received at Huawei. A special thanks go to my company supervisor **Adrian Flanagan**, for his advice and support throughout the experiments and for always guiding me in the right direction whenever I got lost.

I do not want to forget to thank the friends that I made in both Finland and Sweden. I am especially grateful for all the moments that we spent together studying, laughing, and having fun. These moments have taught me more than I had ever envisioned, and I will be forever grateful.

On a final note, I want to mention my deepest appreciation to my parents, **Marleine** and **Boutros** as well as my little sister (and best friend) **Mabelle** for their continuous emotional support and encouragement throughout my study.

Lastly, I want to give a special shout-out to my twin **Marinelle**; this has been one of the longest that we spent apart, but you never cease to be my rock.

Otaniemi, 28.07.2022

Joseph B. Attieh

Contents

List of Acronyms and Abbreviations

[CLS]	Classification Token.
[SEP]	Separator Token.
AI	Artificial Intelligence.
BERT	Bidirectional Encoder Representations from Transformers.
FFNN	Feed-Forward Neural Network.
GLUE	General Language Understanding Evaluation.
IsoBN	Isotropic Batch Normalization.
MLM	Masked Language Modeling.
NLP	Natural Language Processing.
NLU	Natural Language Understanding.
NSP	Next Sentence Prediction.
PC	Principal Component.
PCA	Principal Component Analysis.
RNN	Recurrent Neural Network.
SDG	Sustainable Development Goal.

1 Introduction

1.1 Background

Recent Transformer-based models have achieved significant success on various natural language processing tasks [1]. However, [2] observes that some language models including Bidirectional Encoder Representations from Transformers (BERT) produce contextualized word representations that are not isotropic. In other words, the information in the embeddings is not uniformly distributed amongst all directions in the space. This is not desirable as these representations vary the most in top directions, which limits the expressiveness of the space. [3] referred to this problem as the representation degeneration problem. Even though researchers did not agree on the source of anisotropy, having an isotropic space is very desirable as the more isotropic the space is, the more diverse the embeddings are. Furthermore, having an isotropic space affects the optimization of the model (i.e., convergence and accuracy), and leads to improvement in the performance of the model[4].

1.2 Problem

As mentioned previously, BERT-based models suffer from the problem of having an anisotropic space. This affects the representation capacity of the embedding space and affects the accuracy of the downstream task.

More specifically, [5] highlighted that the Classification Token ([CLS]) token representations are much more anisotropic than all representations in the fine-tuned space. The authors highlighted that this problem becomes even more dramatic after fine-tuning, as this process tends to concentrate information about the target task in the dominant directions (i.e., the top principal components). We also notice that improving the isotropy of the [CLS] embeddings is a relatively unexplored area; in fact, we find one study [6] that motivates the need to improve these embeddings and that proposes a method to improve the isotropy. Therefore, we propose to target these embeddings.

1.3 Objectives

The objective of the thesis is to learn an embedding transformation that improves the isotropy of the [CLS] embeddings; once improved, the embedding space should exhibit better statistical properties, which should result in a better performance on the downstream task (i.e., increase in the model performance).

1.4 Proposed System

The study conducted by [5] showed that improving the isotropy in general does not immediately result in a better performance for the model. Therefore, we propose to learn an embedding transformation that renders the [CLS] embeddings more isotropic without losing the information of the target task. Thus, we proceed by freezing the parameters of the fine-tuned model and the downstream task layer, and adding an Isotropy Layer between these two models. This Isotropy Layer is one feed-forward layer, which goal is to output embeddings of better isotropy. The parameters of this layer are learned using a joint loss function that combines an isotropic loss function and the downstream task loss function.

1.5 Research Methodology

We apply empirical methods to quantitatively measure the improvement of the models in terms of isotropy and performance on the downstream task. The improvement in isotropy is evaluated by computing two measures of isotropy, mainly the isoscore and the explained variance, while the improvement in the model performance is evaluated by computing a dataset-specific metric. Two main experiments are carried out. The first experiment compares the proposed method to the baseline (i.e., finetuned model with no Isotropy Layer), while the second experiment compares the method to Isotropic Batch Normalization (IsoBN) [6], the other only method in the literature that tackles the same goal as ours.

On a final note, my work at Huawei aimed to improve the performance of offensive language detection. Therefore, we present a case study, where we apply the proposed approach to improve the performance of a BERT-based abuse language detection model. We evaluate the approach on one public dataset in terms of F1 score.

1.6 Sustainability Goals

The primary goal of the project is to improve both the isotropy of the [CLS] embeddings generated by the model as well as the performance of this model on the downstream task. This is done by adding one layer between the pre-trained language model and the downstream task layers. Furthermore, applying the approach only requires training the parameters of this layer (instead of training the whole network). Therefore, the proposed solution results in a reduction in both power consumption and complexity (costs) of the system. Since this project contributes to a reduction in power, it promotes sustainability and helps combat CO_2 emissions in the long run.

Therefore, this project contributes to the United Nations Sustainable Development Goals (SDGs) [7] explained below:

- SGD 9: Build resilient infrastructure, promote inclusive and sustainable industrialization, and foster innovation,
- SGD 12: Ensure sustainable consumption and production patterns,
- SGD 13: Take urgent action to combat climate change and its impacts.

1.7 Contributions

To the best of our knowledge, our work is the second study besides IsoBN [6] that aims to improve the isotropy of the [CLS] token representations. The main contributions of this work are as follows.

1. We provide a method to improve the isotropy of the embeddings by adding an Isotropy Layer at the output of the finetuned language model, and only training this layer using a joint loss function.
2. As shown by [5], it is not sufficient to only improve the isotropy of the embeddings, as the embeddings need to maintain the semantics required for the downstream task. Therefore, we propose a novel joint loss function that optimizes both an isotropic loss measure as well as a downstream task loss, and results in embeddings that are more isotropic and that perform better on the downstream task. The isotropic loss is based on the IsoScore. This joint loss function should encourage the community of researchers to include unsupervised quality measures inside the loss function to enforce some statistical properties on the model.
3. We evaluate our method of improving the isotropy of embeddings on multiple datasets from the General Language Understanding Evaluation (GLUE) benchmark for several downstream tasks and compare its performance with the IsoBN method. Experimental results on 3 GLUE datasets demonstrate that our method can improve isotropy significantly, as well as improve the model performance.
4. We discuss abusive language detection and we provide one case study on a selected dataset. Furthermore, we present a discussion that reflects on the ethical considerations that apply to these systems.

1.8 Structure

This project is structured as follows: Chapter 2 discusses the general background needed to understand the content of this project. Chapter 3 introduces the concept of Isotropy as well as the related work that is relevant to the study. Chapter 4 presents the proposed method and describes the implementation of this method. Chapter 5 describes the experimental evaluation as well as the results obtained from these experiments. Chapter 6 provides a case study on a public abuse language detection dataset. Finally, Chapter 7 provides the summary of the findings and conclusions as well as the future scope of this research.

2 Background

This section provides the background needed to understand the content of the thesis. Firstly, we introduce the system used to model our text classification task. Afterward, we introduce both components of the proposed system, mainly the Data Representation module (i.e., the pre-trained language model) and the Downstream Task Layers respectively.

2.1 Text Classification

Text classification is the process of assigning predefined categories to textual documents. Some notable examples are news classification [8], spam detection[9], and abuse language detection [10].

The text classification system used in this study can be modeled as seen in Figure 1. We adopt this structure as it is the most common one in the literature. We present the following two components:

1. Data Representation module: It accepts the textual input and generates the embedding representation of this input. An embedding is a vector of double values that encode information that represents the text. The data representation module in this study is the pre-trained BERT-based model.

2. Downstream Task Classifier: It performs the classification task in question. It accepts the embedding representation of the text and performs the classification.

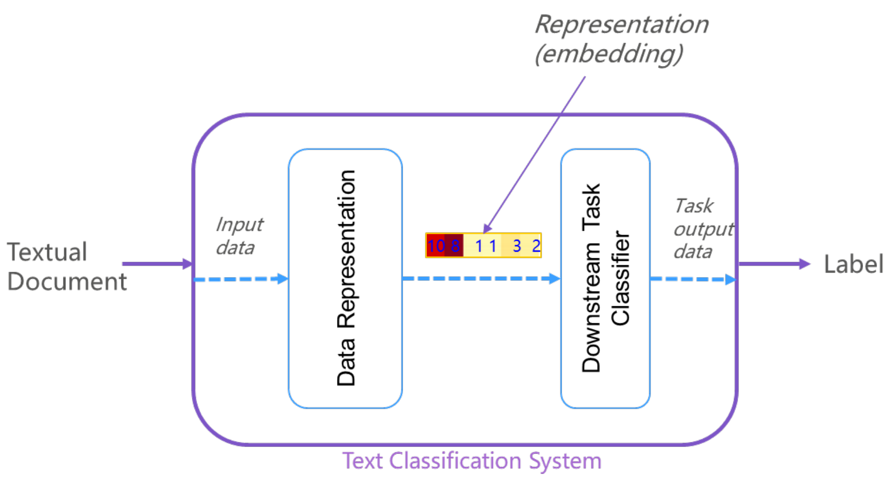


Figure 1: Diagram describing the system used in this thesis.

2.2 Data Representation Module

We choose BERT[11] as the data representation language model in this study. However, to understand BERT, we proceed by first presenting the mechanisms of the transformers.

2.2.1 Transformers

Transformers have been introduced in the famous paper "Attention is All you Need"[12]. They have been introduced as an alternative to the Recurrent Neural Networks (RNNs), as transformers allow to capture the long-term dependencies in the input regardless of the distance in the sequence. Furthermore, transformers allow parallelization of the computations, which is not possible with RNNs (due to their sequential nature). The mechanisms inside the transformer can be shown in Figure 2.

In this section, we summarize the findings from [12]. We first discuss the architecture of the transformer. Then, we explain the three mechanisms that are used by the transformer, mainly the attention mechanism, the self-attention mechanism, and the multi-headed self-attention mechanism.

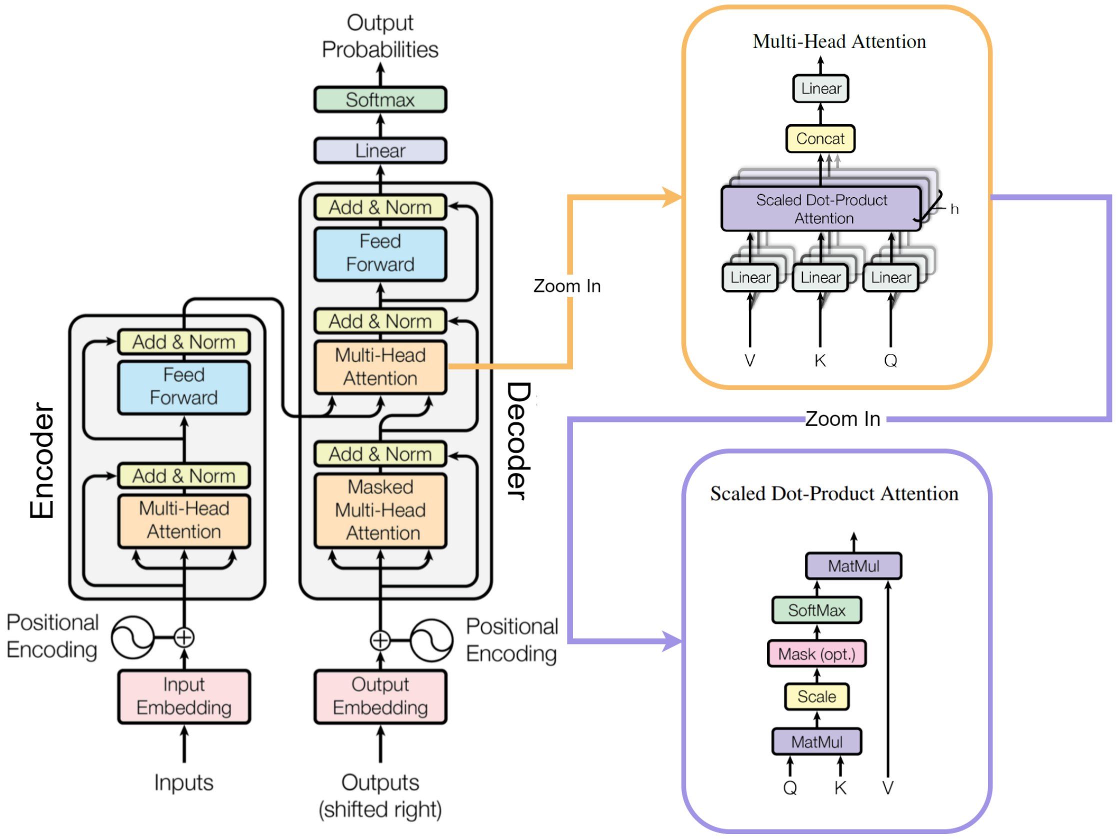


Figure 2: Diagram describing the transformer and some of its components (taken and adapted from [12]).

1. **Architecture:**
   The Transformer model uses an encoder-decoder architecture. The input is fed into the encoder; the encoder outputs the representation of the input and then sends it to the decoder. The decoder generates the output based on the representation received. We describe the encoder and decoder as shown in Figure 2:

   - **Encoder**: The transformer encoder structure consists of a stack of N encoder layers, where the output of each layer is fed to another layer. Each layer consists of two sub-layers, mainly the **Multi-Head Self Attention mechanism** (explained in the following subsections) and a simple fully connected feedforward network. Each of the two sub-layers employs a residual connection followed by a layer normalization (i.e., having a certain input *i* to the Sublayer, the output of each sub-layer is *LayerNorm(i + Sublayer(i))*).
   - **Decoder**: The transformer decoder follows the same architecture as the encoder (i.e., a stack of N decoder layers). The decoder receives two inputs, one from the encoder and one from the previous decoder (i.e., the previous output in the sequence). Furthermore, every decoder layer consists of an additional sub-layer referred to as **the Masked multi-head attention layer**, which performs multi-head attention over the output of the encoder stack. It is the same as the multi-head attention layer, except that the decoder can only see the words generated till the previous input step.

2. **Attention mechanism:**
   The attention mechanism mimics a database retrieval process; in fact, having a query q and a key-value database, the goal is to retrieve a value v_i based on how similar the query is to the keys of these corresponding values. Formally, the input consists of queries and keys of dimension d_k, and values of dimension d_v. The output of the attention mechanism is a weighted sum of the values, in which the weight corresponds to how similar the keys and queries are (similarity is computed in terms of a scaled dot-product). Mathematically, the whole mechanism can be computed by first getting the dot products (i.e., similarities) between the query with all keys, and applying a softmax function to obtain the weights on the values. We can write the formula as follows:

$$Attention(Q, K, V) = Softmax(\frac{QK^T}{\sqrt{d_k}}) \times V \tag{1}$$

The attention mechanism is used to capture long-range dependencies in the input sequence. In other terms, it gives more attention (higher weights) to the relevant parts of the input. At each decoding step, we compute a weighted average of all states of the encoder, changing the weights based on what deserves more attention based on the decoder state.

The presented attention mechanism differs from other mechanisms as the dot-product of the query and key matrices are divided by the dimension of the query and key matrices. This was introduced due to the limitations of the softmax function (it prevents the dot product from growing too large, affecting the output of the softmax function and interfering with gradient-descent algorithms).

3. **Self-attention mechanism:**
   Self-attention refers to the attention mechanism applied to linearly projected values of the same matrix. In other words, the mechanism uses a single vector X as the basis of the query, key, and value. This mechanism allows the model to learn within-sequence dependencies.

   Having three trainable weight vectors W_Q, W_K, and W_V, we compute the query, key, and value from X as follows: $Q = W^Q \times X$, $K = W^K \times X$, and $V = W^V \times X$.

4. **Multi-headed self-attention mechanism:**

   Transformers employs a multi-headed self-attention mechanism. This mechanism is an extension of the self-attention mechanism, where the input vector X is projected multiple times as the query, key, and value vectors (instead of projecting it only once). In other words, we employ different weight vectors for each head, and we compute the self-attention of each. Then, we compute the output by concatenating these values and performing a linear projection by passing them through a weight vector W_f. The process is explained in the equation below:

$$\begin{gathered} MultiHead(Q,K,V) = Concat(head_i)_{i=1,\dots,m} W_f \\ \text{where } head_i = Attention(Q \times W_i^Q, K \times W_i^K, V \times W_i^V) \end{gathered} \tag{2}$$

2.2.2 Bidirectional Encoder Representations from Transformers

Introduced by Google [11], Bidirectional Encoder Representations from Transformers (BERT) has been a state-of-the-art model for a variety of Natural Language Processing (NLP) tasks, such as text classification and text generation. BERT has been popular in the NLP field as it provides contextualized embeddings, i.e., it generates embeddings that change based on the meaning/context of the sentences. BERT consists of the encoder part of the transformers (thus the name Encoder in BERT). To use BERT on a downstream task, the training is done in two phases, known as pre-training and fine-tuning.

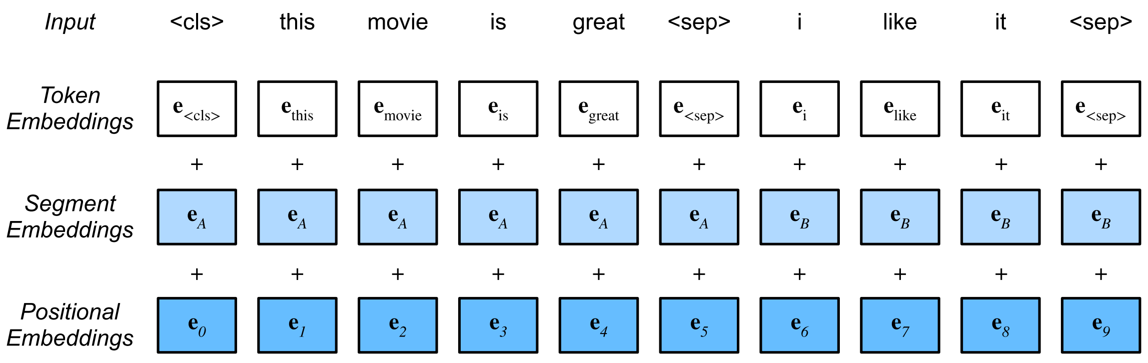
In this section, we first present the architecture adopted by BERT. Then, we explain the pre-training and fine-tuning phases of the training of BERT. Finally, we present some of the most famous variants of BERT.

1. **Architecture**:
   BERT consists of the stacked layers of the encoder part of the transformer. Before feeding the input sequence to BERT, the input text goes through three embedding layers as seen in Figure 3, mainly the following:

   (a) **Token embedding layer:** This layer tokenizes the input text (in the original paper, the tokenizer used is the WordPiece tokenizer [13]), adds a classification token [CLS] to the beginning of the tokenized text, then adds a Separator Token ([SEP]) at the end of every sentence of the input.

   (b) **Segment embedding layer:** This layer distinguishes between the different sentences in the input by returning a different embedding per sentence.

   (c) **Position embedding layer:** This layer encodes the position of each token in the sentence. This is crucial as the transformer processes all tokens in parallel and has no information on the order in the sentences.


Input
<cls>
this
movie
is
great
<sep>
i
like
it
<sep>
Token Embeddings
$e_{<cls>}$ e_{this} e_{movie} e_{is} e_{great} $e_{<sep>}$ e_{i} e_{like} e_{it} $e_{<sep>}$
Segment Embeddings
e_A e_A e_A e_A e_A e_A e_B e_B e_B e_B
Positional Embeddings
e_0 e_1 e_2 e_3 e_4 e_5 e_6 e_7 e_8 e_9
Figure 3: The input goes through three embedding layers before going through BERT[14].

2. **Pretraining BERT:**
   Pre-training BERT consists of updating the parameters of BERT by performing specific training tasks that help BERT learn how to generate semantically rich embeddings for the text. The original BERT[11] was pre-trained on the English Wikipedia dataset and the BookCorpus dataset, over two tasks: the Masked Language Modeling (MLM) and the Next Sentence Prediction (NSP) tasks. The goal of the MLM is to predict a certain masked word based on its context, while the NSP task predicts whether one sentence follows the other. Both tasks are unsupervised (i.e., require no label). We discuss these tasks briefly as follows:

   (a) **Masked Language Modeling task:** This task helps BERT learn some understanding of the syntactic, contextual, and semantic information from the corpus. It proposes to mask a certain fraction of the tokens and train BERT to predict the token that occupies the masked position, by only relying on the context of the masked token (i.e., the tokens that occur in the same input). This is done as shown in Figure 4 by feeding the output of the BERT model through a softmax layer that computes the probability distribution of the masked word.

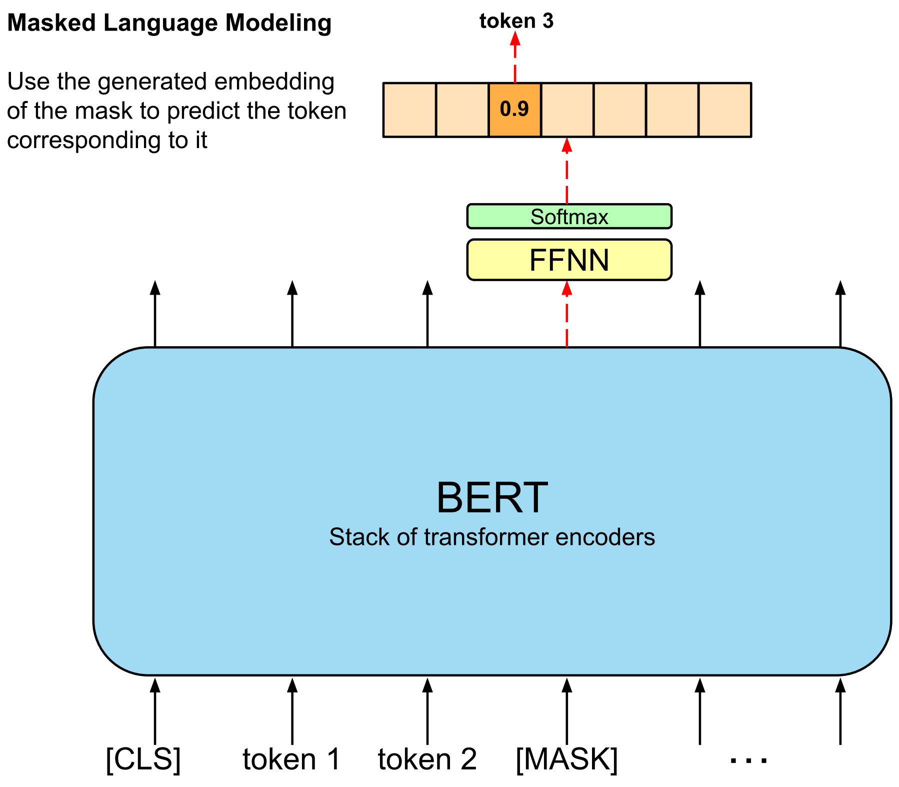

Masked Language Modeling
Use the generated embedding of the mask to predict the token corresponding to it
token 3
0.9
Softmax
FFNN
BERT
Stack of transformer encoders
[CLS]
token 1
token 2
[MASK]
...

Figure 4: Simplified example of the Masked Language Modeling task.

(b) **Next Sentence Prediction task:** This task helps BERT learn the relationships between sentences. This task is a binary classification task. As shown in Figure 5, BERT is given two sentences from the corpus, and it tries to predict whether the second sentence follows the first sentence in the corpus based on the [CLS] embeddings. The [CLS] embeddings have been used thoroughly as sentence embeddings, as they englobe information about the context and meaning of the whole input.

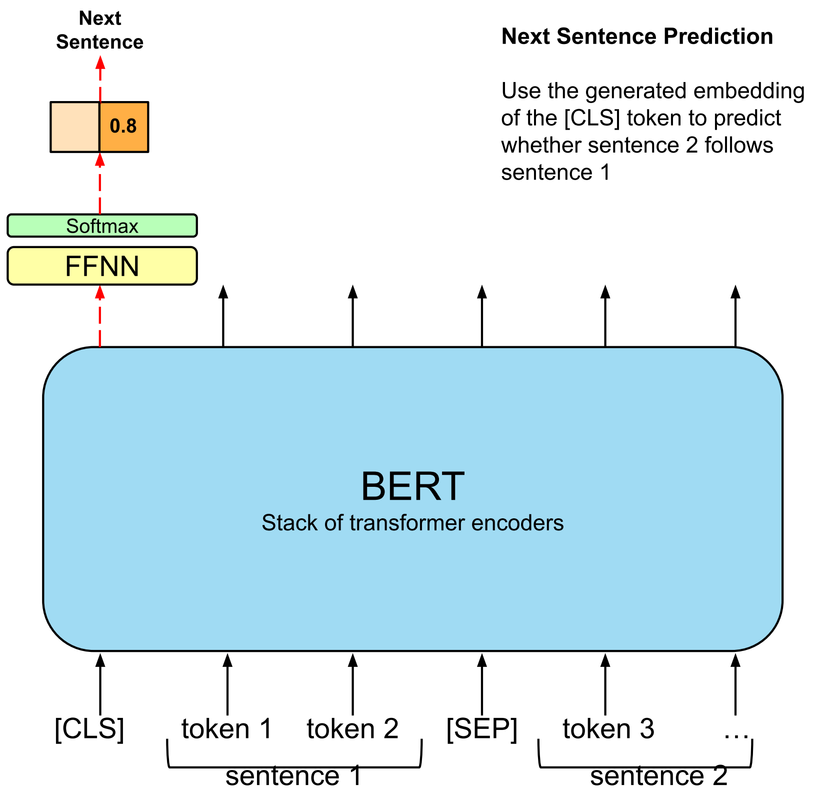

Next Sentence
Next Sentence Prediction
Use the generated embedding of the [CLS] token to predict whether sentence 2 follows sentence 1
0.8
Softmax
FFNN
BERT
Stack of transformer encoders
[CLS]
token 1
token 2
[SEP]
token 3
...
sentence 1
sentence 2

Figure 5: Simplified example of the Next Sentence Prediction task.

3. **Fine-tuning BERT:**
Fine-tuning BERT consists of taking the pre-trained BERT model as it is (same architecture and same parameter as before) and concatenating that model to downstream task layers. During fine-tuning, we proceed by feeding the text to BERT. BERT is going to generate a contextual representation of the input. After generating the representation of the text (i.e, the embeddings), we feed these embeddings to the downstream task layers and we predict the target variable in the downstream task. During training, we perform back-propagation to finetune the parameters of the whole network (i.e, we update the parameters of the downstream-task layers and BERT). The steps are shown in Figure 6.

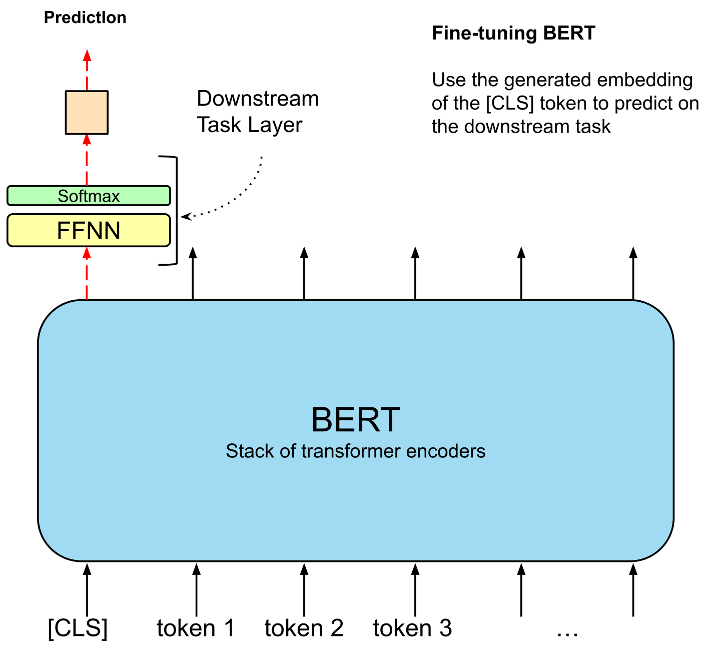

PredictIon
Fine-tuning BERT
Use the generated embedding of the [CLS] token to predict on the downstream task
Downstream Task Layer
Softmax
FFNN
BERT
Stack of transformer encoders
[CLS]
token 1
token 2
token 3
...

Figure 6: Simplified example of the fine-tuning process on a downstream task using the [CLS] embeddings.

Extracting the embedding representation of the text can be done in different ways. One of the most common methods that are mostly used for fine-tuning is to extract the output of the [CLS] token. Other strategies are briefly discussed in the next subsection. We describe the downstream task layers in Section 2.3.

Fine-tuning BERT is a direct application of Transfer Learning in NLP, as it allows researchers to choose any pre-trained version of BERT that is suitable for them (i.e., pre-trained with certain tasks on a specific domain), and fine-tune it on a dataset to solve their specific problem. Furthermore, this step allows to refine the embeddings and shifts them to the task/domain at hand.

4. **Extracting Embeddings:**
   There exist multiple ways of extracting sentence embeddings from BERT, as seen in Figure 7. Sentence embeddings can be extracted as follow:

   (a) **[CLS] embeddings**: One of the most common approaches consists of using the embedding of the [CLS] token as the sentence embeddings. The [CLS] token was originally used to indicate the start of the textual input and is used in the next sentence prediction task of the pre-training process. It represents the textual input as the embeddings generated by BERT are contextual, i.e., this embedding representation englobes some information about the other tokens in the text (i.e., represents the text).

   (b) **Average of token embeddings**: Some other works suggest producing the sentence embedding by averaging the embeddings of the tokens that occur in the input. However, the embeddings of the tokens can be obtained using several strategies. The token embedding can be:

   - The output of one of the layers of BERT (from the first layer, up to the last hidden layer).
   - The sum (or average) of the token embeddings produced by the last n layers of BERT.
   - The concatenation of the token embeddings produced by the last n layers of BERT.

   No strategy is known to be better than others. It all depends on the application of BERT (training tasks used and datasets).

5. **Variants:**
   Multiple variants of BERT have been introduced over the years. We discuss some of the most prominent variants, mainly RoBERTa[15], mBERT[11], and XLM-RoBERTa[16].

   (a) **RoBERTa**: It is a BERT model that was trained solely on the MLM task by masking the tokens dynamically during training. The training also consisted of a bigger batch size and a larger number of epochs. Furthermore, the tokenizer used is the Byte-Pair Encoding (BPE) tokenizer [17] instead of the WordPiece tokenizer. RoBERTa was able to outperform BERT considerably on many different benchmarks.

   (b) **Multilingual BERT (mBERT)**: It is a BERT model that is trained on a text of multiple languages from Wikipedia (i.e., the multilingual equivalent of BERT). To prevent the language imbalance problem of Wikipedia, small languages were oversampled and large languages were under-sampled during training.

   (c) **XLM-RoBERTa**: It is a RoBERTa model trained on a large multilingual dataset using the multilingual MLM objective (same as the MLM objective, except the sample training sentences come from corpora in different languages). XLM-RoBERTa outperforms mBERT on various benchmarks of the most common NLP tasks.

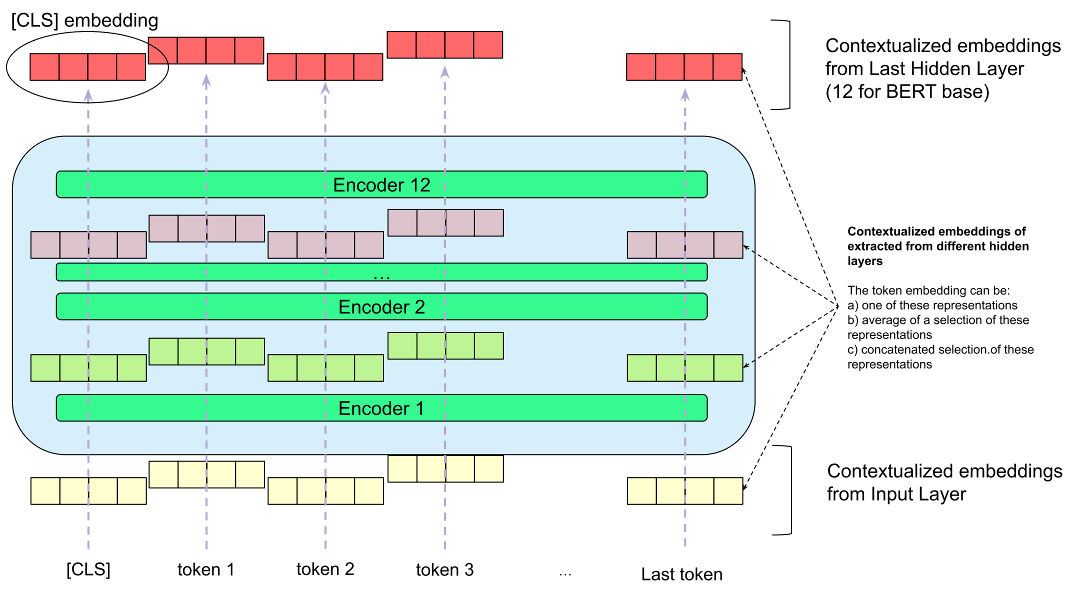

[CLS] embedding
Contextualized embeddings from Last Hidden Layer (12 for BERT base)
Encoder 12
...
Encoder 2
Encoder 1
Contextualized embeddings of extracted from different hidden layers
The token embedding can be:
a) one of these representations
b) average of a selection of these representations
c) concatenated selection.of these representations
Contextualized embeddings from Input Layer
[CLS]
token 1
token 2
token 3
...
Last token

Figure 7: Different strategies to extract the sentence embedding representations from BERT.

2.3 Downstream Task Layers

The downstream task layers accept the embeddings extracted by BERT and predict the outputs of the downstream task. Examples of downstream tasks are regression and classification. The downstream task layers are usually layers of a Feed-Forward Neural Network (FFNN). FFNNs are supervised machine learning techniques that usually accept an input vector and predict the desired output vector through a multi-connection of layers of neurons. Each neuron in a certain layer computes a weighted sum of the inputs from previous layers connected to it with an activation function, and produces a non-linear function of this sum as its output value. These networks are known as nonlinear models, which allows them to model complex relationships. The number of layers, the number of neurons per layer, the activation functions, and the activation thresholds of each neuron are all parameters that need to be properly tuned as part of the learning and decision-making processes. An example of a neural network is shown in Figure 8.

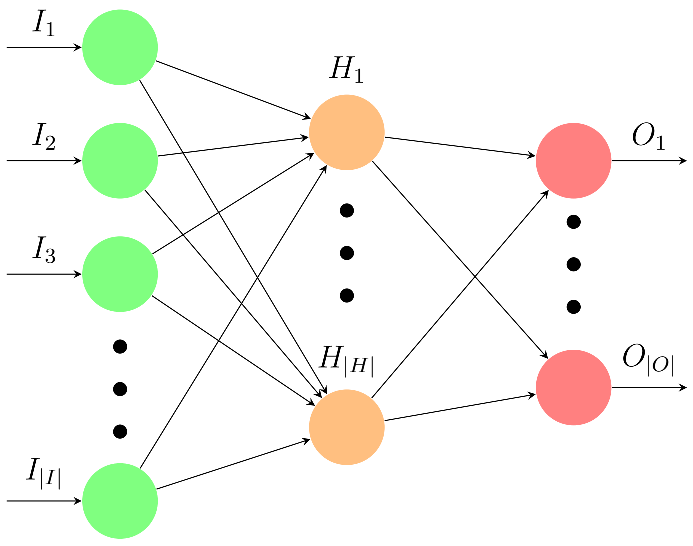


Figure 8: Simple Feed Forward Neural Network with |I| input neurons, one hidden layer of |H| neurons, and |O| output neurons.

3 Isotropy

Isotropy is a geometric property that assesses the distribution of the points in space [18]. In an ideally isotropic space, the embeddings are uniformly distributed in all directions of the space, i.e., the embeddings are not biased in a specific direction.

3.1 Properties

Isotropy has been linked to multiple properties in space. For instance, in an anisotropic space, randomly sampled words tend to be highly similar to one another when measured by cosine similarity [2]. Furthermore, the representations exhibit word-frequency bias, as the high-frequency words concentrate densely in the embedding space while low-frequency words disperse sparsely in the space [19].
Multiple studies tried to explain the source of the anisotropy. [20] showed that the anisotropy in contextual models is a product of rogue dimensions of the entire embedding space. These rogue dimensions drive the similarity metrics, explaining the high similarity property between random embeddings of these spaces. However, the authors mentioned that these models still perform well as their behavior is not greatly affected by these rogue dimensions. Their analysis showed that these dimensions handle a small subset of the model's linguistic abilities (i.e., punctuation and positional information). Furthermore, [18] showed that embeddings do not occupy a narrow cone, but rather only appear as a cone when projected to a lower-dimensional space. The authors showed that during training, word embeddings share the same direction gradients, therefore are shifted in one dominant direction in the vector space.

3.2 Measures

There is a need to approximate the degree of isotropy of the space, i.e., the spatial utilization of the embeddings. As mentioned previously, an isotropic space has embeddings that are distributed uniformly in all dimensions. In other words, the embeddings have elongations that are similar across different directions of the space. Therefore, methods that are based on the Principal Component Analysis (PCA) are the most appropriate to find and study the most elongated directions of the space. We present the two most robust PCA-based methods to quantify the isotropy, mainly the explained variance ratio and the IsoScore as highlighted in [21].

3.2.1 Explained Variance Ratio

The explained variance ratio, which we refer to as EV_k Score, measures how much total variance is explained by the first k principal components of the data. Given that λ_i is the i^{th} largest singular value of the embeddings matrix E, the variance explained ratio is computed as follows:

$$EV_k(E) = \frac{\sum_{i=1}^{k} \lambda_i^2}{\sum_{i=1}^{D} \lambda_i^2} \tag{3}$$

This metric measures the difference in variance in different directions of the embedding space. However, computing it requires the specification of a certain number of Principal Components (PCs). Therefore, in this study, we will be numerically examining this score for the first three PCs, and graphically for the top components.

3.2.2 IsoScore

The IsoScore [21] of an embedding space can be interpreted as the fraction of dimensions uniformly used by the embedding space. This score is derived from an isotropy defect that is calculated by computing the distance between the identity matrix and the normalized covariance matrix of the PCA-reoriented data. The IsoScore scales linearly with the number dimensions used and is stable when distributions contain highly isotropic subspaces. A high IsoScore (i.e., close to 1.0), indicates that the principal components are uniformly distributed across all dimensions of space, implying that the space is isotropic. However, a small IsoScore (i.e., close to 0.0) indicates that the first components explain almost all the variance of the data, implying a highly anisotropic space.

Table 1 presents a comparison between the Explained Variance and the IsoScore, as presented in [21].

Property	EV_k	**Isoscore**
Mean Agnostic	✓	✓
Scalar Covariance	✓	✓
Maximum Variance	x	✓
Rotation Invariance	✓	✓
Dimensions Used	x	✓
Global Stability	✓	✓

Table 1: Performance of current methods for measuring isotropy (i.e., spatial utilization).

3.3 Related Work

In this section, we present some related work aiming to solve the representation degeneration problem and improve the isotropy of the space. We can split the studies into two: (1) studies that regularize the embeddings during the training stage and (2) studies that post-process the embeddings after the training phase.

3.3.1 Regularizing the embeddings

Multiple studies applied regularization to improve the isotropy of the learned embeddings. Firstly, [3] employed cosine regularization to decrease the similarity between the embeddings and increase the representation power of the space. However, [22] proposed the Laplacian regularization as a better alternative to the cosine regularization, as it minimizes the similarities between the embeddings with similar contexts (instead of applying it to all embeddings). Moreover, [4] mitigated the fast singular value decay phenomenon of anisotropic space using spectrum control. Finally, [23] learned embeddings of better isotropy by alleviating the word frequency bias of anisotropic spaces using adversarial training.

3.3.2 Postprocessing the embeddings

Other studies proposed to post-processes the learned embeddings to improve the isotropy of the space. Firstly, [24] introduced the All-but-the-top method that removes the common vector and dominant directions from the embeddings, rendering them more isotropic. Secondly, [25] increased the isotropy by clustering the embeddings and nulling the principal components of each cluster. Third, [26] applied a weighted removal of a selected number of dominant directions from the embedding. The weights were learned through a word similarity task applied to the embeddings.

3.3.3 Discussion

Our method of improving the isotropy of embeddings is similar to the regularization-based approaches as it improves the embeddings through a penalty term. However, the measure used in our method is unsupervised and more robust, and it directly estimates isotropy instead of computing an indicator of isotropy. Furthermore, our method is not as computationally expensive as the regularization methods, as it only trains one feed-forward layer instead of the whole language model. Our method is also a form of postprocessing of the embeddings, as the layer introduced transforms the embeddings and makes them more isotropic, without compromising the modeling power of the space.

4 Method

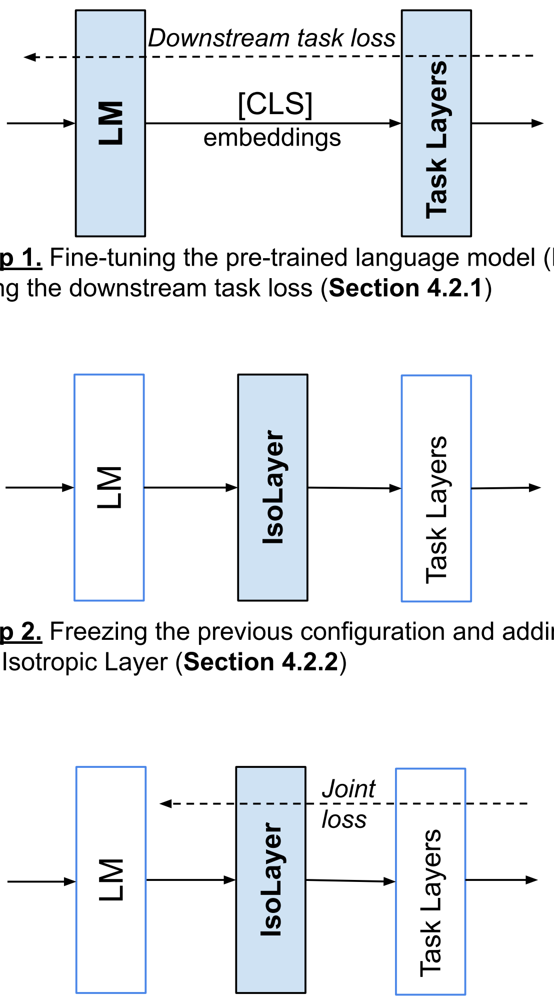
In this section, we introduce the method we have used to improve the isotropy of the embeddings. First, we remind the reader of the scope of the thesis. Second, we propose the overall approach and explain its steps in detail.

4.1 Scope

As mentioned previously, we limit the scope of our work to the embedding space of the [CLS] representations. To our knowledge, the only study besides ours which improves the [CLS] embeddings is the Isotropic Batch Normalization (IsoBN) [6]. In their study, [6] first highlighted the anisotropic nature of the [CLS] embeddings. Then, they proposed to improve the isotropy of these embeddings using an isotropic variant of the batch normalization method. Furthermore, the downstream task considered in our study is classification. However, we could generalize the method to different downstream tasks.

4.2 Proposed Approach

The proposed approach can be visualized in Figure 9. In summary, the approach consists of adding a layer, referred to as the Isotropy Layer, between the data representation module (i.e., the pre-trained language model BERT) and the downstream task layers. This Isotropy Layer is responsible for transforming the [CLS] embeddings of BERT into embeddings with a better isotropic property. Since the goal of the study is to improve the isotropy of the space to perform better on the downstream task, we condition the learning process by freezing the parameters of the downstream task layers. In other words, the parameters of the Isotropy Layer are learned to output embeddings that are isotropic, without losing the knowledge acquired to solve the downstream task.
Therefore, we proceed by freezing both BERT and the downstream task layer, and only training the Isotropy Layer to optimize two metrics, mainly the isotropy score of the space and the performance metric of the entire model. Since only this layer is updated during training, we are learning a clear transformation of the space; this transformation post-processes the embeddings to improve the distribution of the embeddings in the space while keeping the semantics needed to perform the downstream task. The process can be summarized in Figure 9. We present the details of the approach in the following subsections.



<u>Step 1.</u> Fine-tuning the pre-trained language model (LM) using the downstream task loss (**Section 4.2.1**)

<u>Step 2.</u> Freezing the previous configuration and adding the Isotropic Layer (**Section 4.2.2**)

<u>Step 3.</u> Training the Isotropic Layer using the Joint Loss (**Section 4.2.3**)

Figure 9: Diagram summarizing the approach described in the thesis.

4.2.1 Fine-tuning the language model using the downstream task loss

Our proposed approach assumes that the pre-trained language model is first fine-tuned on the downstream task. This is done by extracting the [CLS] embeddings from the BERT-based model and feeding these embeddings to a predefined set of downstream task layers. Then, both the pre-trained language model and the downstream task layers will be trained to perform the downstream task. Fine-tuning the model on a downstream task allows us to leverage the knowledge encoded in this model, and transfer it to perform the task at hand.

4.2.2 Freezing network and adding Isotropy Layer

We insert one feed-forward layer, called the Isotropy Layer, between the fine-tuned model and the downstream task layers. The goal of the Isotropy Layer is to transform the [CLS] embeddings outputted by the fine-tuned model to a new space that is more isotropic.

It should be noted that this layer could be replaced by a more complex neural network, with the only limitation that the output of this neural network is of the same size as its input (i.e., the embedding vector outputted by the pre-trained model), to ensure compatibility of the output of the network with the downstream task layers. We refer to it as Isotropy Layer instead of Isotropy Layers since we only used one layer in this study.

As mentioned previously, improving the isotropy by itself is not sufficient [5]. Therefore, the Isotropy Layer needs to maintain the semantics needed to perform the downstream task at hand. To do so, we perform the following:

- We freeze the parameters of the fine-tuned model. As we know, the fine-tuned model has learned some part of the knowledge required to perform the downstream task. Freezing this fine-tuned model preserves the knowledge learned.
- We freeze the parameters of the downstream task layers. As mentioned previously, the output of the Isotropy Layer (i.e., the transformed embeddings) needs to maintain the semantics needed to perform the downstream task. Therefore, we freeze these layers to condition the output of the Isotropy Layer to adjust to the knowledge of this layer during the training.

4.2.3 Training the Isotropy Layer using a Joint Loss

- **Joint Loss:**
  Now that the fine-tuned model and the downstream task layers are both frozen, we train the Isotropy Layer using the proposed joint loss function:

$$\mathcal{L} = \alpha \times \mathcal{L}_1 + (1 - \alpha) \times \mathcal{L}_2 \tag{4}$$

  We define the variables in the equation as the following:

  1. $\mathcal{L}_1$ is the loss used to fine-tune the pre-trained model. It varies with the downstream task (i.e., CrossEntropy for the classification, Mean Squared Error for regression tasks). This measure is computed over the output of the new network, i.e., the embeddings are fed to the pre-trained model, transformed through the

Isotropy Layer, then go through the downstream task layers to generate an output. This term ensures that the transformed embeddings maintain the semantic information required for the downstream task.

2. $\mathcal{L}_2$ quantifies the degree of the isotropy of the space of embeddings at the output of the Isotropy Layer. It is an unsupervised measure computed over a mini-batch of embeddings at a time. Intuitively, the bigger the batch of embeddings, the more accurate the isotropy measure is. This term acts as a regularizer for the new embeddings, and it pushes the weights of the Isotropy Layer to produce more isotropic embeddings.

   We propose to use the IsoScore as the measure of quality used to compute $\mathcal{L}_2$. We usually desire to optimize a decreasing function. Knowing that the IsoScore increases for a better isotropic space, we propose to use the logarithmic of the inverse of the IsoScore. This decision was taken due to its sensitivity to the change in the measure used (i.e., IsoScore).

- **Balancing both losses**
  Ideally, setting α to 0.5 should result in equally acceptable isotropic property and downstream task performance. However, both losses in the joint loss operate around different scales. Therefore, the learning process might be biased toward one of the losses and might optimize one of the losses at the expense of the other one. Inspired by the work done by [27], we overcome this issue by normalizing the losses by computing a Moving Average as follows:

  1. We keep track of the set of mean values of each loss $\overline{\mathcal{L}}_1$ and $\overline{\mathcal{L}}_2$, as well as the average $\overline{\mathcal{L}}$ of both means.
  2. We estimate the set of normalized losses as $\hat{\mathcal{L}}_i = \mathcal{L}_i \frac{\overline{\mathcal{L}}}{\overline{\mathcal{L}}_i}$ where $i \in \{1, 2\}$.
  3. At iteration k of the learning process, we update the set of mean values using exponential moving average, where s is the smoothing factor (set as 0.15, as done in [27]).

  $$\overline{\mathcal{L}_i^{new}} = \mathcal{L}_i \times \frac{s}{1+k} + \overline{\mathcal{L}}_i \times (1 - \frac{s}{1+k}) \tag{5}$$

  4. Therefore, the loss computed for the training is $\hat{\mathcal{L}} = \frac{1}{2}(\hat{\mathcal{L}}_1 + \hat{\mathcal{L}}_2)$. This loss $\hat{\mathcal{L}}$ is used to optimize the parameters of the Isotropy Layer.

5 Experiments

To evaluate the proposed method, we have conducted two main experiments. The first experiment evaluates the proposed method and compares it to our baseline (i.e., the text classification system without the Isotropy Layer) in terms of IsoScore, Explained Variance, and performance measurement. The second experiment compares the proposed method with the IsoBN method [6] on three GLUE benchmark datasets.

5.1 Experimental Setup

Firstly, we present the architectures of the BERT-based models used. Secondly, we introduce the datasets used in our experiments. Thirdly, we present the hyperparameter tuning strategy that we have employed as well as the set of hyperparameters tuned before the training. Finally, we report the average training and inference time per model per dataset. In the upcoming sections, we will refer to our work as IsoLayer for conciseness.

5.1.1 Models

The models were implemented using the transformers library provided by HuggingFace[28]. The optimizer used is AdamW[29]. Furthermore, the approach was implemented using PyTorch. Early stopping was applied according to task-specific metrics on a validation set (train/validation split of 70/30). The approach is evaluated on two pre-trained language models (BERT-base-cased and RoBERTa-large) on the dev sets that were provided by the datasets [1]. We experiment with the following modules in our Text Classification System:

- **Data Representation Module:** We evaluate our approach on two pre-trained language models (BERT-base-cased and RoBERTa-large). We chose the same models as IsoBN (to be able to compare our results with theirs). Furthermore, these models seemed appropriate since they have good performance for the English language (all datasets used are in English). It should be noted that we have applied the method on mBERT and XLM-RoBERTa on internal datasets at Huawei. However, the results will not be disclosed in this thesis.

[1]The labels of the test sets were not provided (they are only evaluated through the leaderboard at https://gluebenchmark.com/leaderboard). This setup also follows the work done by IsoBN.

- **Isotropy Layer:** This layer is a one-layer feed-forward neural network. The number of neurons at the output is equal to the number of neurons at the input of this layer, which is the same as the size of the [CLS] embedding extracted from the data representation module (768 in the case of BERT and 1024 in the case of RoBERTa). We chose to only have one layer for simplicity. However, a more complex neural network might result in better results. We leave this direction for future studies.
- **Downstream Task Layers:** This module differs from dataset to dataset. However, all the datasets that we use are binary classification datasets. Therefore, the downstream task layers consist of only one classification layer of 2 neurons. The activation function used is the softmax function and the loss used is the binary cross-entropy loss.

5.1.2 Datasets

To evaluate our approach, we utilized multiple datasets from the General Language Understanding Evaluation (GLUE) benchmark [30]. The GLUE benchmark is a collection of Natural Language Understanding (NLU) tasks including question answering, sentiment analysis, and textual entailment. GLUE datasets favor models that learned to represent linguistic knowledge to favor sample-efficient learning and knowledge transfer across tasks [30]. Each dataset has its metric to evaluate the model performance. We selected the three specific datasets described in Table 2 because they cover the three main tasks of the GLUE benchmark, which are Inference Tasks(RTE), Single-Sentence Tasks (CoLA), and Similarity and Paraphrase Tasks (MRPC).

Dataset ID	**Dataset Name**	**Dataset Description**	**Metric**	**Training Samples**
RTE	Recognizing Textual Entailment	Determines whether each sentence entails a given hypothesis or not	Accuracy	2.5k
CoLA	Corpus of Linguistic Acceptability	Determines whether each sentence is grammatically correct or not	Mathew's correlation coefficient	8.5k
MRPC	Microsoft Research Paraphrasing Corpus	Consists of a pair of sentences and determines whether the sentences are paraphrases from one another or not	Accuracy	3.7k

Table 2: Details of the datasets used in the experimental evaluation.

5.1.3 Hyperparameter Tuning

Hyperparameters need to be set before the training stage. These parameters are usually set through a process called hyper-parameter optimization. We employ random search [31] for 10 trials to find the best set of hyper-parameters. We perform this step a first time to fine-tune the pre-trained language model used, and a second time to train the Isotropy Layer. Table 3 presents the hyperparameters that we chose to optimize, the search space, as well as their values per dataset and language model for the Isotropy Layer training (we do not report the parameters used for fine-tuning). Increasing the number of trials should find better hyperparameters. We also argue that better parameters can be found for RoBERTa.

	Hyperparameter	**Search Space**	**Optimal Parameter**		
			RTE	**CoLA**	**MRPC**
BERT-base	Learning Rate	[1e-7, 1e-3]	1e-4	1e-5	1e-4
	Batch Size	{8, 16, 32}	8	8	8
	Number of epochs	{5, 10}	5	5	5
RoBERTa-L	Learning Rate	[1e-7, 1e-3]	7e-4	2e-4	6e-7
	Batch Size	{8, 16, 32}	16	16	8
	Number of epochs	{5, 10}	10	5	5

Table 3: Hyperparameters optimized and used to train the Isotropy Layer.

5.1.4 Training and Inference Time

In this section, we report the average training and inference run times for the models that we have trained (displayed in Table 4). We should note that the training time corresponds to the training time of the Isotropy Layer only (we exclude the training time for fine-tuning from the results). One could avoid fine-tuning a model on a certain dataset by simply loading the appropriate fine-tuned model from HuggingFace[28]. We can see that the run time of RoBERTa is larger than the run time of BERT, which can be explained by the fact that RoBERTa's Isotropy Layer has more parameters (RoBERTa's embedding is larger).

	RTE		**CoLA**		**MRPC**	
	Training Runtime	**Inference Runtime**	**Training Runtime**	**Inference Runtime**	**Training Runtime**	**Inference Runtime**
BERT-base	171.6	1.9	411.1	6.7	395.1	3.2
ROBERTA-L	2005.8	5.3	2760.4	16.9	1205.4	12.8

Table 4: Training and Inference run time of the Isotropy Layer for different language models and different datasets. The run time is reported in seconds.

5.2 Comparing the proposed approach to the baseline

5.2.1 Model performance and IsoScore

We proceed by selecting a random seed and then running our approach once per dataset, per language model. We evaluate the performance of the model using the metric associated with the selected dataset. Then, we evaluate the isotropy of the transformed embedding space using the IsoScore. These measures have been computed over the dev set (which did not participate in the training process). Results are displayed in Table 5.

Dataset	Method	Perf.	Isoscore
RTE	BERT-base	67.87	0.0051
	IsoLayer	71.84	0.025
	Improvement	**+5.8%**	**+390.2%**
	RoBERTa-L	85.56	0.0049
	IsoLayer	87.36	0.0145
	Improvement	**2.1%**	**195.9%**
CoLA	BERT-base	61.61	0.004
	IsoLayer	63.57	0.0255
	Improvement	**+3.2%**	**+537.5%**
	RoBERTa-L	67.23	0.0012
	IsoLayer	68.77	0.0023
	Improvement	**2.3%**	**+91.67%**
MRPC	BERT-base	85.29	0.0033
	IsoLayer	87.99	0.0103
	Improvement	**+3.2%**	**+221.2%**
	RoBERTa-L	90.93	0.0016
	IsoLayer	91.17	0.00245
	Improvement	**+0.26%**	**+53.13%**

Table 5: Internal evaluation of the approach on 3 GLUE benchmarks with random seeds over one run. We can observe a huge increase in the isoscore and a notable increase in the downstream task performance.

We can observe that the proposed approach results in improvements in terms of the performance of the classifier and isotropy of the embeddings. This experiment suggests that improving the isotropy of the space increases the expressiveness of this space, resulting in an improvement in the performance of the model on the selected downstream tasks.

5.2.2 Explained variance

We examine the explained variance curve of the models trained by computing the metric over the top K=20 principal components. The results are displayed in Figure 10. We can see that using the Isotropy Layer resulted in embeddings with a smaller explained variance compared to the baseline. This means that the singular values distribute more uniformly in the transformed space, inferring that the information is spread across more principal components uniformly (the space is more isotropic).

We also notice that the proposed approach had limited improvement in explained variance for RoBERTa, compared to BERT. We provide multiple explanations in the discussion section for such behavior.

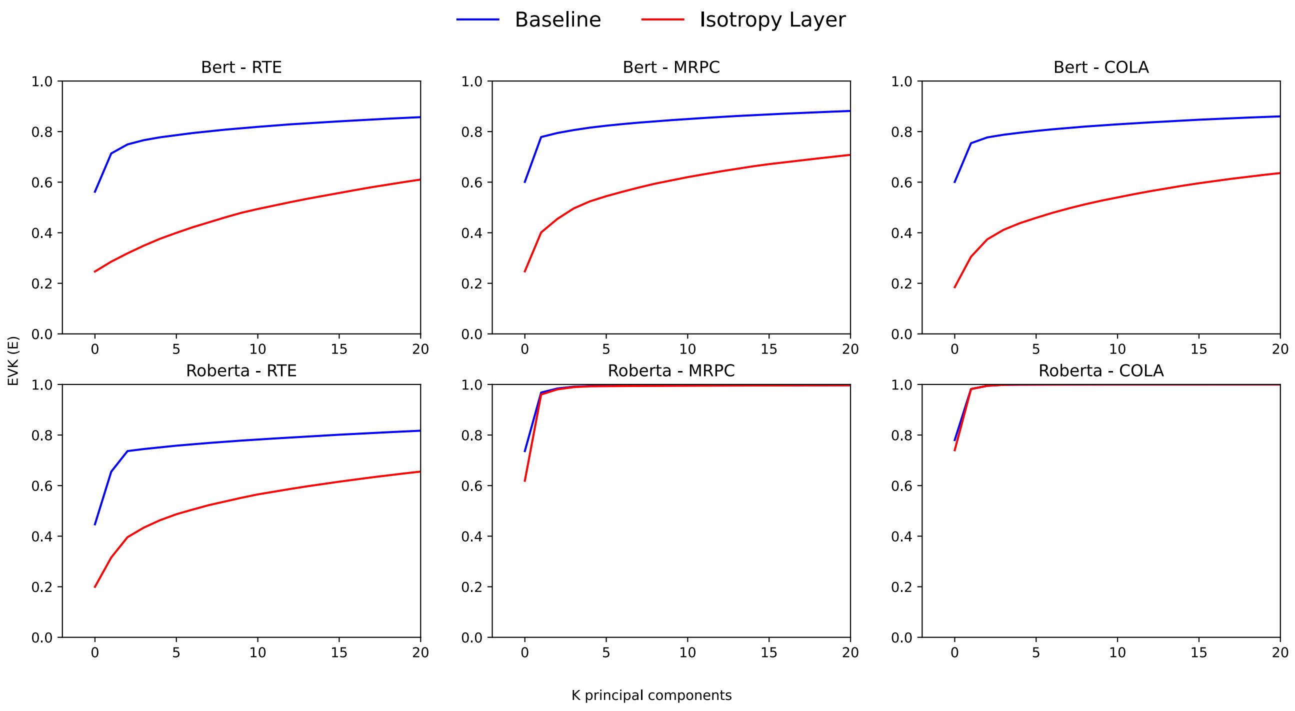


Figure 10: The plots present the explained variance of the top principal components on 3 selected datasets. These plots show that the proposed method results in a smaller explained variance compared to the baseline, which indicates that the variations of the embeddings tend to distribute equally in all directions (i.e., more isotropic space).

5.3 Comparing the proposed approach with IsoBN

We compare the proposed approach to IsoBN, the only approach in the literature besides ours that aims to improve the isotropy of the [CLS] embeddings. To prove that the improvements incurred by our approach are consistent, we run the approach on each dataset with 5 random seeds. Furthermore, we measure the isotropy by examining the explained variance of the top 3 principal components. As we can see, our approach provides consistent improvements in both performance and isotropy. We notice that our approach is on par with IsoBN in terms of model performance. As for the isotropy, we can see that our approach results in better-explained variance for all BERT models, while IsoBN results in better explained variance for the RoBERTa models. This is interpreted in the discussion section.

Dataset	**Method**	**Performance Measure**		**Isotropic Measure**	
		IsoBN	**IsoLayer**	**IsoBN**	**IsoLayer**
RTE	BERT-base	67.87 (1.1)	67.72 (0.83)	0.88/0.93/0.95	0.61/0.75/0.78
	BERT-base+Isotropy	70.75 (1.6)	70.40 (1.05)	0.49/0.72/0.85	0.27/0.35/0.38
	RoBERTa-L	84.47 (1.0)	85.56 (0.8)	0.53/0.66/0.70	0.44/0.65/0.73
	RoBERTa-L+Isotropy	87.00 (1.3)	87.36 (0.6)	0.15/0.29/0.37	0.19/0.31/0.39
CoLA	BERT-base	60.72 (1.4)	60.89 (0.81)	0.49/0.58/0.64	0.60/0.75/0.77
	BERT-base+Isotropy	61.59 (1.6)	62.82 (0.85)	0.25/0.37/0.48	0.18/0.30/0.32
	RoBERTa-L	68.25 (1.1)	67.33 (0.9)	0.83/0.88/0.90	0.66/0.87/0.91
	RoBERTa-L+Isotropy	69.70 (0.8)	68.77 (0.8)	0.21/0.38/0.49	0.41/0.63/0.76
MRPC	BERT-base	85.29 (0.9)	86.64 (1.09)	0.76/0.87/0.89	0.63/0.80/0.81
	BERT-base+Isotropy	87.5 (0.6)	87.5 (0.42)	0.37/0.68/0.77	0.43/0.49/0.52
	RoBERTa-L	90.68 (0.9)	90.93 (0.2)	0.86/0.90/0.91	0.73/0.96/0.98
	RoBERTa-L+Isotropy	91.42 (0.8)	91.17 (0.3)	0.18/0.36/0.43	0.61/0.96/0.98

Table 6: Results on the dev sets of selected GLUE tasks after running 5 times with different random seeds. For the performance measures, we report the median and standard deviation over the 5 models. As for the isotropy measure, we report the explained variance of the model that exhibits median EV1. Results from IsoBN have been extracted from [6].

5.4 Discussion

5.4.1 BERT vs RoBERTa

We notice that the improvements on BERT were higher than the improvements in RoBERTa in terms of performance and isotropy. Finding the exact source of this discrepancy is outside the scope of this thesis and is left for future work. However, we provide some hypotheses that could explain the observed behavior. One source of this discrepancy can be attributed to the highly anisotropic nature of the RoBERTa embedding space; Figure 10 shows that most of the variance is concentrated in the top 5 principal components (unlike BERT, where the variance is more distributed among the PCs). Furthermore, RoBERTa is more complex than BERT (larger architecture). Therefore, a solution worth exploring is to increase the complexity of the Isotropy Layer (i.e., replacing the one feed-forward neural network with a deeper network that can learn a more complex transformation resulting in better embeddings). Another source of the discrepancy observed could be due to the strict constraint imposed by the downstream task layers (that are frozen). In other terms, the downstream-task layers might rely heavily on the information encoded in the top principal components. A solution worth exploring in future work is to retrain the Isotropy Layer on the improved embeddings, fine-tune the downstream task layers on these transformed embeddings, and repeat both steps until convergence. Another potential solution is to jointly train both the Isotropy Layer and the downstream-task layers with the joint loss function.

5.4.2 IsoLayer vs IsoBN

We pinpoint an interesting analogy between both approaches; the IsoBN approach employs an isotropic batch normalization to regularize the embeddings, while our method learns a transformation that adds an isotropic penalty term to regularize the embeddings. We should note that the IsoLayer method trains only one feed-forward neural network, while the IsoBN method performs the training for the whole network. A disadvantage of our method is that the performance is highly constrained by the downstream task layers. Perhaps, a more isotropic embedding space with better semantic properties can be reached with a different downstream task network.

6 Case Study - Abusive Language Detection

Abusive language (also known as hate speech) has seen a significant rise in the last decade. Consequently, researchers have worked on limiting the impact of abusive statements by building Automatic Abusive Language Detection models to solve this problem. In this section, we first define what we mean by Abusive Language and Abusive Language Detection. Second, we present the technical details of the applied IsoLayer method for this problem. Finally, we introduce a discussion covering the ethical concerns of Abusive Language Detection. This case study is provided for completeness (the work performed at Huawei was conducted on the abusive language detection task).

6.1 Abusive Language

Abusive language has appeared under different terms in the literature. It was mentioned as hate speech [32], insult [33], and racism [34]. Most studies define abusive language as using of hateful and hostile language to provoke, intimidate, display contempt for, or harm a targeted entity, which could be a single person or a group of individuals that share a membership (such as race, religion, sexual orientation), The plethora of studies covering the topic highlights the need for a more specific and nuanced definition of abusive language. For instance, certain statements classified as hate speech in [35] would not satisfy the definitions for this category in the work performed by [36]. This shows the importance of defining the scope of the research to unify the work done by researchers. In this study, we adopt the same definition as before for abusive language.

6.2 Abusive Language Detection

Usually, the victims of language abuse on social media are from the most vulnerable groups in society, such as people with disabilities or members of racial and ethnic minorities. Hateful statements can spread hate which might cause psychopathological symptoms of anxiety, depression, and social phobia that can lead to extreme acts, such as self-harm and death[37]. In other cases, abusive language can lead to violent occurrences in real life motivated by hatred. To protect the safety of the users online and offline, social media platforms need to ensure that the users do not use abusive language. Traditionally, these platforms employed human moderators to flag inappropriate content. However, these human moderators are unable to keep up with the volume of content posted to the platforms. Furthermore, as a result of daily exposure to harmful content, moderators go through burn-out, depression, and post-traumatic stress disorder [38, 39].

Therefore, there is a need for a mechanism that reduces the amount of work for moderators. This is referred to as Abusive Language Detection. Abusive language Detection is defined as a supervised classification problem. The main task of the classification varies between the systems, as the system could detect whether a text is abusive or not, the category of the abuse, or even the target of the abuse.

6.3 IsoLayer for Abusive Language Detection

Abusive Language Detection can be modeled as a **Text Classification** problem, where the social media post is inputted into the model and the degree of abusiveness is outputted by the model. As this is a text classification problem, we can apply the method proposed in this thesis. This section presents the experiments conducted.

6.3.1 Dataset

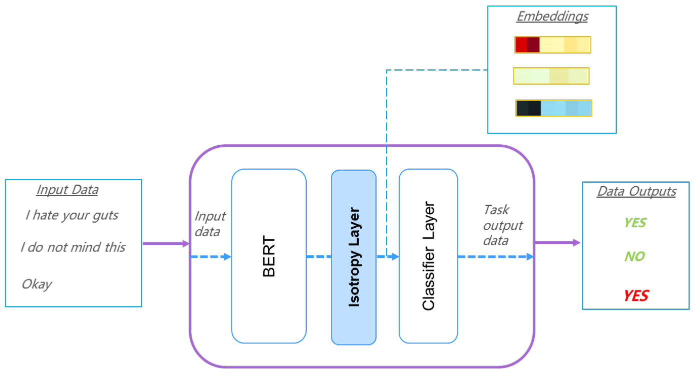
In this study, we employ the Offensive Language Identification Dataset, known as OLID[40]. This dataset has been introduced in the OffensEval 2019 challenge, in which participants were asked to perform three tasks. In this study, we experiment with Task A from the challenge, i.e., the Offensive Language Detection task.

6.3.2 Method

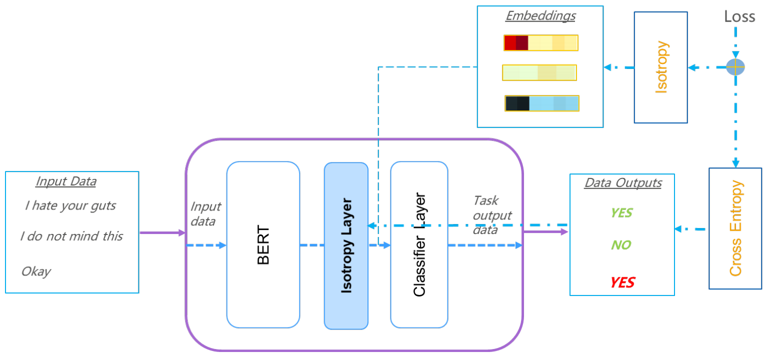
We provide a simple visual example of the method proposed previously with a special application on Abuse Language Detection (i.e., Profanity Detection). As mentioned already, we proceed with a text classification system consisting of two modules, mainly BERT and the classifier layer. BERT accepts the textual input and generates embeddings. These embeddings are then fed to the classifier layer that outputs whether the text is abusive or not. Furthermore, the goal of the approach is to improve the isotropy of the textual embeddings to improve the performance of the whole text classification system.

As proposed in the approach, we describe the contents of Figure 11 as follows:

- We proceed by first adding the Isotropy Layer between the output of BERT and the input of the classification layer. The layer is responsible for transforming the embeddings outputted by BERT to more isotropic embeddings, while maintaining the semantic information required to perform the task at hand.



(a) Adding the Isotropy Layer between the finetuned BERT and the classifier layer.



(b) Training by only updating the weights of the Isotropy Layer by computing a joint loss.

Figure 11: Visual example of the approach for abusive language detection.

- There is a need for a measure that assesses both the quality of the isotropy of the embeddings and the performance of the system. Therefore, we use the IsoScore as the basis for the loss used to assess the isotropy of the embeddings outputted by the Isotropy Layer, and the cross-entropy Loss as the loss used to assess the performance of the system (since we are dealing with binary classification). During the training, we feed multiple batches of textual documents to the new classification system. For each batch of textual input, we can generate a batch of embeddings from BERT and a batch of transformed embeddings from the Isotropy Layer as well as a batch of predictions. We compute the isotropy loss on the transformed embeddings and the cross-entropy loss on the predictions, balance both losses, then perform backpropagation and only train the Isotropy Layer. This will push the weights of the Isotropy Layer to tend to generate embeddings that are more isotropic without losing the semantics needed to perform the binary classification.

6.3.3 Experimental Evaluation

1. **Experiment A:**
   We apply the approach using bert-base-case, with one Isotropy Layer and one downstream-task layer. We perform hyperparameter tuning as we did in the previous experiments. We first fine-tune BERT, freeze the setup and then train the Isotropy Layer. We compare our model to the baseline (i.e., the same setup without the Isotropy Layer). The metric used to evaluate the performance of the model is the F1 Score, while the metrics used to evaluate the isotropy of the embeddings are the explained variance and the IsoScore. Results are shown in Table 7. Table 7 shows that applying our approach improves the isotropy of the space, as well as the performance of the model in the case of abusive language detection.

Approach	**EV1**	**EV2**	**EV3**	**IsoScore**	**F1 Score**
Baseline (fine-tuned BERT)	0.8	0.76	0.55	0.0043	73.87
IsoLayer (our approach)	0.65	0.63	0.59	0.0101	75.01

Table 7: Comparison between the approach and the baseline in terms of isotropy and model performance.

2. **Experiment B:**
   To investigate the results even more, we select the sentences from the test set where the predictions of the baseline and the proposed approach are not the same. We display these sentences in Table 8. As we can see from the table, the sentences that are no longer considered offensive contains some sensitive terms that might infer some degree of offensiveness. We can imply that improving the isotropy resulted in a better understanding of the semantics of the text. This observation can be linked to the word frequency bias associated with anisotropic spaces. In fact, having a more isotropic space alleviates the word frequency bias of the model and allows BERT to model the sentences more properly. Therefore, we can imply that the approach allowed to effectively leverage the contextual understanding ability of BERT.

Sentence	OFF
@USER #Zuckerberg lies.	0
#SerenaWilliams Serena was caught cheating with her coach but because it's collaborative she thinks she is not a thief. In her mind the real thief is the umpire who spotted what was going on and called it out. It's like a couple of bank robbers blaming a guard for calling police	0
@USER Let's just be friends and have a civil discussion." So it's totally fine if he is told to his face that his life is sinful. But criticize him ever so lightly and that is not civil anymore, which deserves a block!"	0
@USER Yes, we are supposed to believe that Kavanaugh, who seemed to be a heavy drinker in high school, has perfect memory of everything he did, while she is mixed up." Blame the victim. He has been equivocating/misrepresenting himself, but he should be believed. Not! #StopKavanaugh"	0
#NEWS Jeff Sessions: If you want more death, 'listen to the ACLU, Antifa, Black Lives Matter'" URL #CNBC"	1
#Ford's attorney says she's receiving death threats and has had to move her family to safety". Where's the proof that #Democrats #Antifa are not doing this to promote this circus?"	1
#KendallJenner looks amazing!!! Who cares if she is nude! Not everyone likes nudity for sexual desire. Some like it for the artistry in it. #me @USER Every woman on the planet has the same parts. She is just comfortable with hers. #YouGotThis	0
@USER Laws are for the law abiding citizens. We have the second amendment. We don't need more gun control. Criminals don't care they will use guns until the end of time! Figure out how to stop them from using guns!	0

Table 8: The sentences that were not correctly predicted by the baseline, but that were correctly predicted by the improved model. Offensive sentences have label 1.

3. **Experiment C:**
   Finally, we investigate some of the design decisions in our approach in the case of abusive language detection. We define different setups, where every setup represents a variant of the proposed method, differing by the modules that we choose to train as well as the loss function used. We use the same hyperparameters for all training experiments. Furthermore, we compute the IsoScore on the embeddings outputted by the IsoLayer (if not present, the IsoScore is computed on the embeddings outputted by BERT). Table 9 presents the different setups that we experimented with. Let's analyze the results:

   - **Setups 1, 2, and 3:** Setup 1 corresponds to the baseline from Experiment A. We freeze the parameters of BERT from setup 1, then add the isotropy and the classification layers, and train these layers using the joint loss function (setup 2). Afterward, we freeze the parameters of both BERT and the classification layer from setup 1 and train the Isotropy Layer using the joint loss function (setup 3). Both setups 2 and 3 achieve a better performance in terms of IsoScore and F1 score compared to setup 1. We can observe that setup 2 generates embeddings of better isotropy, while setup 3 performs a more accurate classification. We can infer that freezing the classification layer and training the Isotropy Layer result in embeddings that are finely tailored toward the downstream task. However, freezing the classification layer adds a constraint on the improvements that can be achieved in terms of isotropy of the embeddings. This is aligned with the approach presented in this thesis.

- **Setups 4, 5, and 6:** In setups 4, 5, and 6, we train different modules in the system using the cross-entropy loss as shown in Table 9. The three setups generate embeddings that have a low IsoScore and do not achieve significant performance gain compared with the baseline (setup 1). This experiment confirms that the improvements observed in the approach proposed in this thesis are a result of improving the isotropy of the embeddings using the novel joint loss function.
- **Setups 7, 8, and 9:** In setups 7, 8, and 9, we train different modules in the system using the joint loss (when possible) as shown in Table 9. This experiment shows that the improvements incurred by these variants are not as significant as the improvements incurred by our approach. We can clearly see that it is better to fine-tune BERT using the cross-entropy loss before adding the Isotropy Layer.

Setup	**BERT**	**IsoLayer**	**Class. Layer**	**Loss**	**IsoScore**	**F1 Score**
1	✓	–	✓	CE	0.0043	73.87
2	(1)	✓	✓	Joint	**0.0445**	74.94
3	(1)	✓	(1)	Joint	0.0101	**75.05**
4	✓	✓	✓	CE	0.0036	72.87
5	(4)	✓	✓	CE	0.0030	73.79
6	(4)	(5)	✓	CE	0.0030	74.06
7	✓	✓	✓	Joint	0.0045	73.86
8	(7)	✓	✓	Joint	0.0064	73.97
9	(7)	(8)	✓	CE	0.0064	73.66

Table 9: Results of the different setups considered. We use ✓to indicate that the module is trained, − to indicate that the module is absent from the system, and (setup number) to indicate that the module was taken from a certain setup and its parameters were frozen. The losses that we experimented with are the cross-entropy loss (CE) and the joint loss.

6.4 Ethical Considerations

We present our own analysis of the abusive language detection systems in the light of the ethical issues of the Artificial Intelligence (AI) Systems highlighted in [41]. Furthermore, we provide recommendations to ensure that the system does not violate the most important ethical values. We should note that this section discusses the ethical concerns of Abusive Language Detection systems in general (we discuss ethical concerns that are specific to our solution in Section 7). We present the following concerns and recommendations:

1. **Privacy and Data Protection:** To perform well, the models must be trained on real-world data. Since the real-world data involves the user's social media posts, the user's informational privacy must be protected to ensure their confidentiality. This may be accomplished by:

    - Using privacy-preserving methods [42] on the data (i.e., removal of personal information such as usernames and addresses) and the models (i.e., federated learning [43]).
    - Securing the model and infrastructure to assure security, prevent data leaking [44], and protect against adversarial attacks [45].
    - Obtaining the user's permission to include the data in the training and allowing the user to withdraw their permission.

2. **Reliability and Integrity:** The system developed needs to be reliable. This may be accomplished by:

    - Ensuring that the training set represents and fulfills the particular task of the abusive language model (the data should cover different domains and include implicit and explicit samples).
    - Identifying the system vulnerabilities [46] by running the model in a real-world setup (testing outside the experimental setting).

3. **Accountability and Explainability:** There is a need to monitor the abusive language detection system and hold it accountable for its decision. This can be done by:
    - Ensuring that the system's decision is explainable to the user by using explainable models or tools that provide interpretability [47].
    - Ensuring that the system's outcome is not definitive; in other words, if the system detects that a user post is abusive, the user should be allowed to appeal that decision (the moderator can audit the system and review the appeals, providing a human entity in the system and job security despite the automation).
    - Enforcing cyber laws that prohibit abusive online activity, and involving the authorities if needed.
4. **Moral and Ethical Responsibility:** The final point heavily relies on the integrity of the creators of the Abusive Language Detection system. In fact, the system creators need to ensure moral and professional responsibilities as follows:
    - Ensuring that the deployed system fulfills the performance demands.
    - Updating the system to reflect the ongoing changes in the definition of abusive language (i.e., what was considered to be abusive a century ago is different than what is abusive now).
    - Maintaining the greatest level of integrity, by informing themselves of the dangers of misusing the system. All systems have the potential to cause harm in the wrong hands. Abusive language detection might be potentially misused for content moderation. Content moderation restricts the user's freedom of speech, as well as the user's freedom of looking at content (content moderation alters the user's perception of the social media network). Therefore, the system needs to strike the necessary balance between the human right to freedom and the human right to safety/security.

7 Conclusions and Future Work

7.1 Conclusions

As mentioned previously, BERT-based models suffer from the problem of having an anisotropic embedding space. This affects the representation capacity of the embedding space and affects the accuracy of the downstream tasks. In this thesis, we proposed to learn an embedding transformation that improves the isotropy of the [CLS] embeddings by adding an Isotropy Layer at the output of the fine-tuned language model and only training this layer using a joint loss function. Once trained, the layer will output transformed embeddings of better statistical properties that result in a better performance on the downstream task. We applied empirical methods to quantitatively measure the improvement of the models in terms of isotropy and performance on the downstream task. The experimental results on 3 GLUE datasets showed that our proposed method is on par with the state-of-the-art, as it achieves performance gains of around 2-3% on the downstream tasks compared to the baseline.

Then, we performed a case study on Abusive Language Detection models. We showed that improving the isotropy of the embeddings resulted in a change in the detected abusive sentences in the test set. Then, we discussed the ethical concerns of Abusive Language Detection systems and proposed recommendations for such systems.

7.2 Ethical Concerns

As discussed in the case study, many ethical concerns arise when using AI-based systems. We can list the following concerns: 1) privacy and data protection, 2) reliability and integrity, 3) accountability and explainability, and 4) moral responsibility. Since our contribution lies in the method, we chose not to discuss all of these ethical aspects. However, we remind the reader of the importance of moral responsibility, as the improvement in the performance of text classification systems can be positive or negative, based on the intended use case.

The Isotropy Layer approach is limited in terms of Accountability as it does not support explainability. Even though empirical evidence shows improvements in the proposed approach, we did not study the approach from the context of explainability. Since this property is not supported by default, there is a need to employ tools that provide interpretability to the approach.

7.3 Future work

This section highlights the most prominent directions to explore in future work. First, it would be interesting to investigate the discrepancies between the results of our method on RoBERTa compared to BERT (as discussed in Section 5.4). More specifically, we could try performing a better and more precise tuning of the hyperparameters used to train the Isotropy Layer of RoBERTa. Moreover, we could increase the complexity of the *Isotropy Layer* and expand it into an *Isotropy Network* (i.e., a multi-layer feed-forward neural network) to allow this network to handle the complexity of the RoBERTa embeddings. We also suggest to relax the strict constraint imposed by the downstream task layers (i.e., freezing the layers) by fine-tuning both the Isotropy Layer and the downstream task layers using the joint loss function.

On another hand, a promising direction would be to understand the impact of our solution on the semantics of the model. To do so, we propose to employ tools that allow us to navigate the embedding space, giving us insights on the distribution of the concepts in the embedding space (i.e., reach more interpretable results).

Finally, it would be interesting to explore whether the proposed method performs well on regression problems. This can be done by employing the same model proposed, with different downstream task layers and a different downstream task loss.

References

[1] Katikapalli Subramanyam Kalyan, Ajit Rajasekharan, and Sivanesan Sangeetha. Ammus : A survey of transformer-based pretrained models in natural language processing, 2021.

[2] Kawin Ethayarajh. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and GPT-2 embeddings. volume abs/1909.00512, 2019.

[3] Jun Gao, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tieyan Liu. Representation degeneration problem in training natural language generation models. In *International Conference on Learning Representations*, 2019.

[4] Lingxiao Wang, Jing Huang, Kevin Huang, Ziniu Hu, Guangtao Wang, and Quanquan Gu. Improving neural language generation with spectrum control. In *International Conference on Learning Representations*, 2020.

[5] S. Rajaee and Mohammad Taher Pilehvar. How does fine-tuning affect the geometry of embedding space: A case study on isotropy. In *EMNLP*, 2021.

[6] Wenxuan Zhou, Bill Yuchen Lin, and Xiang Ren. Isobn: Fine-tuning bert with isotropic batch normalization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(16):14621–14629, May 2021.

[7] Transforming Our World: The 2030 Agenda for Sustainable Development, author = United Nations General Assembly, year=2015.

[8] Kuncahyo Setyo Nugroho, Anantha Yullian Sukmadewa, and Novanto Yudistira. Large-scale news classification using bert language model: Spark nlp approach. In *6th International Conference on Sustainable Information Engineering and Technology 2021*, SIET '21, page 240–246, New York, NY, USA, 2021. Association for Computing Machinery.

[9] Vijay Srinivas Tida and Sonya Hsu. Universal spam detection using transfer learning of bert model, 2022.

[10] Tommaso Caselli, Valerio Basile, Jelena Mitrović, and Michael Granitzer. HateBERT: Retraining BERT for abusive language detection in English. In *Proceedings of the 5th Workshop on Online Abuse and Harms (WOAH 2021)*, pages 17–25, Online, August 2021. Association for Computational Linguistics.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[13] Xinying Song, Alex Salcianu, Yang Song, Dave Dopson, and Denny Zhou. Fast wordpiece tokenization, 2020.

[14] 14.8. bidirectional encoder representations from transformers (bert).

[15] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[16] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale, 2020.

[17] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics.

[18] Daniel Biś, Maksim Podkorytov, and Xiuwen Liu. Too much in common: Shifting of embeddings in transformer language models and its implications. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5117–5130, Online, June 2021. Association for Computational Linguistics.

[19] Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li. On the sentence embeddings from pre-trained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9119–9130, Online, November 2020. Association for Computational Linguistics.

[20] William Timkey and Marten van Schijndel. All bark and no bite: Rogue dimensions in transformer language models obscure representational quality. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4527–4546, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[21] William Rudman, Nate Gillman, Taylor Rayne, and Carsten Eickhoff. IsoScore: Measuring the uniformity of embedding space utilization. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3325–3339, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[22] Zhong Zhang, Chongming Gao, Cong Xu, Rui Miao, Qinli Yang, and Junming Shao. Revisiting representation degeneration problem in language modeling. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 518–527, Online, November 2020. Association for Computational Linguistics.

[23] Chengyue Gong, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tie-Yan Liu. Frage: Frequency-agnostic word representation. *ArXiv*, abs/1809.06858, 2018.

[24] Jiaqi Mu and Pramod Viswanath. All-but-the-top: Simple and effective post-processing for word representations. 2018. Publisher Copyright: © Learning Representations, ICLR 2018 - Conference Track Proceedings.All right reserved.; 6th International Conference on Learning Representations, ICLR 2018 ; Conference date: 30-04-2018 Through 03-05-2018.

[25] S. Rajaee and Mohammad Taher Pilehvar. A cluster-based approach for improving isotropy in contextual embedding space. In *ACL*, 2021.

[26] Yuxin Liang, Rui Cao, Jie Zheng, Jie Ren, and Ling Gao. Learning to remove: Towards isotropic pre-trained bert embedding. In *Artificial Neural Networks and Machine Learning – ICANN 2021: 30th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 14–17, 2021, Proceedings, Part V*, page 448–459, Berlin, Heidelberg, 2021. Springer-Verlag.

[27] Davood Zabihzadeh. Ensemble of loss functions to improve generalizability of deep metric learning methods. *ArXiv*, abs/2107.01130, 2021.

[28] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu,

Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing, 2019.

[29] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2017.

[30] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics.

[31] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

[32] Anna Schmidt and Michael Wiegand. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10, Valencia, Spain, April 2017. Association for Computational Linguistics.

[33] Dheeraj Pranav, Swetha Mahajan, Mahesh Reddy Kondapareddy, and B. Srinu. Classification for detecting insulting and abusive content (the international journal of analytical and experimental modal analysis), 05 2020.

[34] José Benítez-Andrades, Álvaro González-Jiménez, Álvaro López-Brea, Jose Aveleira, Jose Alija-Perez, and María García-Ordás. Detecting racism and xenophobia using deep learning models on twitter data: Cnn, lstm and bert. *PeerJ Computer Science*, 8:e906, 03 2022.

[35] Zeerak Waseem. Are you a racist or am I seeing things? annotator influence on hate speech detection on Twitter. In *Proceedings of the First Workshop on NLP and Computational Social Science*, pages 138–142, Austin, Texas, November 2016. Association for Computational Linguistics.

[36] Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. Automated hate speech detection and the problem of offensive language. 03 2017.

[37] Valeria Saladino, Stefano Eleuteri, Valeria Verrastro, and Filippo Petruccelli. Perception of cyberbullying in adolescence: A brief evaluation among italian students. *Frontiers in Psychology*, 11, 2020.

[38] Cristina Criddle. Facebook moderator: ’every day was a nightmare’, May 2021.

[39] Mithun Das, Abhisek Dash, Siddharth Jaiswal, Binny Mathew, Punyajoy Saha, and Animesh Mukerjee. Platform governance: Past, present and future. *GetMobile: Mobile Comp. and Comm.*, 26(1):14–20, may 2022.

[40] Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*, 2019.

[41] Bernd Carsten Stahl. *Ethical Issues of AI*, pages 35–53. Springer International Publishing, Cham, 2021.

[42] Tania Carvalho, Nuno Moniz, Pedro Faria, and Luís Antunes. Survey on privacy-preserving techniques for data publishing, 01 2022.

[43] Guodong Long, Tao Shen, Yue Tan, Leah Gerrard, Allison Clarke, and Jing Jiang. Federated learning for privacy-preserving open innovation future on digital health, 2021.

[44] Prasad Jadhav and Pramila Chawan. Data leak prevention system: A survey. 06:04, 09 2019.

[45] Yao Li, Minhao Cheng, Cho-Jui Hsieh, and Thomas C. M. Lee. A review of adversarial attack and defense for classification methods. *The American Statistician*, 0(0):1–17, 2022.

[46] Shannon Leigh Eggers and Char Sample. Vulnerabilities in artificial intelligence and machine learning applications and data. 12 2020.

[47] Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. Explainable ai: A review of machine learning interpretability methods. *Entropy*, 23(1):18, Dec 2020.