Compression of the embedding layer in an LSTM model using tensor train decomposition for NLP

S.A.Jonnalagadda

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Compression of the embedding layer in an LSTM model using tensor train decomposition for NLP

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

S.A.Jonnalagadda

November 17, 2022

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Abstract

Natural Language Processing (NLP) deals with understanding and processing human text by any computer software. There are several network architectures in the fields of deep learning and artificial intelligence that are used for NLP. Deep learning techniques like recurrent neural networks and feed-forward neural networks are used to develop language models that perform several NLP tasks. Over the years, researchers have worked on developing state-of-the-art language models that achieve high accuracy and performance for NLP applications. With the development of deep neural network language models, the computational resources requirements and the energy costs for training and running language models increased. This led to research to compress the language models, thereby reducing the computational complexity of the language models. One of the methods used for this is tensor decomposition, like the tensor-train (TT) decomposition. During this thesis work, the application of the TT-decomposition method for compressing the embedding layer in a long-short-term memory model was investigated. Specifically, the effect of factorization and the order of factors in the embedding layer when it is represented in the TT-matrix format on the maximum test accuracy of the long-short term memory model for the NLP task of sentiment analysis was investigated. This was done by considering three different factorizations of the embedding layer in the model. Further, the effect of change in TT-ranks (hyperparameters of the model when the embedding layer is represented in the TT-matrix format) on the maximum test accuracy was also investigated. Based on the investigation and empirical results obtained, this thesis concludes that by having a larger number of factors in the factorization of the embedding layer, the maximum test accuracy of the model increases. Further, in a particular factorization, when the factors were arranged in such a way that the maximum values of the TT-ranks had a smaller gap, the maximum test accuracy of the model improved. In one particular configuration of the model, the number of parameters was reduced by 24.5 times compared to that of the original uncompressed model, and a maximum test accuracy of 77.10% was achieved compared to a maximum test accuracy of 78.05% in the case of the original model.

Table of Contents

List of Figures

List of Tables

Acknowledgements

After a little more than 3 years, my journey at TU Delft is nearing its end. From the start of my master's in systems and control, I have faced various obstacles that have tested my academic and personal abilities. Lots of mistakes were made, and much more was learned. This report documents the work done in the last year for the thesis project, and this is the last stop of my master's journey at Delft, and I could not have made it here if I were by myself.

Firstly, I want to extend my heartfelt gratitude to my supervisor, Dr. Kim Batselier, for giving me the opportunity to work on this topic. His academic insights have always helped me restructure my approach to a certain problem, and he always made sure that his door was open whenever I wanted to discuss anything with him. My PhD supervisor, Ir. Frederiek Wesel, always provided constructive feedback whenever it was necessary and helped me improve my knowledge on the fundamentals of this work. I want to thank the both of them for being so patient and understanding throughout the entire duration of this thesis project. Both of them always treated me like a fellow colleague rather than a student, and that helped me improve my confidence and maneuver my way through roadblocks in the project.

Next, I want to thank my mom and dad for trusting me and always being there for me throughout my life. Words won't justify whatever the both of you did for me. I want to thank my sister for always understanding my struggles and being a positive force in my life. I want to thank all my extended family members, especially my grandparents and my cousin Sandeep, who have always stood by my side and encouraged me to perform better.

Apart from supervisors, my friends Akash and Vishnu never hesitated to share their knowledge on topics related to this thesis. The two of you helped me get back to the thesis project even when things were getting too heavy. I also want to thank my friend Tanay from Systems and Control, who has always helped with the course work at TU Delft.

I want to especially thank Evert Vixseboxse, my academic advisor from the department of 3mE at TU Delft, who has helped me during the difficult times that I have faced in Delft. I also want to thank Jai RambaratSingh for being a mentor in my professional and personal life.

I want to thank my brother Ajith for always having my back for the last 13 years of my life and my friend Thanisha for supporting me in every way she could. Finally, I want to thank all my other friends that I met in Rijswijk over the last three years and my friends from India, especially Sanya, Rajan, and Kumar who helped me reach this stage of my life.

Delft, University of Technology S.A.Jonnalagadda
November 17, 2022

This thesis work is dedicated to you, amma and nanna.

> "Hope is a good thing, maybe the best of things, and no good thing ever dies."
> — *The Shawshank Redemption*

> "All is One and One is All"
> — *Unknown*

Chapter 1

Introduction

NLP is broadly defined as the automatic manipulation of natural text or speech by any type of software. The development of NLP first started with research on linguistics. Although some of the earliest research on NLP dates back to the 1940s, it is a field that is still very much in development. This is because natural language is complex and processing and analyzing it requires a deep understanding and expertise of the language. For example, some of the challenges in a language include understanding emotions behind sentences. After research on linguistics and the development of machine translation (translation of text from one language to another), the research on NLP paved towards statistical NLP with a dominance of statistical approaches [21]. Statistical NLP methods employ mathematical techniques like probability theory to develop NLP models. The rapid development in computational power enabled the efficient implementation of statistical NLP methods, which became a mainstream tool for building and understanding natural language models.

We can trace back the origins of NLP to World War Two. One of the most significant revolutions in the field of automatic computing was when Alan Turing worked with the British government to build an automatic computing machine that could decipher the Enigma machine's settings. The Enigma machine was used by the German military to transmit messages during the war. Alan Turing developed the first ever intelligent computer that could crack and decipher the information that was being sent through Enigma machines [13].
His work on automatic computing machines led him to publish a study where he described what is now called the "Turing test" [42].
The work done by Turing, is considered to be one of the most important milestones in the field of artificial intelligence (AI). According to the Oxford dictionary, AI is defined as the theory and development of computer systems able to perform tasks normally requiring human intelligence, such as visual perception, speech recognition, decision-making, and translation between languages.
With the advancements in computational power and machine learning, scientists in the 1970s almost 20 years after Turing's research started focusing on NLP again which led to widespread recognition of NLP [21]. Data-driven approaches like machine learning gained more attention as complex NLP tasks could be performed. To perform NLP, language models are used.

Complex language models use neural networks consisting of millions of neurons and network layers for obtaining an accurate or desired output.

Most language models for NLP use word embeddings, which are continuous vector representations of a word. In a neural network, the input data consists of words or sentences, which in some cases are represented in a vector format of any size filled with numerical values [14]. Neural network-based language models using word vectors have a long history, wherein researchers used continuous vectors to represent words and trained neural networks to learn new word vectors [29]. With the introduction of neural networks in the field of AI and further development in the usage of word vectors for NLP, Mikolov et al. [29] in 2013 developed the skip-gram model and the continuous-bag-of-words model, where information in a text was represented using word vectors in a vector space. The skip-gram model and the continuous-bag-of-words model will be discussed in detail in the upcoming chapters.

Further studies in the field of NLP led Vaswani et al. [45] to discover the transformer architecture. The transformer architecture is a simple network architecture that is based on attention mechanisms. In 2018, deep contextualized word vectors were introduced where high-quality word vectors are learned and data is represented and processed using these word vectors [26]. Deep contextualized word vectors are word vectors learned by training existing word vectors on a bidirectional language model. These deep contextualized word vectors capture important aspects of a word (more context as to what the word represents) which make them rich or of high quality.
The current state-of-the-art model was recently introduced in 2022 with an architecture consisting of 20-billion parameters for NLP applications [11].

1-1 Why do we need NLP?

In today's world, there are several practical applications of NLP. Some of them are listed here to give a context as to why NLP is of essence.
When an e-mail box is opened, one of the first things that can be noticed is that the emails are automatically categorized as spam. This is known as spam detection [30]. Text classification is a classic application of NLP, where a language model takes in loads of data (text in the emails in this example) and tries to understand the kind of actions taken on different email. Machine translation is another real-world application of NLP. Machine translation is a task that involves translation of text from one language to another language using a language model. Another application of NLP in the real-world is grammatical and spelling error correction.
The above mentioned applications are only a few examples of how important and relevant NLP is in today's society. As mentioned above, these applications sometimes need complex neural network based language models.

1-2 What is NLP?

As described previously, NLP is an area of science that deals with processing human text. There are various definitions used to describe what NLP is but one of the definitions that capture its essence the most is given by Liddy [24]. According to her NLP is defined like this:

> *NLP is a theoretically motivated range of computational techniques for analyzing and representing naturally occurring texts at one or more levels of linguistic analysis to achieve human-like language processing for a range of tasks or applications.*

From the above definition, we can infer that NLP can be accomplished using multiple techniques and can be applied to any naturally occurring human text (such as English, Spanish, French, etc...). To achieve human-like processing, several neural network architectures used to build language models for NLP are based on machine learning. Hence, it can be said that NLP is part of the field of AI.

1-3 Motivation for thesis work

With the advancements in neural network architectures and attention mechanism-based language models like transformer, processing a large amount of data from various texts becomes computationally expensive. This is because, as more neural network layers are added, the number of neurons increases. As the number of neurons increases, the number of mathematical computations also increases. Finally, due to the increase in mathematical computations, memory requirements and storage requirements increase [14, 43].

Because of their computational complexity, neural network-based language models are not always suitable for running on low-memory devices such as mobile phones, drones, medical devices, and so on. These neural network-based language models often require powerful processing units that have substantial energy requirements for training and running. Hence, the need to reduce the storage and compress the size of these models for efficient computation became necessary [27].
Studies on the compression of neural network architectures led to the use methods like pruning [39], quantization [36], weight sharing [39] and use of different tensor decomposition methods in language models to reduce the number of parameters and model size [31]. Tensor decomposition methods can be a tool used to reduce the computational and storage cost of training a particular neural network architecture. Reducing storage and computational requirements helps in reducing the CO2 emissions incurred in the process of building and training state-of-the-art neural network architectures. To understand more about how tensor decomposition methods are useful for sustainable AI, please refer to [28].

There are various tensor decomposition methods which have been used to compress a neural network architecture. In this work, one of the tensor decomposition methods called the TT-decomposition method introduced by Oseledets [32] will be used to compress an LSTM model.

The idea for this thesis work is inspired by the work done by Hrinchuk et al. [19], wherein the TT-decomposition was successfully used to compress a bi-directional LSTM model for the application of NLP. In the bi-directional LSTM model, the embedding matrix (the matrix containing word vectors) is compressed using the TT-decomposition. Specifically during the experimental work conducted by Hrinchuk et al. [19], the embedding matrix of the model is factorized into different configurations, however the effects TT-ranks, a hyper-parameter of TT-decomposition on the performance of the language model are not explicitly studied. Hence, in this study the effects of the TT-ranks on the performance of an LSTM model will be investigated. Further, the effect of factorization and the order in which these factors are placed will be analysed in detail. The following research questions will be answered through this work:

- *In an LSTM model used to perform NLP, when standard embeddings are replaced by TT-embeddings, how does the factorization of the embedding matrix and subsequently the TT-ranks effect the test accuracy and the compression rate?*
- *In the same LSTM model used to perform NLP, when TT-embeddings are used, how does the order of the factors in each factorization effect the compression rate and the test accuracy of the model?*

Chapter 2

Deep learning and word representation models

Before looking into state-of-the art techniques in NLP, it is important to understand the basic concepts related to the field of NLP. This chapter will talk about features of the human text, language models (LMs), some neural network architectures used in LMs to perform NLP and how deep learning plays a role in the field of NLP.
Note: In the following sections of the report unless specified, a scalar is represented by normal letters (a, A), a vector is represented by small bold letters ($\mathbf{x}$), a matrix is represented by capital bold letters ($\mathbf{X}$) and a tensor is represented by capital roman-style letters ($\mathcal{X}$).

2-1 Different features of the human text

For a computer software to process human text, knowledge about certain linguistic characteristics is essential. A LM is built in a manner in which it can incorporate these characteristic to perform complex NLP tasks. The following can be categorized as different aspects or levels of NLP [24]:

- **Morphology:** Humans break down words into smaller units, referred to as morphemes, to understand the meaning of a particular word. For example, the word "unhealthy" can be split into two parts. The first part "un" is the prefix, and the root word is "healthy". Another case is the word "danced" where the suffix "ed" means that the action "dance" has already been completed in the past. Similar to humans, machine learning algorithms used in LMs for NLP can also look for morphemes in the word and recognize the meaning that is being conveyed. These features in a word are also called morphological features.
- **Syntactic:** This level is where a LM deals with the grammatical structure of a sentence and the relationship between two words in a sentence. For example, "Aravind rescued

the dog" and "The dog rescued Aravind" are two sentences that differ in syntax. The placement of the words "Aravind" and "dog" in the sentence can change the meaning of the sentence.

- **Semantics:** At this level, the LM attempts to process and comprehend the meaning of a sentence by analyzing the interactions between the words in the sentence.
- **Pragmatic:** Sometimes, the usage of the same word in two different sentences can entirely change the context in which the word is used. To read into the meaning of the sentences, LMs need a lot of knowledge and data to be trained on. For example, in the following two sentences, the context of the word "they" is different.

  "The board of examiners refused the students a permit for an online examination because they feared fraud and cheating."
  "The board of examiners refused the students a permit for an online examination because they encouraged fraud."
  In the above two sentences, the word "they" refers to two different subjects. In the first sentence, "they" refers to the board of examiners, while in the second sentence, "they" refers to the online examinations.

In order to understand the human language in-depth, NLP systems require LMs that are modelled based on the above mentioned different aspects of NLP.

2-2 Deep learning: An effective tool for NLP

With the advancements in statistics and computer technology, deep learning algorithms have become a popular tool used to build LMs for NLP [14]. The availability of large amounts of data was another important factor that was exploited by researchers to their advantage to train deep learning architectures for NLP.
Neural network architectures based on recurrent neural networks and feed-forward neural networks started to gain attention as complex NLP tasks could be completed using LMs built upon these neural network architectures. Deep learning can be described as a branch of machine learning that processes complex data using neural networks with more number of neurons and network layers [43]. Deep learning is a subset of machine learning, with some machine learning methods using neural network architectures that are not deep in the sense that they do not have many neurons or network layers while deep learning has more neurons and network layers that are used for complex NLP tasks.

2-2-1 Neural networks: Foundation of deep learning

Deep learning algorithms process input data using neural networks (NNs).
A NN is similar to the human brain. The human brain collects information, passes that information through neurons, and processes the information. Similarly, a NN collects data through an input layer and processes this information by passing it through several neurons

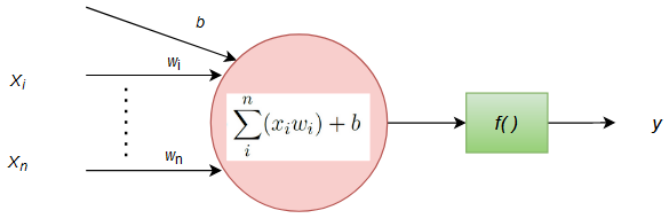


Figure 2-1: Perceptron architecture. The inputs x_i to x_n are taken by the perceptron and mapped to the output y. Weights w_i to w_n are assigned to each input value. The bias b is applied to the weighted sum. $f()$ is the activation function.

which are divided between different layers in the NN. One can think of neurons as computational cells of the NN with each cell linked to others.
The basic unit of a NN, the neuron is based on the architecture of the perceptron [14]. The perceptron accepts input $\mathbf{x} \in \mathbb{R}^n$ consisting of features x_1 to x_n. Each feature x_i, is given a learned weight $w_i \in \mathbb{R}$ to map the input to an output $y_i \in \mathbb{R}$. An activation function f then, takes the weighted sum as an input and a bias b is added to it to produce the output y. For a binary classification problem, the equations of the perceptron can be written as [43]:

$$y(x_1, ..., x_n) = f(w_1 x_1 + ... + w_n x_n). \tag{2-1}$$

In the vector format, the above equation can be re-written as:

$$y(\mathbf{x}) = f(\mathbf{w}^T \mathbf{x} + b). \tag{2-2}$$

where, $\mathbf{x} = [x_i \ x_{i+1} \ ... \ x_n]$, $i = 1$ to n and $f(\cdot) : \mathbb{R}^n \rightarrow \mathbb{R}$ is the activation function.

A multilayered perceptron links multiple perceptrons together into a network. Because of this the mutilayered perceptron by extension can also be referred to as a NN. A NN has many layers, with each layer consisting of neurons. There is an input layer and an output layer. All the layers in between these two layers are called hidden layers [43]. Each layer of the NN consists of several nodes or neurons that receive and transform data from each layer to the next layer [34]. Figure 2-2 shows an example of a NN consisting of an input layer, one hidden layer and an output layer. During the training of the NN, the data is first propagated forward to predict an output $\hat{y}$. The equations for the NN given in figure 2-2 are given as follows:

$$\begin{aligned} h_1 &= f(w_{1,1} x_1 + b_{1,1} + w_{2,1} x_2 + b_{2,1}) \\ h_2 &= f(w_{1,2} x_1 + b_{1,2} + w_{2,2} x_2 + b_{2,2}) \\ \hat{y} &= f(h_1 w_1 + h_2 w_2 + b). \end{aligned}$$

In the above set of equations, both the hidden nodes h_1, h_2 and the output node are activated by an activation function $f() : \mathbb{R}^n \rightarrow \mathbb{R}$. The inputs x_1 and x_2 are features of vector $\mathbf{x} \in \mathbb{R}^n$. The weights $w_{1,1}$, $w_{1,2} \in \mathbb{R}$ are weights for the channels connecting the first node of the input layer to the nodes of the hidden layer. The weights $w_{2,1}$, $w_{2,2} \in \mathbb{R}$ are weights for the channels connecting the second node of the input layer to the nodes of the hidden layer.

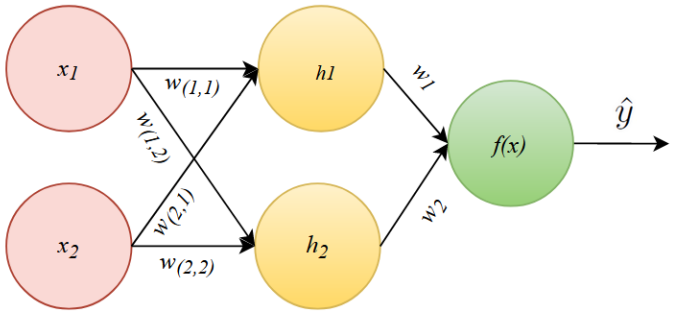


Figure 2-2: Example of a NN having an input layer, one hidden layer and an output layer. The activation functions for all the nodes is given by $f(x)$. The weights $w_{1,1}$, $w_{1,2}$ are weights for the channels connecting the first node of the input layer to the nodes of the hidden layer. The weights $w_{2,1}$, $w_{2,2}$ are weights for the channels connecting the second node of the input layer to the nodes of the hidden layer.

Activation functions are non-linear functions that are used to introduce non-linearity in the NN. Activation functions are generally non-linear, as choosing linear functions would restrict the network to only being able to learn linear transforms of the input data. This leads to the network producing only linear outputs that are often not accurate. These activation functions decide whether a neuron can contribute to the next layer in the network. Popular activation functions include the sigmoid function, the hyperbolic tangent (tanh) function and the Rectified Linear Unit (ReLU) function [43].

After the data is propagated forward, the next step during the training of the NN, is the error computation step where the error between the predicted output value and the true output value is computed. After this step, the error is propagated backwards to adjust the weights and biases of the NN in order to make the model more accurate. More details about these steps can be found in [14, 43].

2-3 Neural network architectures

The concepts explained above are essential to understand how NNs work. Types of NN architectures, particularly recurrent neural networks and long Short term memory networks are discussed in the subsequent sections below.

2-3-1 Recurrent neural networks

Recurrent neural networks (RNNs) are NNs designed to handle sequential data. RNNs have feedback loops in the hidden layers, which differentiates them from regular feed-forward NNs. In order to incorporate sequential context into the next time step's prediction, the RNN preserves a memory of the previous time step. The RNNs ability to integrate information over many time steps makes it useful to obtain accurate predictions and enhance the learning of word vectors in NLP. The architecture of a RNN is shown in figure 2-3. The RNN is shown in

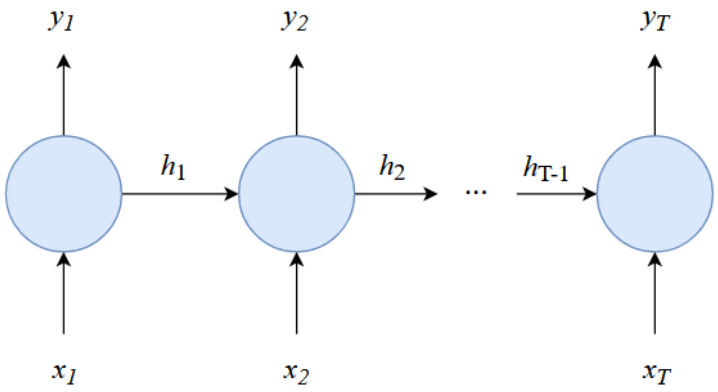


Figure 2-3: Architecture of a RNN containing an input sequence $\mathbf{x}_1$, $\mathbf{x}_2$...$\mathbf{x}_t$, hidden states h_1,h_2,...,h_t and output y_t.

figure 2-3 is described as follows: Given a sequence of inputs $\mathbf{x}_1$ to $\mathbf{x}_t$ such that $\mathbf{x}_T \in \mathbb{R}^n$ where n is the number of features of each vector and $t = 1$ to T, the RNN computes a sequence of hidden states (h_1 to h_t) and a sequence of outputs for each input (y_1 to y_t).The output y_t can be defined as:

$$y_t = f(\mathbf{x}_t, h_{t-1}). \tag{2-3}$$

where, in the above equation, the function $f(\dot{}) : \mathbb{R}^n \rightarrow \mathbb{R}$ maps the memory from the previous time step h_{t-1} and input at current time step $\mathbf{x}_t$ to the output y_t.
The output vector y_t can be considered as the history vector for the entire sequence up to and including time t. This means:

$$h_t = y_t = f(\mathbf{x}_t, h_{t-1}). \tag{2-4}$$

By noticing the above equation, it can be seen why we call these RNNs. It is because the output is directly dependent on the previous result and for each instance the same function is applied [43]. RNNs are trained in a similar way to that of feed-forward NNs. Data is propagated forward through the network, the error function is computed, the gradients are computed for each set of weights/parameters during back propagation and the parameters are updated to minimize the error using an optimizer.
Like the feed-forward NNs, it is difficult to train RNNs with the gradient descent algorithm because of the vanishing gradient problem. The gradient decays exponentially as it is back propagated through time [18].
Instead, an alternative architecture called the long short term memory (LSTM) network, started to gain popularity in NLP applications.
For an LSTM network, the RNN model is modified to include memory units that are designed to store information for long periods of time [20].

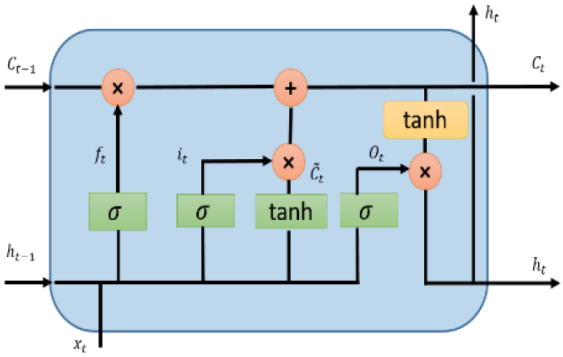


Figure 2-4: Design of LSTM cell where, i_t is the input gate, f_t is the forget gate, o_t is the output gate. $\mathbf{c}_t$ is the new cell state and $\mathbf{c}_{t-1}$ is the previous cell state. $\mathbf{h}_t$ and $\mathbf{h}_{t-1}$ represent new and previous hidden states respectively. $\mathbf{x}_t$ is the input data. tanh and σ are activation functions. $\times$ represents point-wise multiplication, $+$ indicates point wise addition. This figure is taken from [27].

2-3-2 LSTM networks

In a LSTM network, the RNN node in the hidden layer is replaced with a LSTM cell, which is designed to store long-term information. The LSTM cell uses three gates to control the usage and update the text history information. These gates are the input gate (i_t), the forget gate (f_t) and the output gate (o_t). The memory cell and the gates enable the LSTM to store long-distance history information unlike a traditional RNN which cannot store the past information due to the vanishing gradient problem. The structure of a LSTM cell is given in figure 2-4. In the LSTM cell, the gates are used to add or remove information from cell state $\mathbf{c}_t$. The calculation process of LSTM mainly includes the following steps [8]:

- Firstly, the values of the forget gate f_t and input gate i_t are computed. The equation for f_t is given as [27]:

$$f_t = \sigma(w_f[\mathbf{h}_{t-1}, \mathbf{x}_t] + b_f). \tag{2-5}$$

  In the above equation w_f $[\mathbf{h}_{t-1}, \mathbf{x}_t] = w_f\mathbf{h}_{t-1} + w_f\mathbf{x}_t$.
  The forget gate f_t accepts input data at current time step t, $\mathbf{x}_t$ and the recurrent information from the previous hidden state $\mathbf{h}_{t-1} \in (-1, 1)^h$ where h are number of LSTM cells. The forget gate has weight w_f, bias b_f and an activation function $\sigma(\cdot) : \mathbb{R}^n \to (0, 1)$. If any data or information is irrelevant for the new cell state $\mathbf{c}_t \in \mathbb{R}^n$, the forget gate's value is close to 0, and it is close to 1 if the data is relevant for the new cell state. After determining the value of the forget gate, it is point-wise multiplied by the previous cell state $\mathbf{c}_{t-1}$ to determine whether the previous cell states are relevant or irrelevant. Given the previous hidden state and the new input data $\mathbf{x}_t$ in the sequence, the forget gate decides which pieces of long-term memory should have less weight (or be forgotten).

- Secondly, the input gate i_t takes in the same inputs as the forget gate. The equation for the input gate is given by [27]:

$$i_t = \sigma(w_i[\mathbf{h}_{t-1}, \mathbf{x}_t] + b_i). \tag{2-6}$$

The input gate has weight w_i, bias b_i and activation function $\sigma(\cdot) : \mathbb{R}^n \to (0, 1)$. The input gate is sigmoid activated, which acts a filter identifying the components of the new memory vector $\hat{\mathbf{c}}_\mathbf{t}$ that are worth retaining.

- The new memory vector $\hat{\mathbf{c}}_\mathbf{t}$ is obtained by taking into account the previous hidden state and the input data $\mathbf{x}_t$.
  The equation for the new memory vector is given by [27]:

$$\hat{\mathbf{c}}_\mathbf{t} = \tanh(w_c[\mathbf{h}_{t-1}, \mathbf{x}_t] + b_c). \tag{2-7}$$

  The vector is $\tanh(\cdot) : \mathbb{R}^n \to (-1, 1)$ activated with a bias of b_c and weight w_c. Given the new data $\mathbf{x}_t$ and the previous hidden state $\mathbf{h}_{t-1}$, this vector determines how much to update each component of the new cell state $\mathbf{c}_t$.

- After calculating the values of the new memory vector and the input gate, the values are point-wise multiplied.
  This is given by:

$$\mathbf{c}_t = f_t.\mathbf{c}_{t-1} + i_t.\hat{\mathbf{c}}_\mathbf{t}. \tag{2-8}$$

  So, the input gate decides if the data in the new memory vector is worth adding to the new cell state $\mathbf{c}_t$. The value is added to the new cell state $\mathbf{c}_t$ after point-wise multiplication.

- The update of the new cell state is complete after calculating the values for the forget gate f_t, the input gate i_t, and the new memory vector $\hat{\mathbf{c}}_\mathbf{t}$.

- Finally, the output gate o_t decides the value of the new hidden state $\mathbf{h}_t$. The equations for the output state and the new hidden state are given by [27]:

$$o_t = \sigma(w_o[\mathbf{h}_{t-1} + b_o]), \tag{2-9}$$

$$\mathbf{h}_t = o_t.\tanh(\mathbf{c}_t). \tag{2-10}$$

  In the above equations, w_o and b_o represent the weight and the bias of the output gate. The "." (dot operator) denotes element-wise multiplication.
  The value of the newly updated cell state $\mathbf{c}_t$, the previous hidden state $\mathbf{h}_{t-1}$, and the input data $\mathbf{h}_t$ are all fed into the output gate. The new cell state is tanh activated, and the values are obtained in the range (-1,1). Then, $\mathbf{h}_{t-1}$ and $\mathbf{x}_t$ are passed through sigmoid activation function to filter out values that are not necessary to be stored in the output (that is the new hidden state $\mathbf{h}_t$). The newly updated cell state (after passing through tanh function) and the output of the sigmoid activation function are point wise multiplied to obtain the new hidden state $\mathbf{h}_t$.

2-4 Word representation models

For NLP, first a LM architecture is required that can process the information and store the data in a sentence or set of sentences. Secondly, the LM should be capable of understanding the information present in the language text. This can be done by pre-training the LM using large data sets of text or by developing an algorithm for the LM.
As mentioned earlier, words cannot be directly taken as input data for any NN. Words are often represented as continuous vectors which have elements that can take real valued-numbers. Work on using vectors to represent words with similar semantics or syntactic properties predates the 1990s [43]. One of the key ideas to make word representations using vectors more efficient was done by Hinton et al. [15] which also introduced the idea of learning representations through back-propagation for NNs in 1986 [7]. One of the most popular models which led to further research on word vectors was introduced by Bengio et al. [4] in which a feed-forward NN was used to jointly learn word vector representation and a statistical LM.
It is important to note that the foundational ideas of these LMs come from the field of distributional semantics. Distributional semantics is a subfield of NLP predicated on the idea that word meanings are derived from their usage. The distributional hypothesis states that words used in similar contexts have similar meanings. That is, if two words often occur in the same set of words, then they are semantically similar in meaning (occur in similar contexts) [15].

During the first decade of the twenty-first century, researchers concentrated on how to use word vectors to simplify and improve NLP applications. People began researching how sentences for NLP could be better represented. NN-based NLP started to gain popularity because of its ability to process complex data and be trained on large datasets. Based on the work of Bengio et al. [4], Mikolov et al. [29] proposed two novel architectures for computing continuous vector representations of words from large data sets. The idea was to focus on distributed representations of words as they are shown to perform much better than previous models like Latent Sematic Analysis (LSA) [10], Latent Dirichlet Allocation (LDA) [9] and n-gram models [29].
Before explaining the concepts of the CBOW and Skip-gram model, the n-gram model and the feed-forward neural network based language model (NNLM) will be explained briefly.

2-4-1 The n-gram model

n-grams are sequences of characters or words extracted from a text. n-grams can be divided into character based n-grams and word based n-grams. In a character based n-gram model, the n-gram set consists of n consecutive characters from a word in the text. The typical values of n are 2, 3 or 4. When the n value is 2, it is called a bigram. When the value of n is 3, it is referred to as a trigram. For example, the word university results in the generation of the bigrams:
U, UN, NI, IV, VE, ER, RS, SI, TY, Y where '*' denotes a padding space. For a word containing n characters, there are $n + 1$ bigrams and there are $n + 2$ trigrams.

In language modeling, the n-gram model is used to predict the occurrence of a current word based on the history of occurrences of the previous words in the text. In a given set of

training data (corpus), the n-gram model constructs tables of conditional probabilities for the next word, for each one of a large number of contexts (i.e, the combinations of the last $n-1$ words). So, in the n-gram model, only the occurrence of the previous $n-1$ words is taken into context. This is known as the nth order Markov property [43]. The conditional probability can be calculated for an n-gram model as:

$$p(w_1 w_2 \cdots w_L) = \prod_{i=n}^{L} p(w_i | w_{i-1} \cdots w_{i-n-1}). \tag{2-11}$$

In the above equation, there are L words and n represents context, w_i is the current word [43]. The values of n can be chosen as 1, 2, 3... and so on. For n =1, we get a unigram model. For n=2, we get a bi-gram model. The equation for $n = 2$ is given as, yielding L-1 bigrams:

$$p(w_1 w_2 \cdots w_L) = \prod_{i=2}^{L} p(w_i | w_{i-1}). \tag{2-12}$$

The n-gram model is used for speech recognition, information retrieval, spelling correction, statistical stemmers and other NLP tasks [14].

2-4-2 Probabilistic feed-forward NNLM

Bengio et al. [4] proposed a probabilistic feed-forward NNLM which consists of input, projection, hidden and output layers. The main aim of this NNLM is to learn the joint probability function of sequences of words in a language by reducing the effects due the curse of dimensionality : if one wants to model the joint distribution of 10 consecutive words in a natural language with a vocabulary V of size 100,000, using 1-of-V coding where each word is assigned the number 1 in a vector with all the other entries as 0, there are potentially

$$100,000^{10} - 1 = 10^{50} - 1 \tag{2-13}$$

free parameters to model.
In this method, the effect of the curse of dimensionality is reduced by learning a distributed representation of words, which allows each training sentence to inform the model about an exponential number of semantically neighboring sentences [4].
The word representation vector or the feature vector represents different aspects of the word. The following approach is followed in this method [4]:

- Associate a distributed word feature vector (a real valued vector in $\mathbb{R}^m$) to each word in the vocabulary. Here m is the number of features for each word and is much smaller than the number of words in the vocabulary V.
- Express the joint probability function of word sequences in terms of the word vectors of these words in the sequence.
- Learn simultaneously the word feature vectors and the parameters of the probability function.

Please refer to [4] for an in-depth understanding of the feed-forward NNLM.

2-4-3 Continuous bag-of-words model and skip-gram model

The probabilistic feed-forward-NNLM paved way for two novel architectures which were proposed by Mikolov et al. [29]. They are the continuous bag-of-words (CBOW) model and the skip-gram model. These two LMs are trained in the following way:

- Continuous word vectors are learned using a simple model such as the statistical trigram model [1] and
- The n-gram NNLM is trained on top of these distributed representations of words.

The CBOW model

The CBOW model is based on a projection layer, that predicts a target word given a context window of C words to the left and the right side of the target word. The input layer in the model maps each context word through an embedding matrix $\mathbf{W} \in \mathbb{R}^{V \times N}$ to a vector representation of dimension N. Here, N is the size of the vector space onto which each word vector is projected and V is the size of the vocabulary consisting of different words.
The resulting vectors of the context words are averaged across each dimension to yield a single vector of dimension N. The embedding matrix $\mathbf{W}$ is shared by all context words in the model. The CBOW model has an input layer, where the input data containing words is given as input to the model. The input data in the model is considered to be of size V. Here each input consists of vector $\mathbf{x}_i$ that are one hot coded or 1-of-V coded vectors. As a result, only one unit in the vector $\mathbf{x}_i$ will have the value 1, and the vector's other values will be 0. For example, take a vocabulary V containing just 4 unique words, then the words are represented in the vector format like this (vec just represents each word in the 1-of-V vector format):
• vec("cat") = [1 0 0 0]
• vec("pizza") = [0 1 0 0]
• vec("car") = [0 0 1 0]
• vec("plant") = [0 0 0 1]
The architecture of the CBOW model is given in figure 2-5. The CBOW model has a word window of size C, which defines how many words are taken as context for the target word. Thus the input layer contains words represented as vectors $\mathbf{x}_1$, $\mathbf{x}_2 ... \mathbf{x}_C$.
The hidden layer in the model is an N-dimensional vector denoted by $\mathbf{h}$. The output layer produces the output word y. The input vectors are connected to the hidden layer via a $V \times N$ weight matrix denoted by $\mathbf{W}$ and the hidden layer is connected to the output layer via a $N \times V$ weight matrix $\hat{\mathbf{W}}$. Here $\mathbf{W}$ and $\hat{\mathbf{W}}$ are not the transpose of each other!
Each row of the matrix $\mathbf{W}$ is the N-dimension vector representation $\mathbf{v}_w$ of the associated word of the input layer. For the input $\mathbf{x}$ having k units where $k = 1$ to V, with $x_k = 1$ and $x_{k'} = 0$ for all $k' \neq k$ (one-hot coding), the output for the hidden layer can be computed as:

$$\mathbf{h} = \frac{1}{C}\mathbf{W}.(\sum_{i=1}^{C}\mathbf{x}_i). \tag{2-14}$$

The above equation is the average of the input vectors weighted by the matrix $\mathbf{W}$. The inputs to each node of the output layer are computed as:

$$u_j = {\mathbf{v}'_{wj}}^T.\mathbf{h}. \tag{2-15}$$

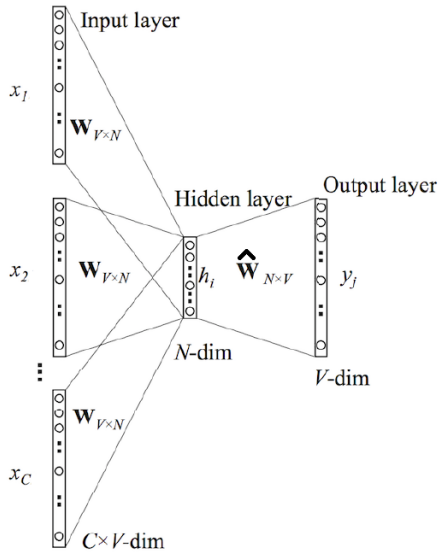


Figure 2-5: Architecture of the CBOW model. The input layer contains inputs $\mathbf{x}_1$ to $\mathbf{x}_C$, where C is the window size. The vocabulary size is V consisting of the input vectors where all the vectors are one-hot encoded. The hidden layer is a N dimensional vector $\mathbf{h}$. The output layer produces output word y_j. The input layer and the hidden layer are connected through weight matrix $\mathbf{W}$, the hidden layer and the output layer is connected by weight matrix $\hat{\mathbf{W}}$. This image was taken from [37].

In the above equation, $\mathbf{v'}_{wj}$ is the jth column of the output matrix $\hat{\mathbf{W}}$. When a dot product is applied to vector $\mathbf{v'}_{wj}$ and hidden layer vector $\mathbf{h}$, a scalar value u_j is obtained. Finally, the output of the output layer is given by passing the input u_j through a soft-max function:

$$y_j = p(w_{yj}|w_1, ..., w_C) = \frac{\exp(u_j)}{\sum_{j'=1}^{V} \exp(u'_j)}. \tag{2-16}$$

The soft-max function is used to convert a vector of V float numbers to a probability distribution by first scaling the numbers so that all values are between 0 and 1, and then normalizing these values so that all values sum to 1 [14]. After obtaining each output y_j for the associated word, similar to NNs, the elements of the weight matrices $\mathbf{W}$ and $\hat{\mathbf{W}}$ are adjusted through back-propagation to minimize the error in the model. Through the training of the CBOW model, the predictions are made closer and closer to the actual context words, and in this way, the embedding matrix $\mathbf{W}$ contains the learned vectors of the input words. Please refer to [37] and [43] for the worked-out derivations of error computation, back propagation and the update of parameters of the CBOW model.

Skip-gram model

The skip-gram model or the continuous skip-gram model is similar to the previous model with a slight difference. The CBOW model predicts the current word based on the inputs

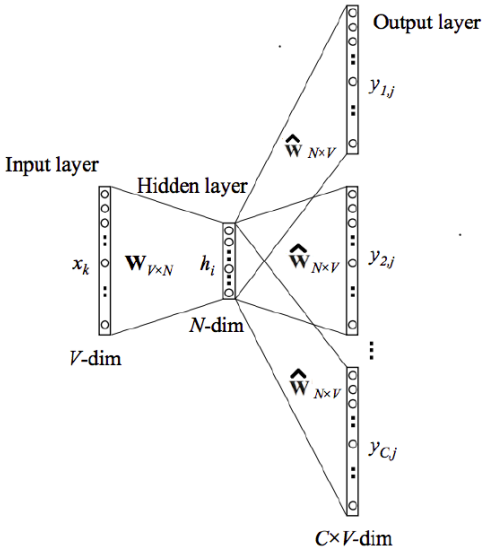


Figure 2-6: Architecture of the skip-gram model. The input layer contains inputs $\mathbf{x}$, C is the context size. The vocabulary size is V consisting of the input vectors where all the vectors are one-hot encoded. The hidden layer is a N dimensional vector $\mathbf{h}$. The output layer produces output words $y_{i,j}$ where $i = 1$ to C. The input layer and the hidden layer are connected through weight matrix $\mathbf{W}$, the hidden layer and the output layer are connected by weight matrix $\hat{\mathbf{W}}$. This image was taken from [37].

given to the model, while the skip-gram model tries to classify a word based on another word in the same sentence. The architecture for the skip-gram model is shown in figure 2-6. In the skip-gram model, an input sequence consists of a word and the model aims to predict C words to the left and the right closest to the given word. The model's input is a single word, denoted by the vector $\mathbf{v_{wI}}$, and the hidden layer $\mathbf{h}$ is an N-dimensional vector. In this case, the vector $\mathbf{h}$ is simply copied and transposed to the hidden weight matrix, $\mathbf{W}$ of size $V \times N$, associated with the input word w_I. The output of the hidden layer $\mathbf{h}$ is formulated as:

$$\mathbf{h} = \mathbf{h}^T\mathbf{W} = \mathbf{W}_{(k,.)} = \mathbf{v}_{wI}. \tag{2-17}$$

In the output layer, instead of computing one probability distribution, C probability distributions are computed. Each output is computed using the hidden to output weight matrix $\hat{\mathbf{W}}$ of size $N \times V$. The output for each context word is given by:

$$p(w_{c,j} = w_{O,c}|w_I) = y_{c,j} = \frac{\exp(u_{c,j})}{\sum_{j'=1}^{V} \exp(u_{j'})}. \tag{2-18}$$

In the above equation, $w_{c,j}$ is the jth word on the cth panel of the output layer, $w_{O,c}$ is the actual cth word in the output context words, w_I is the input word, $y_{c,j}$ is the output of the jth unit on the cth panel of the output layer and $u_{c,j}$ is the net input of the jth unit on the cth panel of the output layer. Because the output layers share the same weights, the net input $u_{c,j}$ can be written as:

$$u_{c,j} = u_j = \mathbf{v'}^T_{w,j}.\mathbf{h}. \tag{2-19}$$

Model	Complexity per one epoch
CBOW model	$O(t.n.N + N.V)$
Skip-gram model	$O(t.C.(N + N.V))$

Table 2-1: Computational complexity for the skip-gram model and the CBOW model per epoch. t is the number of words in the training set (typically upto 1 billion), n number of words in the input sequence, N size of the dimensional vector, V size of the vocabulary. The "." operator is the scalar multiplication operator. C is the size of the context window [29].

In the above equation, $\mathbf{v'}_{w,j}$ is the output vector of the jth word in the vocabulary w_j.
The skip-gram model is also trained after computing the error and back-propagating it through the layers, updating the weight matrices $\mathbf{W}$ and $\hat{\mathbf{W}}$ to minimize the error. In this case, also, the final weight matrix $\mathbf{W}$ (also known as the embedding matrix) after training contains the learned word embeddings for all the unique words in the vocabulary V. Please refer to [37] and [43] for the worked-out derivations of error computation, backpropagation and the update of parameters of the skip-gram model.

The computational complexity of the skip-gram models and the CBOW models are given in table 2-1. After the models are trained to a sufficient extent, semantic relationships between different words can be determined by the models. For example, word pairings like "Paris is to France" are similar to "Berlin is to Germany" can be determined by the LM. In order to to achieve this, the LM has to be trained on large data sets with high dimensional word vectors [29].

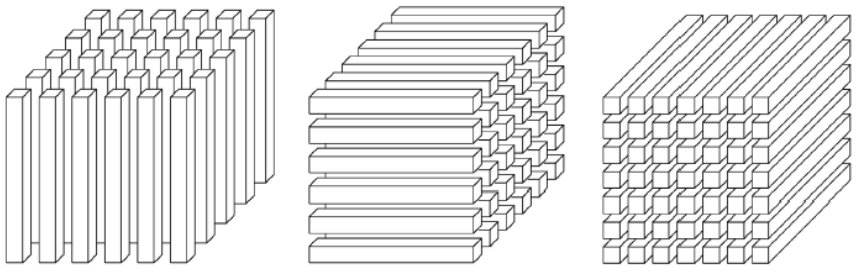
After the introduction of the skip-gram model and word vectors, further studies led to the development of deep contextualized word vectors [26] and the transformer architecture [45]. With the increasing storage and computational requirements for state-of-the-art NN architectures, researchers started working on methods to compress the LMs and to make them more efficient for NLP. The introduction of tensor decomposition methods into the world of NLP led to several state-of-the-art LMs in the field of NLP. The use of the TT-decomposition method to compress the embedding layers of an LSTM model for NLP applications will be discussed in detail in the next chapter. Before looking into the applications of tensors in NLP, it is essential to understand what tensor decomposition methods are and how they work. The next chapter will also explain the basic concepts of tensors and their properties which make them an interesting tool useful for NLP applications.

Chapter 3

Tensor decompositions

Tensors are higher-order arrays with N modes. A third order tensor or a tensor with three modes is denoted by $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$. A vector is a first order tensor with one mode. An array is a second-order tensor or a two-way tensor with two modes. A vector $\mathbf{x}$ can have n number of entries. An array or a matrix $\mathbf{X}$ has (I,J) number of elements. Arrays of order three or higher are referred to as tensors.
Sub-arrays are indicated in a matrix as rows and columns. Similarly, in tensors, these are known as fibers. Fibers are the higher-order analog of matrix rows and columns. Fibers in 3-way tensors are known as column, row and tube fibers. These are denoted as $\mathbf{x}_{:jk}$, $\mathbf{x}_{i:k}$ and $\mathbf{x}_{ij:}$ respectively. The figure 3-1 shows a visualization of fibers. In tensors column fibers are called mode-1 fibers, row fibers are called mode-2 fibers and tube fibers are called mode-3 fibers. Slices are two-dimensional sections of a tensor. They are defined by fixing all but two indices of the tensor. For a third order tensor $\mathcal{X}$ the horizontal, lateral and frontal slices are denoted by $\mathbf{X}_{i::}$ $\mathbf{X}_{:j:}$ and $\mathbf{X}_{::k}$ respectively. The figure 3-2 shows a visualization of slices in a third order tensor [41]. The frobenius norm of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots I_N}$ is given by:

$$||\mathcal{X}||_F = \sqrt{\sum_{i_1=1}^{I_1} \sum_{i_2=2}^{I_2} \ldots \sum_{i_N=1}^{I_N} x^2_{i_1 i_2 \ldots i_N}}. \tag{3-1}$$



(a) Mode-1 (column) fibers: $\mathbf{x}_{:jk}$ (b) Mode-2 (row) fibers: $\mathbf{x}_{i:k}$ (c) Mode-3 (tube) fibers: $\mathbf{x}_{ij:}$

Figure 3-1: Different fibers of three dimension tensor [41].

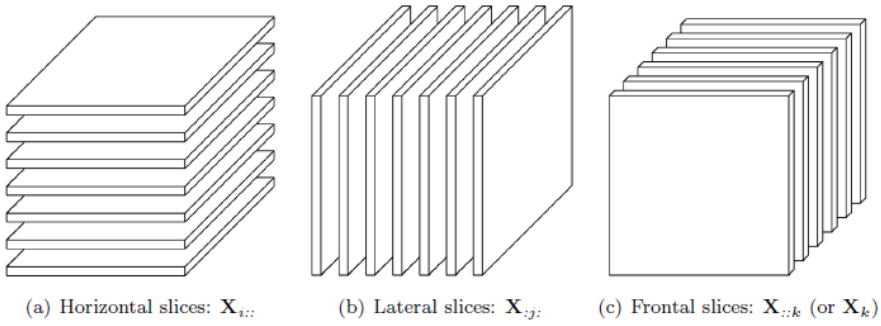


Figure 3-2: Different slices of three dimension tensor [41].

3-1 Tensor decompositions

The number of entries in a tensor grows exponentially in N for a N-way tensor. Because of this, high dimensional problems cannot be handled efficiently as numerical operations and memory requirements grow exponentially as well [32]. Hence, tensor decompositions were introduced to efficiently represent a tensor. In the late 1920s, Hitchcock [16] studied and researched the idea of factorizing a tensor into a sum of a finite number of rank-one tensors. After further research by a few others, this idea came to be known as the canonical decomposition method or CP decomposition. In 2011, the tensor-train decomposition or TT-decomposition was introduced by Oseledets [32]. The TT-decomposition will be discussed in detail in the following sections.

3-1-1 Tensor train decomposition

When a tensor is represented in tensor train format, a d- dimensional $\mathcal{B}$ is approximated by a tensor $\mathcal{A}$ such that $||\mathcal{A} - \mathcal{B}||_F \leq \epsilon ||\mathcal{B}||_F$. $||\mathcal{B}||_F$ is the norm of a tensor that is given by equation 3-1. ϵ is the prescribed accuracy for the approximation. The tensor $\mathcal{A}$ has the following elements:

$$\mathcal{A}(i_1, i_2, ..., i_d) = \mathbf{G}_1(i_1)\mathbf{G}_2(i_2)...\mathbf{G}_d(i_d). \tag{3-2}$$

where, $\mathbf{G}_k(i_k)$ is a matrix of size $r_{k-1} \times r_k$. In the above equation, the product of the matrices on the right side is a matrix of size $r_o \times r_d$. It is important to note that here the matrix $\mathbf{G}_k(i_k)$ is a 3-D array with size $r_{k-1} \times n_k \times r_k$. So, the above equation can be rewritten as:

$$\mathcal{A}(i_1, i_2, ..., i_d) = \sum_{\alpha_0,...,\alpha_{d-1},\alpha_d} \mathbf{G}_1(\alpha_o, i_1, \alpha_1)\mathbf{G}_2(\alpha_1, i_2, \alpha_2)...\mathbf{G}_d(\alpha_{d-1}, i_d, \alpha_d). \tag{3-3}$$

In the above equation, the value of α_k ranges from 1 to r_k over the summation. The product of the parameter dependent matrices in the above equation is a matrix of size $r_o \times r_d$. For the above formulation, the condition $r_o = r_d = 1$ is imposed as a boundary condition. In this decomposition method, the ranks r_k are called TT-ranks and the cores $\mathbf{G}_k$ are known as cores of the TT-decomposition.
Any n-dimesnional tensor $\mathcal{A}$ is said to be in the TT-format with cores $\mathbf{G}_k$ if the elements of the tensor are given by the formulation given in equation 3-3. The bound on r_k can be

Tensor Format	Storage Complexity
Full tensor	$\mathcal{O}(I^N)$
CP-decomposed tensor	$\mathcal{O}(NIR)$
Tucker-decomposed tensor	$\mathcal{O}(NIR + R^N)$
TT-decomposed tensor	$\mathcal{O}(NIR^2)$

Table 3-1: Storage complexities of different tensor decomposition methods [40].

obtained easily by checking the rank of the unfolding or matricized matrix of tensor $\mathcal{A}$:

$$\mathbf{A}_k = \mathbf{A}_k(i_1, ...i_k; i_{k+1}...i_d) = \mathbf{A}(i_1, ..., i_d). \tag{3-4}$$

The size of the matrix $\mathbf{A}_k$ is $\Pi_{s=1}^{k} n_s \times \Pi_{s=k+1}^{d} n_s$. Hence, if the rank of the unfolding matrix $\mathbf{A}_k = r_k$, then it means that a decomposition with TT-ranks not higher than r_k exists.

The storage complexities for each type of tensor decomposition for a N- way tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times ... \times I_N}$ are given in table 3-1. $I = \max(I_1, ..., I_N)$ and R is the maximum rank value of each tensor decomposition, i.e for TT-decomposition, $R = \max(R_0, ..., R_N)$. From table 3-1 it can clearly be seen that the storage complexity for a tensor represented in the TT-decomposition format is lower than that for a full tensor. This is because instead of N being an exponential for the full tensor, it becomes a factor in case of the TT-decomposed tensor.

3-2 TT-decomposition method for NLP

With the development of different network architectures for NLP, the complexity and the size of the LM increased over the past years. This makes it hard to train models on low-end devices because of the large amount of memory that is occupied by these models [27]. This led to research for developing methods to compress the language models without compromising on the performance of the LMs. The tensor-train method was first used by Novikov et al. [31] for NNs to compress the size of the model by reducing the number of parameters while preserving the power of the NN. The idea was to convert the weight matrices of the fully-connected layers in the NN to the tensor train format.

3-2-1 TT-embedding layers

In 2020, Hrinchuk et al. [19] developed a novel method of parameterizing embedding layers based on the TT-decomposition which would compress the model with a small increase/decrease in the performance. Deep NN architectures (NNs used in deep learning) consist of embedding layers that map input words into continuous word representations. As mentioned earlier, as the network architectures became deeper, with more model parameters and hyper-parameters, the memory required to store and run these NNs also increased in size. One of the solutions proposed to deal with the increase in memory requirements was to compress the NNs for the application of NLP [39, 38].
In the work done by Hrinchuk et al. [19], a parameter-efficient embedding layer known as the

TT-embedding layer was developed which can be plugged into any existing model and then trained end-to-end. As most of the parameters in NLP models occupy the embedding layers, the main idea is to compress these embedding layers to reduce the size of the entire language model.
To compress the embedding layers, the embedding matrix of the LM is compressed. In order to do this, the embedding matrix of the LM is reshaped and represented as a full tensor and then it is represented in the TT-decomposition format. This way the storage complexity reduces as the full tensor is represented in the TT-format.

The advantages of plugging in the TT-embedding layer into a model are firstly, instead of storing a huge embedded matrix, a sequence of much smaller 2-D and 3-D tensors are needed for reconstructing the required embeddings. This compresses the size of the model.
Secondly, the overall number of parameters can be small (in the order of 0.1 - 0.3 million) during training which increases the efficiency of training in case of limited resources. The performance of TT-embeddings was tested on several NLP tasks and the results were compared with existing state-of-the-art LMs. It was observed that TT-embeddings can replace standard embeddings to compress the LM with possibly a small drop in the performance of the LM for the respective NLP tasks.

3-2-2 TT-embeddings using the TT-matrix

Consider a N-way tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times ... \times I_N}$ with the entries $\mathcal{X}(i_1, ..., i_N)$ such that $(1 \leq i_k \leq I_k)_{k=1}^N$. When the tensor is represented in the TT-format, which is described in the previous chapter, the minimal values of $(R_k)_{k=1}^{N-1}$ for which the TT-decomposition exists are called TT-ranks. The number of degrees of freedom in such a decomposition is evaluated as:

$$\sum_{k=1}^{N} R_{k-1} I_k R_k. \tag{3-5}$$

Hence, in the case of small ranks, the number of parameters required to store a tensor in the TT-representation is smaller than the number of parameters required to store a full tensor corresponding to the same size. Consider a matrix $\mathbf{X}$ of size $I \times J$. Given two arbitrary factorizations of the sizes, $I = \Pi_{k=1}^N I_k$ and $J = \Pi_{k=1}^N J_k$, the matrix can be reshaped and transposed into a N-way tensor $\mathcal{X} \in \mathbb{R}^{I_1 J_1 \times ... \times I_N J_N}$. After this, the tensor can be represented in the TT-format, to represent it in a compact manner. The tensor formed can be represented in the TT-format as:

$$\mathcal{X}((i_1, j_1)...(i_N, j_N)) = \mathcal{G}^{(1)}[(i_1, j_1), :]...\mathcal{G}^{(n)}[:, (i_N, j_N)]. \tag{3-6}$$

The above representation of the matrix in the TT-format is known as the TT-matrix. The factorizations $(I_1, I_2, ..., I_N) \times (J_1, J_2, ..., J_N)$ is referred to as shape of the TT-matrix. A visual representation of how a TT-matrix is obtained can be seen in figure 3-3. It can be observed that the blue color depicts how a single element in the initial matrix is transformed into a product of vectors and matrices in the TT-format.

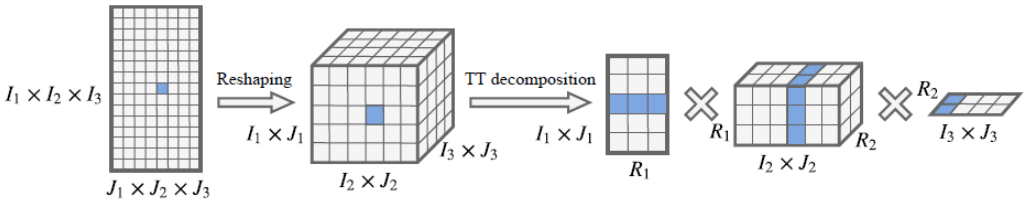


Figure 3-3: Visual Representation of the TT-matrix is obtained [19].

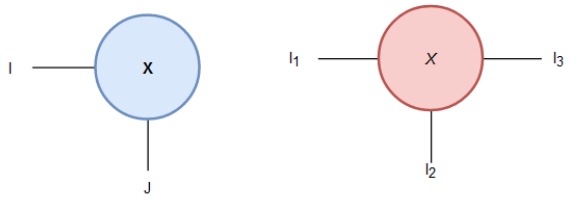


Figure 3-4: Tensor diagram for a matrix **X** and a 3-way tensor $\mathcal{X}$.

3-2-3 Upper bound on TT-rank

For a given factorization of the embedding matrix in the TT-matrix format, an upper bound on the value of each TT-rank can be calculated using a simple formula. Consider a matrix $\mathbf{X}$ of size $I \times J$ with factorizations of the sizes, $I = \Pi_{k=1}^{N} I_k$ and $J = \Pi_{k=1}^{N} J_k$. The matrix can be reshaped and and transposed into an N-way tensor $\mathcal{X} \in \mathbb{R}^{I_1 J_1 \times ... \times I_N J_N}$. As mentioned earlier, after applying TT-decomposition the tensor formed can be represented in the TT-m format as:

$$\mathcal{X}((i_1, j_1)...(i_N, j_N)) = \mathcal{G}^{(1)}[1, i_1, j_1, r_1]\mathcal{G}^{(2)}[r_1, i_2, j_2, r_2]...\mathcal{G}^{(n)}[r_{N-1}, i_N, j_N, 1]. \tag{3-7}$$

In the above TT-matrix format, an upper bound for the TT-ranks (maximum possible value such that there is no over parameterization) r_1 to r_{N-1} can be calculated. The equations to calculate the maximum possible value that each TT-rank can take are given as follows:

$$r_1 \leq \min(I_1 * J_1, I_2 * J_2 * I_3 * J_3 * ... * I_N * J_N), \tag{3-8}$$

$$r_2 \leq \min(I_1 * J_1 * I_2 * J_2, I_3 * J_3 * ... * I_N * J_N) \tag{3-9}$$

$$r_{N-1} \leq \min(I_1 * J_1 * I_2 * J_2 * I_3 * J_3 * ... * I_{N_1} * J_{N-1}, I_N * J_N). \tag{3-10}$$

3-2-4 Tensor diagrams

A tensor represented in the TT-decomposition format and the TT-matrix format can be visualized using tensor diagrams. The idea of tensor diagrams was introduced by Penrose [33]. A tensor can be denoted by a node with branches attached to it. A matrix $\mathbf{X}$ node has two branches, the I and the J attached to it. A 3-way tensor $\mathcal{X}$ has three branches, I_1, I_2 and I_3 attached to it. This is illustrated in figure 3-4.

The tensor diagram for a matrix $\mathbf{X}$ of size $I \times J$ in the TT-matrix format is given in figure 3-5. The indices $I = I_1 \times I_2 \times I_3$, $J = J_1 \times J_2 \times J_3$.

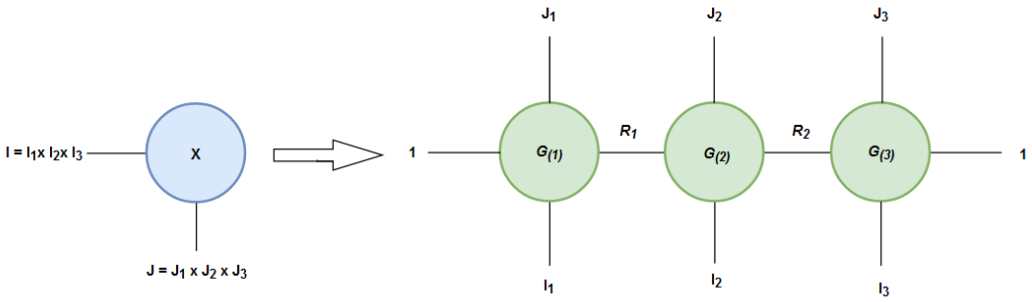


Figure 3-5: Tensor diagram for a matrix **X** represented in the TT-matrix format. Each index has 3 factors.

Procedure to obtain TT-embedding

The TT-embedding is a layer with trainable parameters (TT-cores) represented as a TT-matrix of the tensor shape $(I_1, I_2, ...I_N) \times (J_1, J_2, ...J_N)$ which can be transformed into an embedding layer $\mathbf{E}$ with size $I \times J$ where, $I = \Pi_{k=1}^{N} I_k$ and $J = \Pi_{k=1}^{N} J_k$. To specify the shapes of TT–cores one has also to provide the TT–ranks, which are treated as hyperparameters of the layer and explicitly define the total compression ratio.
The first step to compute the embedding for a word indexed i in the vocabulary is to map the row index i into the N-dimensional vector index. Then the components of the embedding are calculated using the equation 3-6. The pseudo code for the mapping of i into $(i_1, i_2, ..., i_N)$ is given in figure A-1 of appendix A.

The procedure to construct a TT-embedding layer for a vocabulary of size I and embedding dimension of size J, and then to train the model with an embedding layer, is given below:

- Firstly, provide factorizations of I and J. $I = I_1 \times I_2 \times ... \times I_N$ and $J = J_1 \times J_2 \times ... \times J_N$. Specify the set of the TT-ranks $(R_1, R_2, ..., R_{N-1})$.
- Next, initialise the set of parameters of the embedding:

$$\theta = (\mathcal{G}^{(k)} \in R^{R_{k-1} \times I_k \times J_k \times R_k})_{k=1}^{N}. \tag{3-11}$$

- During the training stage, given a batch of indices, compute the corresponding embeddings (the set of TT-cores $\mathcal{G}^{(k)}$ using the equation 3-6.
- Computed embeddings can be followed by any standard layer such as LSTM or self-attention [45] and trained with back propagation using the Adam optimizer, since they depend on the parameter θ (weights and biases).

The initialisation of the embedding matrix is important in the case of TT-embeddings. For TT-embeddings, only the TT-cores can be initialised. The distribution of the elements resulting in the matrix $\mathbf{X}$ is non-trivial. When we initialise the TT-core as:

$$\mathcal{G}^{(k)}(r_{k-1}, i_k, r_k) \sim \mathcal{N}(0, 1). \tag{3-12}$$

the resulting distribution of matrix elements $\mathbf{X}(i,j)$ has the property:

$$E[\mathbf{X}(i,j)] = 0. \tag{3-13}$$

and

$$\Sigma^2 := Var[\mathbf{X}(i,j)] = \Pi_{k=1}^{N} R_k. \tag{3-14}$$

In order to achieve the condition given in equation 3-12, the TT-core is initialised as:

$$\mathcal{G}^{(k)}(r_{k-1}, i_k, r_k) \sim \mathcal{N}\Big(0, (\frac{\sigma}{\Sigma})^{2/N}\Big). \tag{3-15}$$

TT-embeddings initialised in the way mentioned above, improved performance. Apart from the initialisation, two more important structure-specific hyper-parameters are the TT-shapes and TT-ranks.
For TT-embedding, the vocabulary of size I can be represented by any factorization $\Pi_{k=1}^{N} I_k = \tilde{I} \geq I$. In order to achieve a higher compression rate, for a fixed value of $\tilde{I}$, the factors $(I_k)_{k=1}^{N}$ should be as close to each other as possible.

The values of the TT-ranks directly define the compression ratio, so choosing the values which are very large or too small will result in a performance drop.

The TT-embedding layer approach was tested on three NLP tasks. The NLP tasks that were chosen are:

- Sentiment analysis: To predict the polarity of a sentence.
- Neural machine translation (NMT) : This task involves translating text from one language to another,
- Language modeling task in the case of extremely large vocabularies.

Only the results of the sentiment analysis task will be discussed in detail in the following section. The TT-embeddings approach was tested on a bi-directional LSTM architecture. The performance of the baseline model where the embedding matrix is not factorized is compared with factorized TT-embedding models.

Sentiment analysis

For this experiment, a bi-directional LSTM architecture that has a forward hidden and a backward hidden layer was chosen and tested on the IMDB dataset. The size of the hidden layer is $h = 128$ and the dropout parameter of P_{drop} of 0.5 was chosen as a form of regularization in the LSTM architecture. More details about the bi-directional LSTM architecture can be found in [46].
The IMDB dataset is a collection of 50,000 movie reviews from the movie rating website IMDB [2]. Each movie review has a +ve or a -ve review sentiment labelled to it. Neutral reviews are not considered in the IMDB dataset.
The most frequently occurring words were chosen from both datasets. 25000 words were

Dataset	Model	Embedding Shape	Test Acc.	Emb Com.	Params.
IMDB	Full	25000×256	0.886	1	7.19M
IMDB	TT1	$(25,30,40) \times (4,8,8)$	0.871	93	0.86M
IMDB	TT2	$(10,10,15,20) \times (4,4,4,4)$	0.888	232	0.82M
IMDB	TT3	$(5,5,5,5,6,8) \times (2,2,2,2,4,4)$	**0.897**	441	0.81M

Table 3-2: Results for sentiment analysis. Different embedding matrix factorizations are considered for all the models which are given under 'Embedding Shape'. The accuracy scores for the models based on their performance on the IMDB dataset is given under 'Test Acc.'. The Embedding compression ratio and total parameters required for each model are also reported [19].

taken from the IMDB dataset and the value of TT-ranks was set to 16 for all experiments. The embedding compression ratio is the ratio between the number of parameters in the full embedding layer and the TT-embedding layer. The results of the experiments conducted are shown in table 3-2.

It is important to note here that each model configuration is run for 50 epochs and the maximum test accuracy obtained while training the model is reported as 'Test Acc.' in table 3-2. The embedding compression ratio is the ratio between the number of parameters in the original embedding matrix and the number of elements in the TT-matrix. The total number of parameters is the number of parameters in the entire LM, including the parameters in the embedding layer and the fully-connected layer of the bi-directional LSTM model.
From table 3-2, it can be observed that when the factorization of the embedding matrix consists of more number of factors, the compression ratio is high. While, TT1 and TT2 have 3 and 5 factors respectively, in the case of TT3 model, the embedding matrix has 6 factors. The full model with 7.19 M parameters achieves a test accuracy of 0.886 but the TT3 model with only 0.81 M parameters, achieves a test accuracy of 0.897. The TT3 model achieves the best test accuracy amongst all the three TT models.
Hrinchuk et al. [19] conclude that by having more number of factors for the embedding matrix, the test accuracy of the model improves.

Another important point that should be noted here is that even though the embedding compression ration is reported to be as high as 441 (for the case of TT3) it can be seen that the total number of parameters reduces from 7.19 M to 0.81 M. This means that the compression of the total number of parameters in the model (total parameters in the full model/total parameters in the TT3 model) is only about 8.87 times.

In the work done by Hrinchuk et al. [19], TT-embedding layers were also plugged into the transformer-big model for the application of NMT. NMT is a complex task and adding TT-embeddings did not improve the performance of the model, although it did achieve a reduction of model parameters in the transformer model.
Apart from the TT-decomposition, another tensor decomposition method called the block-term tensor decomposition method was also used to compress the size of the transformer architecture. Ma et al. [25] introduced the tensorized Transformer, which used the block-term tensor decomposition method to compress the transformer-XL architecture. Refer to [25] to learn more about the transformer-XL model and its compression using the block-term

decomposition method.

The structure of any full uncompressed LM is altered when TT-embeddings replace the original embedding layer in the model. This means that the hyper-parameter TT-shape which is the shape of the embedding matrix in the TT-format and the hyperparameters TT-ranks play an important role in determining the effect of the TT-embedding on a particular LM. Hrinchuk et al. [19] do not provide detailed analysis as to how the TT-ranks were chosen to be 16 for all experiments. They also don't explain the effects of the factorization of the embedding matrix and thus the TT-ranks on compression and the test accuracy of the bi-directional LSTM model or the transformer-big model.

To answer the research gaps found in the work done by Hrinchuk et al. [19], two research questions are formulated given in the introductory chapter.
The methodology developed and the experimental setup used to answer the research questions introduced in this thesis will explained in the next chapter.

Chapter 4

Factorization of the embedding matrix and its effects

The research conducted by Hrinchuk et al. [19] was executed using PyTorch, an end-to-end machine learning framework. A code was developed on the framework using Python, which is available on GitHub as an open source code (https://github.com/KhrulkovV/tt-pytorch). The code developed in Python was implemented on an NVIDIA DGX-1 workstation specifically designed for AI research. The DGX-1 setup consists of eight NVIDIA V100 Tensor Core GPUs and two 20-core Intel Xeon CPUs.

Due to monetary and computational limitations, a DGX-1 work station or a cloud GPU setup could not be arranged for the thesis work. A PC setup consisting of a 4GB NVIDIA 1650 GTX GPU and a 4-core AMD Ryzen 5 3550H CPU was used to perform all experiments.
Instead of a bi-directional LSTM model that was used in [19], a single layer standard LSTM model consisting of an input layer, a hidden layer with 128 hidden units, a fully-connected layer, and an output layer was chosen as the LM for the NLP task of sentiment analysis. The IMDB dataset is used as the input dataset for this particular task.
Before the methodology to answer the effect of factorization of the embedding matrix on the performance of the LSTM model is described, first the LSTM model architecture used for the NLP application of sentiment analysis will be discussed in detail.

4-1 Spacy tokenizer

The first part of the LSTM architecture consists of a tokenizer which tokenizes each word, and an integer number is assigned to each word. The Spacy tokenizer available as a library in Python, is used for this step [44]. The Spacy tokenizer accepts an input sequence of words, in this case a movie review from the IMDB dataset.
The tokenizer first accepts the raw test data of each movie review and separates words in

a sentence in the text file into tokens based on white spaces between. Punctuation marks like commas, quotes, etc... are also considered as tokens. Words like "U.K." or "N.Y.C" are considered as a single token. These are exceptions to the tokenizer.
After the text data from each movie review is split into tokens, the tokenizer builds a vocabulary where all the tokens are stored as integers. Each word has a different integer value assigned automatically. Spacy also has a function which allows the user to set the number of tokens that have to be generated for each text data [44]. The number of tokens for each movie review is set at 100. In the work done by Hrinchuk et al. [19], this was set to 1000.

Choosing the fixed number of tokens to be 100 reduces the time required for the spacy tokenizer to tokenize each movie review. This decision was taken because of the computational limitations of this thesis work. It should be noted that the IMDB dataset only contains positive-labeled reviews and negative-labeled reviews. This is an advantage as most of the time, an LM can already predict the correct label by looking at the first few sentences of the review, for instance. If neutral reviews were included in the IMDB dataset, restricting the number of tokens to 100 would have posed a problem for the LM, leading to poor test accuracy. After each movie review is tokenized, these tokens are sent as an input sequence to the embedding layer.

4-2 The embedding layer

The embedding layer in the LSTM model, receives each token as an input and produces a word embedding for each word. Recall that in section 2-4 the concept of word embeddings was introduced. Similar to the that in the case of the skip-gram model, in this case the LSTM model takes a vocabulary consisting of a set of words as an input and projects each word onto a N-dimensional vector space. Here, the number of words in the vocabulary is taken as 25,000 and the N-dimensional vector is taken as 128. The N-dimensional vector space is represented by the hidden layer, which has 128 LSTM cells. Hence, the embedding matrix in this model is of size 25000×128.

During training of the LSTM model, each input word is sent to the hidden layer, which consists of LSTM cells. The word embeddings are obtained by back-propagating the error where the error is reduced using the Adam optimizer. The learning rate (a hyper-parameter) for the Adam optimizer is chosen to be 1×10^{-3} (the same taken by Hrinchuk et al. [19]). As mentioned before, in LMs the embedding layer accounts for most of the network parameters, which is why this layer is compressed. A point to be noted here is that the input data is sent to the LSTM model in batches and the batch size is taken to be 128.

4-3 Fully connected layer and output layer

In this model, after the hidden layer, the input data is sent through a fully-connected layer which maps the output of the hidden layer to a desired output size. The data is then sigmoid activated to get an output between 0 and 1 and finally, the output layer consists of 1 neuron which gives a predicted output. The predicted output is compared to the true output value

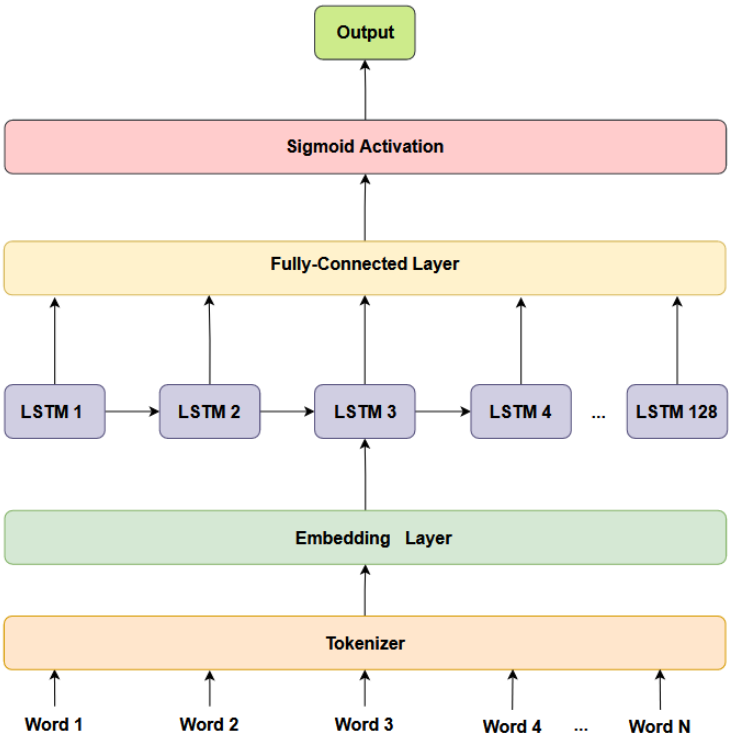


Figure 4-1: A visualization of the LSTM model that is used in this work. The input consists of words from word 1 to word N that are tokenized by the Spacy tokenizer and integers are assigned to them. The Embedding layer is where we obtain word embeddings for each word in the vocabulary. This embedding layer is compressed using the TT-decompostion method. The vocabulary here has 25000 words. The hidden layer consits of 128 LSTM cells. The data from the hidden layer is sent to the fully-connected layer, after which each output is sigmoid activated to get the final predicted output from the output node.

and the error is back-propagated through the network. As mentioned before, the error in the network is minimized by updating the weights and biases of the network using the Adam optimizer.
The LSTM model is trained for 10 epochs, keeping in mind the computational limitations. Tokenization and the training of the LSTM model for 10 epochs takes around 25-30 minutes to complete on the PC setup used for this work.

Unlike in the work done by Hrinchuk et al. [19], there is no dropout added to the network. Dropout is a form of regularization that can be added to the network to reduce the number of parameters. This is not done in order to clearly understand the effect of TT-decomposition on the compression and the performance of the LSTM model. A visualization of the LSTM model architecture used for the thesis work is given in figure 4-1.
Out of the 50,000 movie reviews, the first 25,000 movie reviews are taken as training data, the next 12,500 reviews are taken as validation data, and the last 12,500 movie reviews are taken as testing data. After the LSTM model is trained on the training data, its performance is checked using the testing data.

Model	Embedding Shape
Full	25000×128
TT1	$(25,30,35) \times (4,4,8)$
TT2	$(5,5,7,10,15)) \times (2,2,2,4,4)$
TT3	$(2,3,5,5,5,5,7) \times (2,2,2,2,2,2,2)$

Table 4-1: Different factorization considered to investigate their effect on the LSTM model. Each TT-models have a difference of two factors.

4-4 Methodology - research question 1

In an LSTM model used to perform NLP, when standard embeddings are replaced by TT-embeddings, how does the factorization of the embedding matrix and subsequently the TT-ranks effect the test accuracy and the compression rate?

To compress the LSTM model, the embedding matrix is first reshaped into a tensor. The tensor is then represented in the TT-matrix format. This way, the original embedding matrix can be factorized. The vocabulary size I and the N-dimensional vector space J are factorised into different factors. For TT-embeddings, the vocabulary size is taken as 26250 (which is greater than 25000, the number of words). The following methodology is implemented to investigate the effect of factorizations on the compression and performance of the LSTM model:

- Firstly, three different factorizations of the embedding matrix are considered.
- For each factorization, a different TT-embedding is obtained, which has a different structure consisting of different TT-cores.
- The maximum possible value for each TT-rank in the factorization is calculated.
- TT-ranks are hyperparameters that can be manually defined. The TT-ranks are chosen in a manner such that different compression rates are obtained. This is done for each factorization.
- Based on the chosen TT-ranks, TT-embeddings are obtained as described in section 3-2-4.
- The performance of the LSTM model is compared under different factorizations of the original embedding matrix.
- The effect of varying each TT-rank in the case of a particular factorization on the performance of the model is analyzed.

The different factorizations that are considered are given in table 4-1. The full model is the case where the original embedding matrix is considered. The TT1 model has both the vocabulary size I and the J dimensional space factorized into three factors arranged in ascending order. The TT2 model has the embedding matrix factorized into five factors and the TT3 model has the vocabulary size I and J dimensional space factorized into their prime factors, respectively, with seven factors.

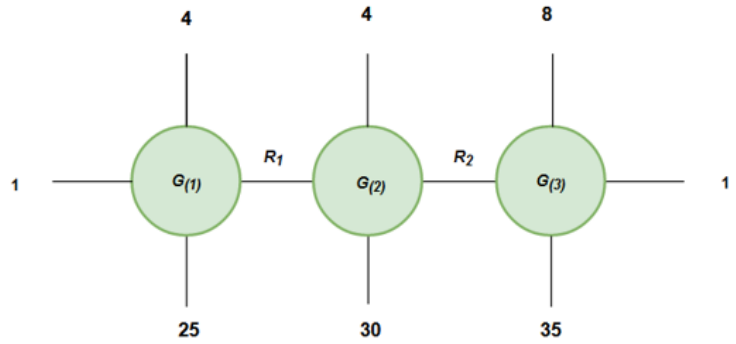


Figure 4-2: Tensor diagram for TT1 Model. The TT-ranks R_1 and R_2 are hyper-parameters that link each TT-matrix core $\mathbf{G}_{(k)}$ (k = 1 to 3) to the next core.

Using the set of equations 3-8 in section 3-2-3, the maximum possible value of the TT-ranks for TT1, TT2 and TT3 models are calculated.

4-5 TT1 model

The TT1 model has the embedding shape (25,30,35) × (4,4,8) with three factors, all of them arranged in ascending order based on the numerical value of the factors. In the TT1 model, there are two TT-ranks R_1 and R_2 which are hyperparameters of the model. The TT1 model can be visualized using the tensor diagram given in figure 4-2. For the TT1 model, the maximum possible value for each TT-rank is calculated as follows:

$$R_1 \leq \min(25 * 4\ ,\ 30 * 4 * 35 * 8) \leq 100,$$
$$R_2 \leq \min(25 * 4 * 30 * 4\ ,\ 35 * 8) \leq 280.$$

The ranks R_1 and R_2 are varied to get eight different compression rates. The following equation defines the compression rate in percentage for all factorizations:

$$\text{Compression rate} = \frac{\text{No. of params. in full model - No. of params. in TT model}}{\text{No. of params. in full model}} * 100 \tag{4-1}$$

The full model has 3.332225 M parameters.
Eight different compression rates ranging from 30% to 95% are considered. For each TT model, the same compression rate can be achieved in more than one way by selecting the TT-ranks. This means that a particular compression rate, can be obtained with more than one combination of the TT-ranks. In the TT1 model, different compression rates are obtained first by varying R_1 and keeping R_2 fixed, then by varying R_2 while R_1 is fixed and finally by varying both R_1 and R_2.

It is important to note that all the TT-ranks in a particular factorization cannot have the maximum possible rank because this would lead to over parameterization of the original LSTM

TT1 Ranks	Parameters	Compression %	Max Acc %
Full Model	3.332225 M	-	78.04%
(100,178)	2.328065 M	30.29%	78.59%
(90,168)	2.002665 M	39.90%	79.40%
(100,120)	1.615825 M	51.51%	79.18%
(52,178)	1.297985 M	61.04%	79.64%
(73,93)	0.980245 M	70.58%	79.22%
(35,120)	0.673325 M	79.79%	79.46%
(100,15)	0.326425 M	90.20%	79.86%
(1,1)	0.132725 M	96.01%	69.50%

Table 4-2: Results obtained for the TT1 model. The TT1 ranks are reported as (R_1,R_2), 'Max Acc' is the maximum accuracy % that is achieved in 10 epochs by the LSTM model. The compression rates are defined with respect to the full model parameters (3.332225 M).

model and the TT model would then have more parameters compared to the full model. Hence, when the ranks are being selected, the TT-rank, which has the highest possible TT-rank, is first reduced to obtain 30% compression, and then the TT-ranks are varied to get other compression rates. In the case of the TT1 model, the maximum possible TT-rank is higher for rank R_2 which is 280. So, R_2 is first reduced, when R_1 is kept constant at 100 to get 30% compression. The value of rank R_2 in this case is 178. The following steps are followed to investigate the effects of the factorization and TT-ranks on the TT1 model:

- Compression rates of 30%, 40%, 50%, 60%, 70%, 80%, 90% and 95% are obtained by varying the TT-ranks as mentioned above (The compression rates are not exactly equal to the numbers mentioned here, they might vary by 1% from that). Three different combinations are possible in the case of TT1. The test accuracy (obtained when the test dataset is sent as input) after 10 epochs is also obtained for every run of the LSTM model. A minimum of two and a maximum of three runs are conducted for each combination of TT-ranks and the scenario with the best test accuracy is chosen.
- For the same amount of compression, the combination of the TT-ranks which yields the best test accuracy is chosen. Then the maximum test accuracy that is achieved in this combination of TT-ranks for the respective amount of compression is reported.
- The effect of the individual TT-ranks R_1 and R_2 is analysed by looking at how the test accuracy varies with respect to each TT-rank.
- Additionally, the evolution of the loss value when training and test datasets are given as inputs is also obtained.

4-5-1 TT1 model results

The final combination of TT-ranks chosen for various compression rates along with the number of parameters (M) and maximum test accuracy (%) for the TT1 model are given in table 4-2. It can be observed from table 4-2 that the combination of R_1 and R_2 as 100 and 178 respectively gives a compression rate of 30.29%. Keeping that as a reference, the TT-ranks

are varied.
The maximum test accuracy achieved in the case of the full model when the embedding matrix is not factorized is 78.04%. It can be observed from table 4-2 that even at 90.20% of compression rate, the TT1-model has a maximum accuracy of 79.86%. The number of parameters here is reduced from 3.332225 M to 0.326425 M. But the performance of the LSTM model does not deteriorate but rather slightly improves from 78.04% in the full model to 79.86% in the TT1 model.
At a compression rate of 96.01%, the maximum test accuracy drops to 69.50%, compared to a test accuracy of 78.04% in the full model. The TT-ranks R_1 and R_2 have the lowest possible value (1,1) for the compression rate of 96.01%, with 0.132725 M parameters in the model. This also shows that only the embedding layer of the LSTM model is compressed while the parameters in the other layers of the model are not compressed. Hence, a compression of more than 96% cannot be achieved.
All the other combinations of TT-ranks considered for the TT1 model along with the maximum test accuracy in each case are reported in section B-1 of appendix B.

4-5-2 Effect of TT-ranks in TT1 model

The effects of each rank are investigated by looking at the change in maximum test accuracy with respect to the change of each TT-rank. This is done by looking at how the maximum test accuracy varies when one TT-rank is varied while the other is kept fixed. The maximum test accuracy results for the TT1 model when rank R_1 is varied and rank R_2 is varied are reported in tables 4-3a and 4-3b.
The plot that shows change of rank R_1 with respect to the maximum test accuracy is given

R1 Rank	Max Acc%
100	78.59%
84	78.78%
68	79.10%
52	79.64%
38	79.12%
20	79.68%
5	79.46%
1	78.46%

(a) The rank R_1 is varied from 100 to 1, when R_2 is fixed at 178.

R2 Rank	Max Acc%
178	78.59%
160	78.90%
120	79.18%
95	78.50%
70	78.68%
45	79.18%
15	79.86%
1	72.85%

(b) The rank R_2 is varied from 178 to 1, when R_1 is fixed at 100.

Table 4-3: Effects of ranks R_1 and R_2 on the maximum test accuracy 'Max Acc %' of the TT1 model.

in figure 4-3. In this case, the rank R_1 is varied while R_2 is fixed at 178. The plot that shows change of rank R_2 with respect to the maximum test accuracy is given in figure 4-4.

An important point to note is that when, $R_1 = 1$ and $R_2 = 178$, the number of parameters in the model is 0.203525 M. When $R_1 = 100$ and $R_2 = 1$, the number of parameters in the model is only 0.154505 M.
Hence, in the figure 4-4, when the rank of R_2 reduces to 1, the test accuracy drops to 72.85%. However, when R_1 reduces to 1, the maximum test accuracy drops to only 78.46% as there

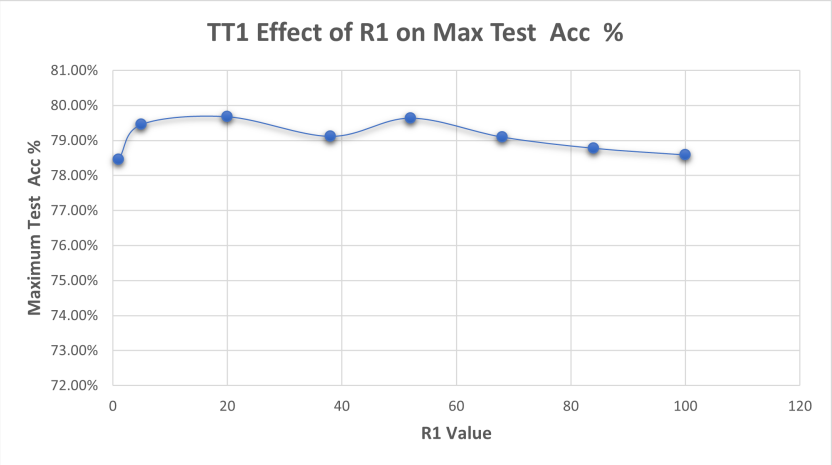


Figure 4-3: Plot between value of rank R_1 and maximum test accuracy % for the TT1 model. The rank R_1 is varied from 100 to 1 while rank R_2 is always 178.

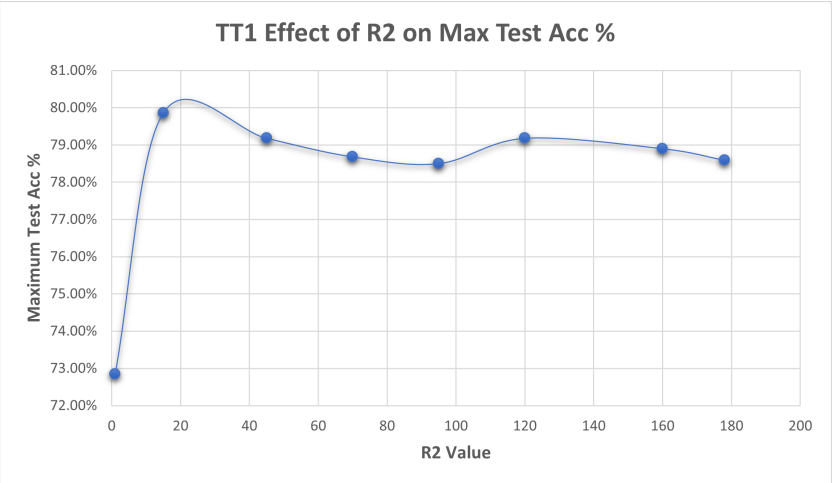


Figure 4-4: Plot between value of rank R_2 and maximum test accuracy % for the TT1 model. The rank R_2 is varied from 178 to 1 while rank R_1 is always 100.

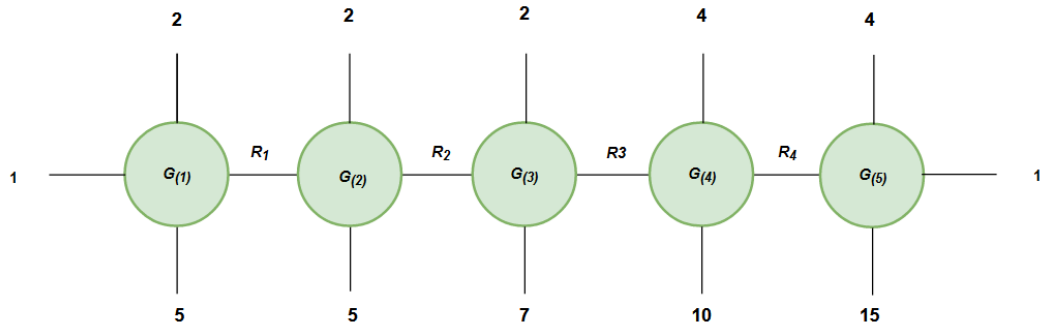


Figure 4-5: Tensor diagram for TT2 Model. The TT-ranks R_1, R_2, R_3 and R_4 are hyper-parameters that link each TT-matrix core $\mathbf{G}_{(k)}$ ($k = 1$ to 5) to the next core.

are a greater number of parameters in this case.
From the plots shown, it can be seen that when R_1 increases from 5 to 100, there is an increase and a decrease in maximum test accuracy, but it stays within the range of 79.68% and 78.59%. A drastic change in the maximum test accuracy is not visible.
In the case of R_2, there is a sudden increase in the maximum test accuracy when the value of R_2 increases from 1 to 15. But a similar pattern to that of R_1 is observed when R_2 varies from 15 to 178.
The fluctuations in the maximum test accuracy might be because of the random noise generated during the training of the LSTM model or over-fitting of the model to the training data.

It can be concluded that in the case of the TT1 model, the maximum test accuracy is not particularly sensitive to changes in either R_1 or R_2.

4-6 TT2 model

The TT2 model has the following embedding shape (5,5,7,10,15) × (2,2,2,4,4) with five factors, all of them arranged in ascending order. In the TT2 model, there are four TT-ranks R_1, R_2, R_3 and R_4 which are hyperparameters of the model. The TT2 model can be visualized using the tensor diagram given in figure 4-5.

The same procedure that was followed to investigate the effects of the TT-ranks on the TT1 model will be followed for the TT2 and TT3 models as well. In the TT2 model, eight different compression rates are obtained by selecting a different combination of all the TT-ranks. The maximum possible values for each rank of the TT2 model are calculated as:

$$R_1 \leq 10, R_2 \leq 100, R_3 \leq 1400, R_4 \leq 60.$$

Like in the case of TT1, first the rank with the highest maximum possible value R_3 is varied to get a compression rate close to 30%, while all other ranks are kept fixed. At 30.61% compression, the rank R_3 is 570. In the case of TT2 also, the compression rate is calculated with respect to the full model parameters (3.332225 M).
After most of the possible combinations of TT-ranks for which different compression rates

TT2 Ranks	Parameters	Compression %	Max Acc %
Full Model	3.332225 M	-	78.04%
(10,100,570,60)	2.311925 M	30.61%	79.30%
(10,60,570,60)	1.988725 M	40.31%	80.20%
(10,100,570,31)	1.648985 M	50.51%	79.32%
(10,100,310,60)	1.323925 M	60.26%	80.37%
(10,100,222,60)	0.989525 M	70.30%	80.26%
(10,40,170,60)	0.643125 M	80.69%	80.42%
(10,100,47,60)	0.324525 M	90.26%	79.53%
(10,10,1,1)	0.133565 M	95.99%	68.05%

Table 4-4: Results obtained for the TT2 model. The TT2 ranks are reported as (R_1,R_2,R_3, R_4), 'Max Acc' is the maximum accuracy % that is achieved in 10 epochs by the LSTM model. The compression rates are defined with respect to the full model parameters (3.332225 M).

are obtained, the final selected set of TT-ranks for the TT2 model which had the best test accuracy is chosen and the results are reported.

4-6-1 TT2 model results

The final combination of TT-ranks chosen for various compression rates along with the number of parameters (M) and maximum test accuracy (%) for the TT2 model are given in table 4-4. The combination of (R_1,R_2,R_3,R_4) as (10,100,570,60) gives a compression rate of 30.61%. Keeping this as a reference, the TT-ranks are varied for this model. It can be observed from table 4-4 that even at a compression rate of 90.26%, the TT2 model achieves a maximum test accuracy of 79.53% compared to 78.04% maximum test accuracy of the full model. At 95.99% compression, the test accuracy drops drastically from 79.53% to 68.05%.
All the other combinations of TT-ranks considered for the TT2 model along with the maximum test accuracy in each case are reported in section B-2 of the appendix B.

4-6-2 Effect of TT-ranks in TT2 model

The effect of each rank is investigated by looking at the change in maximum test accuracy with respect to the change in each TT-rank of the model. This is done similarly to that in the case of the TT1 model. Tables 4-5a and 4-5b shows the change in maximum test accuracy when R_2 and R_4 are varied. Table 4-6 shows the change in maximum test accuracy when R_3 is varied.

The plot that shows change in rank R_2 with respect to maximum test accuracy is given in figure 4-6. The plot that shows change in rank R_3 with respect to maximum test accuracy is given in figure 4-7 and the plot that shows change in rank R_4 with respect to maximum test accuracy is given in figure 4-8.

A plot is not given for change in rank R_1 with respect to maximum test accuracy as the maximum test accuracy was not particularly sensitive to changes in rank R_1.
It is important to note that when, R_2 has the lowest value of 1, the number of parameters in the TT2 model are 1.512005 M. The number of parameters when R_3 has the lowest value of

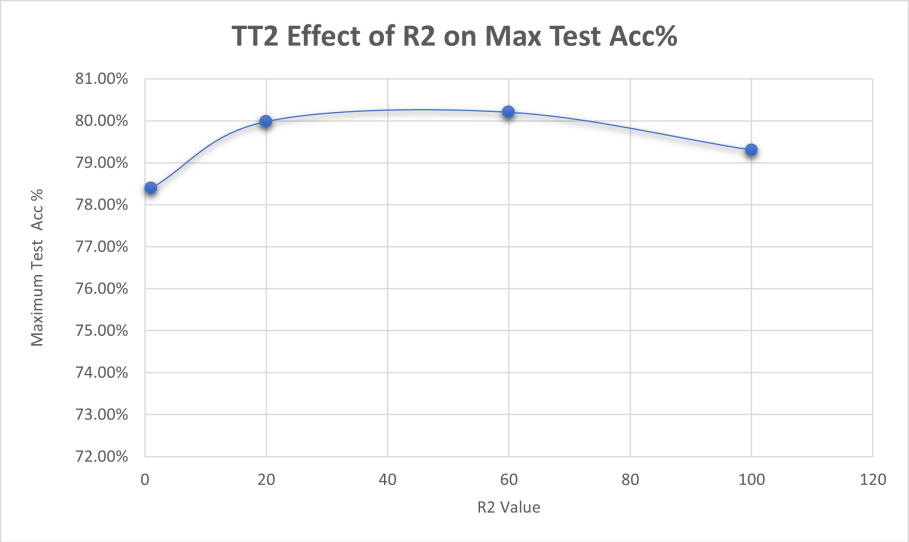


Figure 4-6: Plot between value of rank R_2 and maximum test accuracy % for the TT2 model. The rank R_2 is varied from 100 to 1 while ranks R_1, R_3 and R_4 are fixed at 10, 570 and 60 respectively.

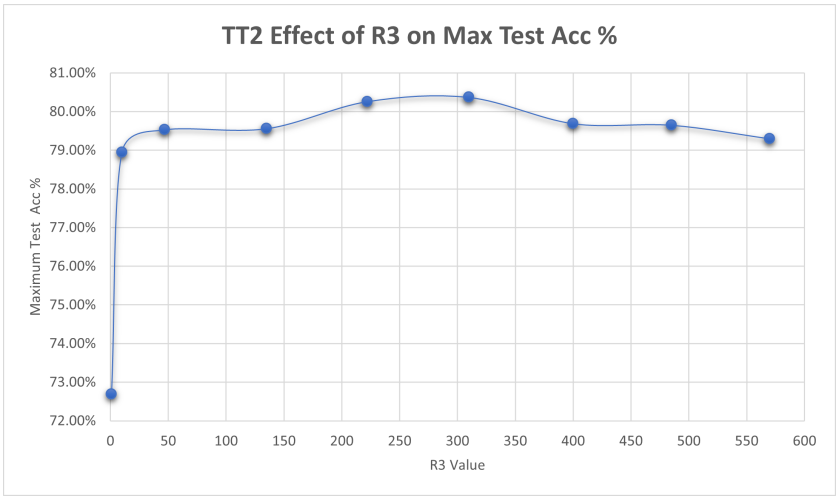


Figure 4-7: Plot between value of rank R_3 and maximum test accuracy % for the TT2 model. The rank R_3 is varied from 570 to 1 while ranks R_1, R_2 and R_4 are fixed at 10, 100 and 60 respectively.

R2 Rank	Max Acc%
100	79.30%
60	80.20%
20	79.98%
1	78.39%

(a) The rank R_2 is varied from 100 to 1, when R_1, R_3 and R_4 are fixed at 10, 570 and 60 respectively.

R4 Rank	Max Acc%
60	79.30%
31	79.32%
17	79.31%
1	74.65%

(b) The rank R_4 is varied from 60 to 1, when ranks R_1, R_2 and R_3 are fixed at 10, 100 and 570 respectively.

Table 4-5: Effects of ranks R_2 and R_4 on the maximum test accuracy 'Max Acc%' of the TT2 model.

R3 Rank	Max Acc%
570	79.30%
485	79.65%
400	79.69%
310	80.37%
222	80.26%
135	79.56%
47	79.53%
10	78.95%
1	72.69%

Table 4-6: Effects of ranks R_3 on the maximum test accuracy 'Max Acc %' of the TT2 model. The rank R_3 is changed from 570 to 1, when R_1, R_2 and R_4 are fixed at 10, 100 and 60 respectively.

1 is 0.149725 M. Similarly, when R_4 has the lowest value of 1, it is 0.963185 M.

From the plots, it can be seen that the maximum test accuracy % is not particularly sensitive to R_2 and R_3. In figure 4-6, it can be seen that maximum accuracy fluctuates between a range of 78.39% and 80.30% in the case of R_2. In figure 4-7, it can be seen that maximum test accuracy fluctuates between a range of 78.95% and 80.37% in the case of R_3. Only when R_3 is of value 1, the accuracy drops to 72.69% because the number of parameters is only 0.149725 M.
The fluctuations here in the maximum test accuracy might be because of the random noise generated during the training of the LSTM model or over-fitting of the model to the training data.

An interesting observation can be inferred from figure 4-8. Here, when the value of R_4 drops to 1, the maximum test accuracy drops to 74.65% while the number of parameters is 0.963185 M.
When R_3 is being varied, at a value of $R_3 = 222$, the number of parameters is around the same as with 0.989525 M. In the case of R_3, the maximum test accuracy is 80.26% which is much greater than that in the case of R_4 which is only 74.65%. This indicates that the maximum test accuracy is more sensitive to changes in rank R_4.

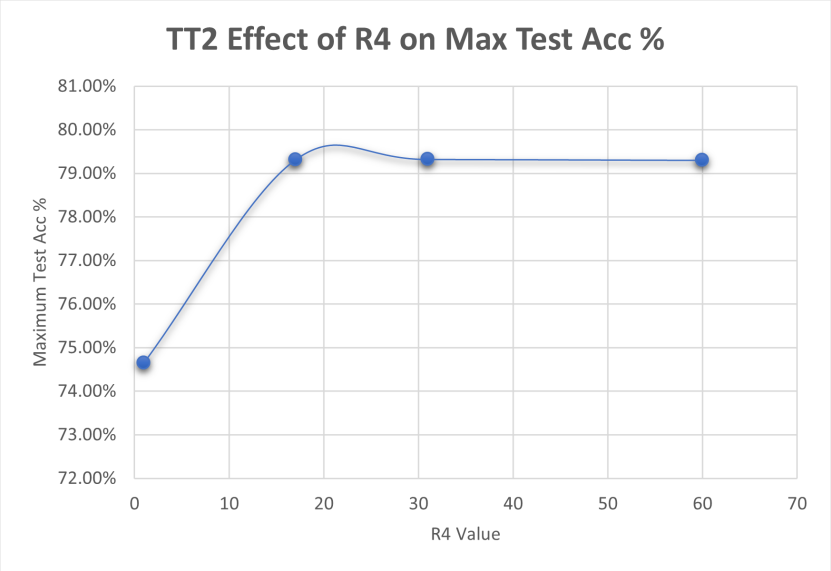


Figure 4-8: Plot between value of rank R_4 and maximum test accuracy % for the TT2 model. The rank R_4 is varied from 60 to 1 while ranks R_1, R_2 and R_3 are fixed at 10, 100 and 570 respectively.

It can be concluded that while ranks R_2 and R_3 do not have a particular effect on the maximum test accuracy, the last rank, which connects the last two TT-cores ($\mathbf{G}_{(4)}$ and $\mathbf{G}_{(5)}$) has an effect on the maximum test accuracy. The test accuracy is more sensitive to the changes in rank R_4 in the TT2 model. This also suggests that in order to achieve better test accuracy, it is important to make sure that the value of rank R_4 is not below a certain value. Here, the rank R_4 has a maximum possible value of 60, and is connected to the core $\mathbf{G}_{(4)}$, to which rank R_3 which has the highest maximum possible rank of 1400 is linked.

4-7 TT3 model

The TT3 model has the following embedding shape (2,3,5,5,5,7) $\times$ (2,2,2,2,2,2,2) with seven factors, all of them arranged in ascending order. In the TT3 model, there are six TT-ranks R_1, R_2, R_3, R_4 R_5 and R_6 which are hyperparameters of the model. The TT3 model can be visualized using the tensor diagram given in figure 4-9. In the TT3 model, eight different compression rates are obtained by selecting different combination of all the TT-ranks. The maximum possible values for each rank of the TT3 model are calculated as:

$$R_1 \leq 4, R_2 \leq 24, R_3 \leq 240, R_4 \leq 1400, R_5 \leq 140, R_6 \leq 14.$$

Similar to the previous TT models, first the rank with highest maximum possible value R_4 is varied to get a compression rate close to 30% while all other ranks are kept fixed. At a compression rate of 30.97%, the rank R_4 is 550. The compression rate is calculated with respect to the full model parameters (3.332225 M).
After most of the possible combinations of TT-ranks for which different compression rates

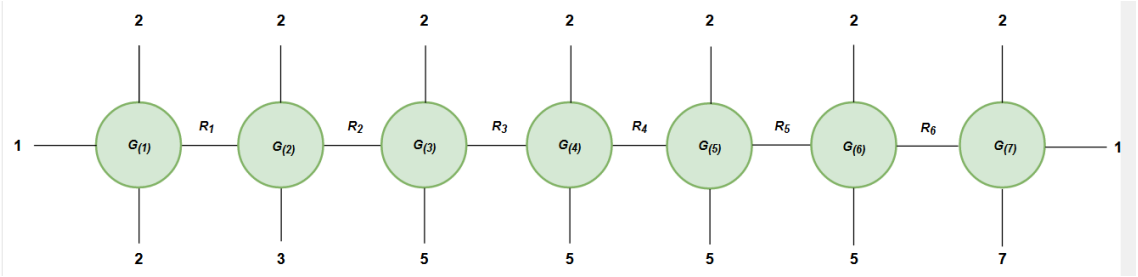


Figure 4-9: Tensor diagram for TT3 Model. The TT-ranks R_1, R_2, R_3, R_4 R_5 and R_6 are hyper-parameters that link each TT-matrix core $\mathbf{G}_{(k)}$ ($k = 1$ to 7) to the next core.

TT3 Ranks	Parameters	Compression %	Max Acc %
Full Model	3.332225 M	-	78.04%
(4,24,240,550,140,14)	2.300213 M	30.97%	79.26%
(4,24,240,550,84,14)	1.984373 M	40.44%	79.74%
(4,24,240,375,140,14)	1.635213 M	50.92%	78.99%
(4,10,68,550,140,14)	1.303077 M	60.89%	80.00%
(4,24,70,550,80,14)	0.986013 M	70.40%	79.28%
(4,10,240,130,140,14)	0.670277 M	79.88%	80.03%
(4,10,240,40,140,14)	0.328277M	90.14%	80.02%
(4,8,8,8,10,8)	0.135425 M	95.93%	73.09%

Table 4-7: Results obtained for the TT3 model. The TT3 ranks are reported as (R_1,R_2,R_3,R_4,R_5,R_6), 'Max Acc' is the maximum accuracy % that is achieved in 10 epochs by the LSTM model. The compression rates are defined with respect to the full model parameters (3.332225 M).

are obtained, the final selected set of TT-ranks for the TT3 model which had the best test accuracy is chosen and the results are reported.

4-7-1 TT3 model results

The final combination of TT-ranks chosen for various compression rates along with the number of parameters (M) and maximum test accuracy (%) for the TT3 model are given in table 4-7. The combination of (R_1,R_2,R_3,R_4,R_5,R_6) as (4,24,240,550,140,14) gives a compression rate of 30.97%. Keeping this as a reference, the TT-ranks are varied for this model. It can be observed from table 4-7 that even at a compression rate of 90.14% with only 0.324525 M parameters, the TT3 model achieves a maximum test accuracy of 80.02% compared to 78.04% maximum test accuracy of the full model. Only at 95.93% compression, the test accuracy drops drastically from 79.53% to 73.09%.

All the other combinations of TT-ranks considered for the TT3 model along with the maximum test accuracy in each case are reported in section B-3 of the appendix.

4-7-2 Effect of TT-ranks in TT3 model

The effects of each rank are investigated by looking at the change in maximum test accuracy with respect to the change in each TT-rank of the model. This is done similarly to that in the case of the TT1 and TT2 models. Tables 4-8a and 4-8b show the change in maximum test accuracy when R_3 and R_5 are varied. Table 4-9 shows the change in maximum test accuracy when R_4 is varied.

R3 Rank	Max Acc%
240	79.26%
190	79.04%
130	78.66%
68	79.22%
8	79.49%
1	79.26%

(a) The rank R_3 is varied from 240 to 1, when R_1, R_2, R_4, R_5 and R_6 are fixed at 4, 24, 550, 140 and 14 respectively.

R5 Rank	Max Acc%
140	79.26%
100	79.68%
84	79.74%
68	79.45%
28	79.10%
1	71.22%

(b) The rank R_5 is varied from 140 to 1, when ranks R_1, R_2, R_3, R_4 and R_6 are fixed at 4, 24, 240, 550 and 14 respectively.

Table 4-8: Effects of ranks R_3 and R_5 on the maximum test accuracy 'Max Acc%' of the TT3 model.

The plot that shows change in rank R_3 with respect to maximum test accuracy is given in figure 4-10. The plot that shows change in rank R_4 with respect to maximum test accuracy is given in figure 4-11 and the plot that shows change in rank R_5 with respect to maximum test accuracy is given in figure 4-12. A plot for change in rank R_1, R_2, and R_6 with respect to maximum test accuracy is not provided because the maximum test accuracy was not particularly sensitive to changes in any of these TT-ranks. It is important to note that when, R_3 has the lowest value of 1, the number of parameters in the TT3 model is 0.928353 M. The number of parameters when R_4 has the lowest value of 1 is 0.214013 M. Similarly, when R_5 has the lowest value of 1, the number of parameters is 1.516253 M.

From the plots, it can be seen that the maximum test accuracy % is not particularly sensitive

R4 Rank	Max Acc%
550	79.26%
465	78.86%
375	78.99%
290	78.70%
205	79.66%
120	80.51%
30	79.42%
1	71.84%

Table 4-9: Effects of ranks R_4 on the maximum test accuracy 'Max Acc %' of the TT3 model. The rank R_4 is changed from 550 to 1, ranks R_1, R_2, R_3, R_5 and R_6 are fixed at 4, 24, 240, 140 and 14 respectively.

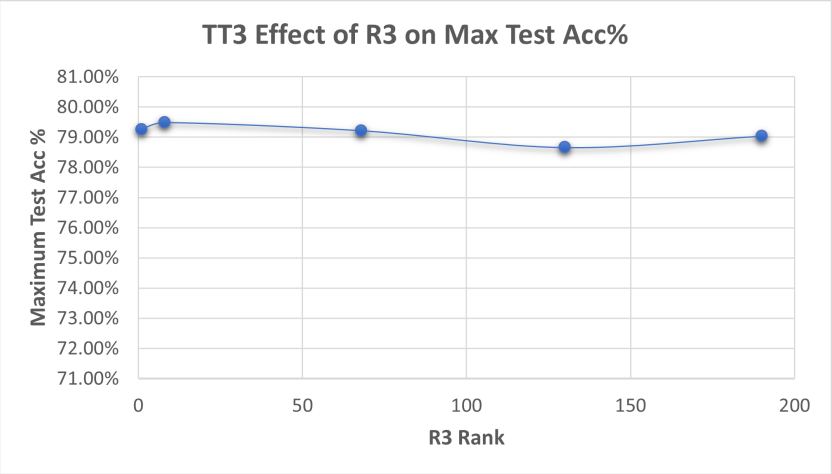


Figure 4-10: Plot between value of rank R_3 and maximum test accuracy % for the TT3 model. The rank R_3 is varied from 240 to 1 while ranks R_1, R_2, R_4, R_5 and R_6 are fixed at 4, 24, 550, 140 and 14 respectively.

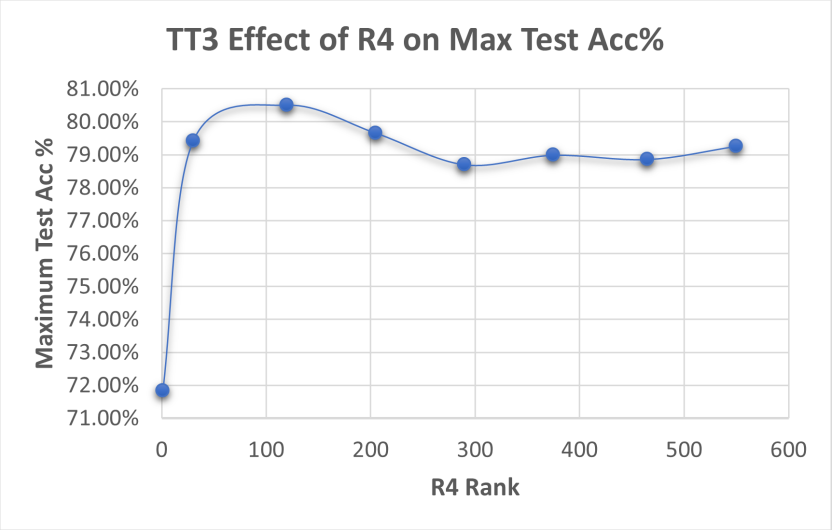


Figure 4-11: Plot between value of rank R_4 and maximum test accuracy % for the TT3 model. The rank R_4 is varied from 550 to 1 while ranks R_1, R_2, R_3, R_5 and R_6 are fixed at 4, 24, 240, 140 and 14 respectively.

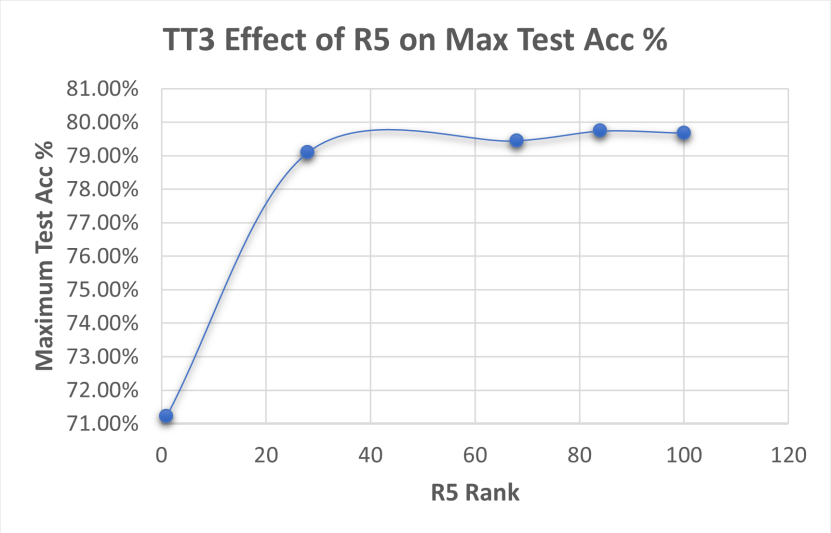


Figure 4-12: Plot between value of rank R_5 and maximum test accuracy % for the TT3 model. The rank R_5 is varied from 140 to 1 while ranks R_1, R_2, R_3, R_4 and R_6 are fixed at 4, 24, 240, 550 and 14 respectively.

to R_3 and R_4. In figure 4-10, it can be seen that maximum accuracy fluctuates between a range of 78.66% and 79.49% in the case of R_3. In figure 4-11, it can be seen that maximum test accuracy fluctuates between a range of 71.22% and 79.74% in the case of R_5. When R_4 is of value 1, the accuracy drops to 71.22% because the number of parameters is only 0.214013 M.

An interesting observation can be inferred from figure 4-12. Here, when the value of rank R_5 drops to 1, the maximum test accuracy drops to 71.22%. while the number of parameters is 1.51623 M parameters. In the case of R_4 when the rank is 290, the maximum test accuracy is 78.70%, while the number of parameters are only 1.3122113 M. The number of parameters in this case is less than the number of parameters when R_5 has a value of 1. This indicates that the maximum test accuracy in the TT3 model is sensitive to changes in R_5.

It can be concluded that while ranks R_3 and R_4 do not have a particular effect on the maximum test accuracy, rank R_5 which connects the two TT-cores ($\mathbf{G}_{(5)}$ and $\mathbf{G}_{(6)}$) has an effect on the maximum test accuracy. The test accuracy is more sensitive to the changes in rank R_5 in the TT2 model. This also suggests that in order to achieve better test accuracy, it is important to make sure that the value of rank R_5 is not below a certain value. Here, the rank R_5 has a maximum possible value of 140, and is connected to the core $\mathbf{G}_{(5)}$, to which rank R_4 which has the highest maximum possible rank of 1400 is linked.

Looking at the investigation that was conducted, it can be seen that in the case of TT1 model, the TT-ranks do not have a particular effect on the maximum test accuracy. However, in the TT2 and TT3 models, the rank connected to the core, which has the highest maximum possible rank, had an interesting effect on the maximum test accuracy. In the case of TT2 model, maximum test accuracy is more sensitive to changes in rank R_4 connected to the TT-cores $\mathbf{G}_{(4)}$ and $\mathbf{G}_{(5)}$ of the model. The TT-core $\mathbf{G}_{(4)}$ is in turn linked to rank R_3 which

has the maximum possible rank of 1400.
In the case of TT3 model, maximum test accuracy is more sensitive to changes in rank R_5 connected to cores $\mathbf{G}_{(5)}$ and $\mathbf{G}_{(6)}$. The TT-core $\mathbf{G}_{(5)}$ is linked to rank R_4 which has the highest maximum possible rank of 1400. This means that the performance of the LSTM model is also affected by the TT-core, which has the maximum possible rank linked to it.
The compression rates in all the three TT models are varied linearly with respect to the change in TT-ranks. The compression rate can be increased by reducing the value of each TT-rank.

It can thus be conjectured that the rank which is on the right end of the TT-core, which is linked to the rank which has the highest maximum possible rank, effects the maximum test accuracy % in the TT models. So, one form of regularization strategy to make sure that the performance is optimum is to make sure that this particular rank does not have a value below a certain threshold.

4-8 Comparison of TT1, TT2 and TT3 models

To understand the effect of the factorizations and if having a greater number of factors has an advantage for the same compression rates, the maximum test accuracy obtained for each compression rate for all the three TT models will be compared. Additionally, the evolution of the loss value in all the three models for different compression rates will be analysed.

The final selected combinations from all the 3 models are used for this comparison, which were reported in the previous sections. The plot that shows the evolution of the maximum test accuracy with respect to the compression rate is given in figure 4-13. From figure 4-13, it can be seen that the change in the maximum test accuracy with respect to the compression rates is similar for all the three TT models when the compression rates range from 30% to 90%. However, after the number of parameters in the TT models is reduced to get a compression rate of close to 95%, there is a drastic drop in the maximum test accuracy. The region where the number of parameters is close to almost 0.13 M seems to be sensitive. This is where the effect of factorization appears to be more clear. At around 95% compression rate, the TT3 model with the highest number of factors (seven) has a test accuracy of 73.09% while the TT1 model only has a test accuracy of 69.50%. However, the TT2 model has a slightly lower accuracy of 68.05% than that of the TT1 model.

4-8-1 Evolution of loss in TT1, TT2 and TT3 models

In order to give an explanation as to how the TT3 model with a greater number of factors performs better, the evolution of loss can be analysed. During the training of the LSTM model with the training dataset, the loss values were recorded for each epoch. The loss values are also recorded for when the LSTM model is run with the test dataset as the input. A plot is generated to understand the evolution of the loss value with respect to the number of epochs for both the training dataset and the test dataset.

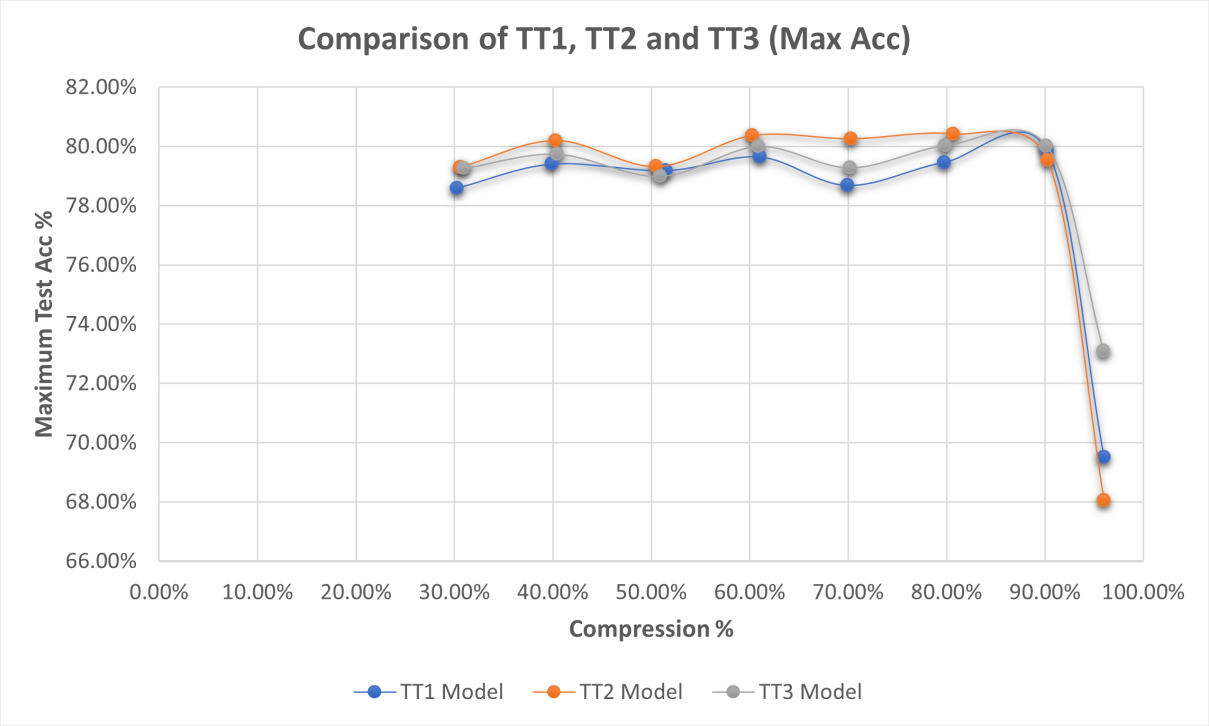


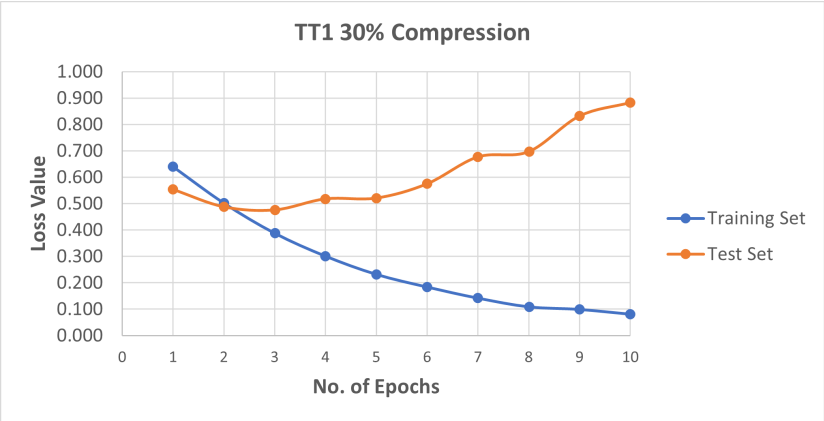
Figure 4-13: Plot that shows the evolution of maximum test accuracy in each TT model with respect to eight different compression rates, ranging from 30% to 96%. The trend for TT1 model is depicted by the blue line, the trend for TT2 model is depicted by the orange line and the trend for the TT3 model is depicted by the grey line.

These plots are generated for all three TT-models for the compression rates close to 30%, 50%, 70% and 90%. The plots for the evolution of loss value in the TT1, TT2 and TT3 models for 30% compression rate are given in figures 4-14a, 4-14b and 4-14c respectively.

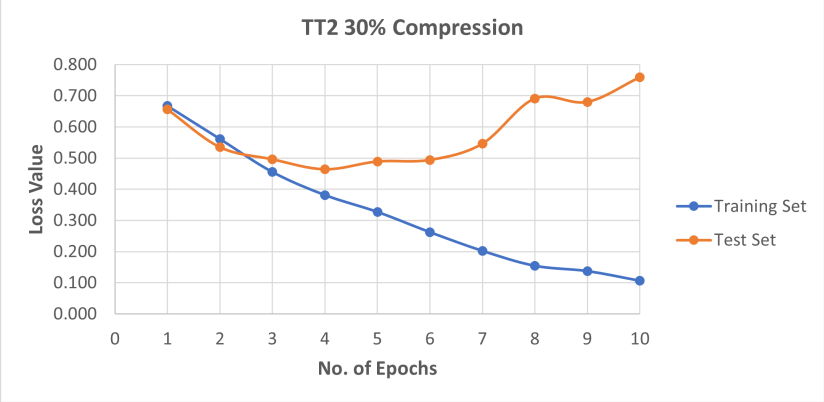
From the figures 4-14a, 4-14b and 4-14c given for the evolution of loss at 30% compression rates, it can clearly be seen that the training loss decreases with increase in number of epochs for all the three TT models. However, the test loss does not reduce and increases with an increase in the number of epochs for both the TT1 and TT2 models. The models struggle to follow the trend of loss that is achieved during the training set.
In the TT3 model, the test loss decreases with an increase in the number of epochs to some extent. The trends for both test loss and the training loss for the TT3 model have relatively similar trends.
The plots for the evolution of loss for all the three TT models at a compression rate 90% are given in figures 4-15a, 4-15b and 4-15c. Even when the compression rate is 90%, in the case of the TT3 model, the test loss and the training loss have a similar trend. The test loss follows the training loss more closely. The test loss and the training loss both decrease as the number of epochs increase.
In case of the TT2 model at 90% compression, the test loss increases as the number of epochs increase. The evolution of test loss is relatively closer to that of the training loss to some extent than that in the case of the TT1 model but the TT3 model outperforms both the TT1 and TT2 models. The plots for the evolution of loss when the compression rates are close to 50% and 70% are given in section B-4 of the appendix. Even at 50% and 70% compression rates, the evolution of test loss is the best (follows the trend of the training loss) for the TT3 model.

After looking at the evolution of loss in all the three TT models, the observations made suggest that the TT3 model has good generalization capability. Work on exploring the generalization capability for a particular NN architecture has been done done before. The concept of generalizing capability of a NN was explored by Hochreiter and Schmidhuber [17], where an algorithm was presented that searches for a minimum of the error function during training of the NN to increase the generalizing capability of the model. Li et al. [23] and Arora et al. [3] also studied how generalization can be measured in deep learning models.

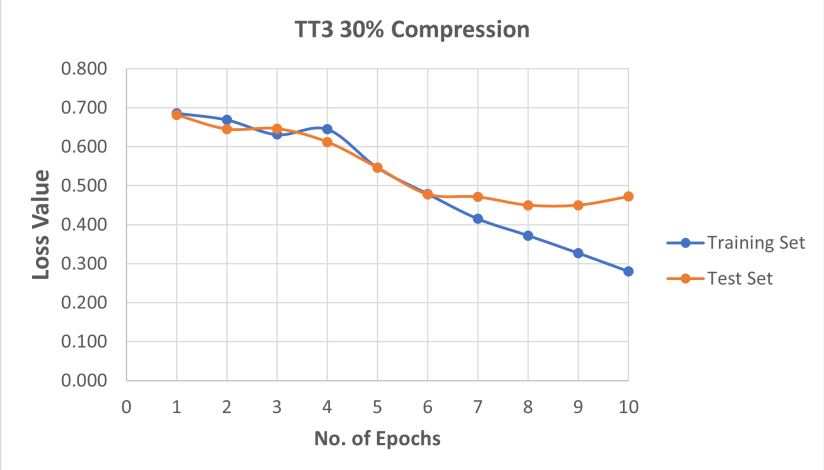
The generalizing capability of a model is used to describe the ability of a particular NN based LM model to predict an output when unseen data is given as input. This means that any model that has good generalizing capability will be able to achieve higher performance as the model will be able to map the input data to the output data with the parameters that it has. Empirically, in the case of the TT3 model, it can be concluded that the model has good generalization capability by observing the plots for the evolution of loss.
In the case of the TT3 model, the test loss is closer to the training loss which means that the TT3 model is able to predict the output more accurately compared to the TT1 and TT2 models.
This also empirically explains why the TT3 model has a higher test accuracy of 73.09% at a compression rate close to 95% than the TT1 (69.50%) and TT2 model (68.05%).
The generalization capability of the TT3 model can also be linked to the TT-ranks of the model. Since, there are more TT- ranks to tune and choose from in the TT3 model, this makes the model more wide (similar to having more number of nodes in the layer of a NN)



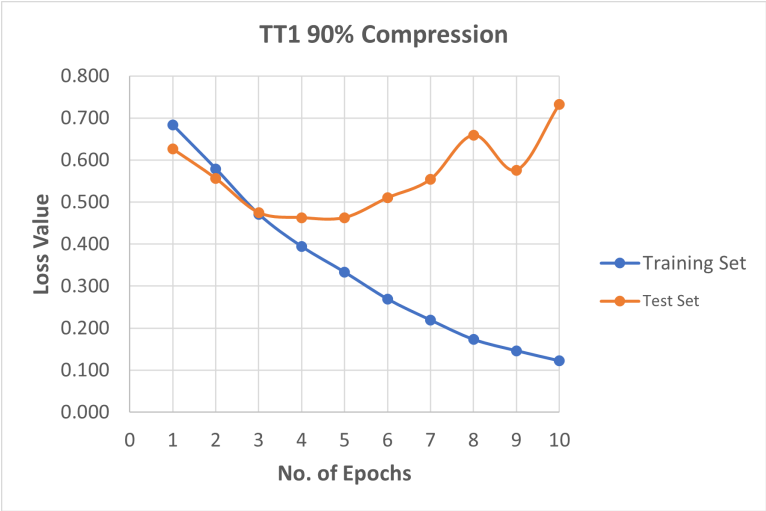
(a) Evolution of loss in the case of both training and test datasets in the TT1 model when the compression rate is close to 30%.



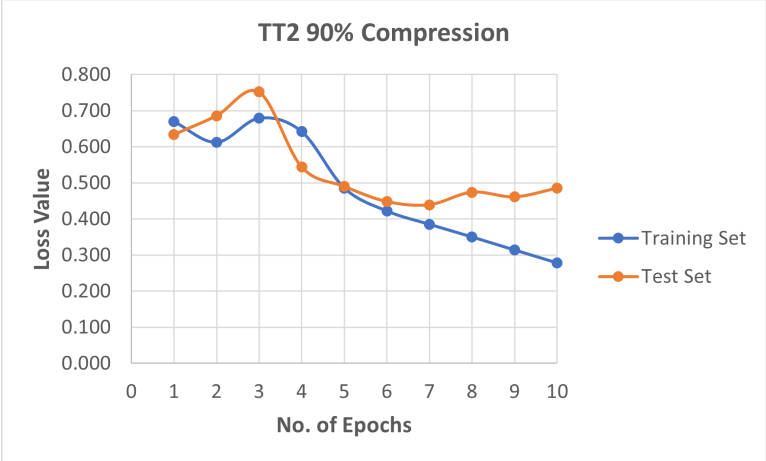
(b) Evolution of loss in the case of both training and test datasets in the TT2 model when the compression rate is close to 30%.



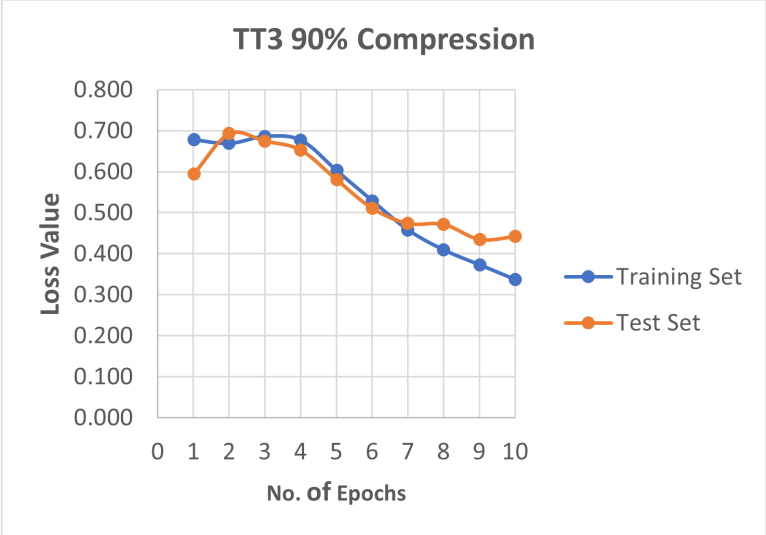
(c) Evolution of loss in the case of both training and test datasets in the TT3 model when the compression rate is close to 30%.



(a) Evolution of loss in the case of both training and test datasets in the TT1 model when the compression rate is close to 90%.



(b) Evolution of loss in the case of both training and test datasets in the TT2 model when the compression rate is close to 90%.



(c) Evolution of loss in the case of both training and test datasets in the TT3 model when the compression rate is close to 90%.

and hence can help the TT3 model to retain/improve the generalization capability of the original full model. To arrive at the exact mathematical explanation as to why TT3 has more generalization capability is difficult and this can be a prospective path for future work.

By looking at results obtained during the investigation into the first research question, it can be concluded that having a greater number of factors in the factorization does indeed have an advantage, and improves the test accuracy of the LSTM model used for this work. The TT3 model with seven factors, achieves a maximum test accuracy of 73.09% on the IMDB dataset while the number of parameters is only 0.135425 M. The number of parameters in this case is reduced by 24.605 times compared to the number of parameters in the full model, which had 3.332225 M parameters.
The TT3 model also achieves a test accuracy of 80.02% when the number of parameters is only 0.328277 M (90.14% compression rate). Here, the number of parameters is reduced by 10.15 times compared to the parameters in the model. With almost 10.15 times fewer parameters, the TT3 model achieves a higher test accuracy of 80.02% compared to that of 78.04% of the full model.
Based on the evolution of loss observed in the all the three TT models, it can be empirically explained that the reason for TT3 having more number of factors can be attributed to the generalization capability of the model.

It can be inferred from the above results that increasing the number of factors in the factorization of the original embedding matrix in the LSTM model indeed has an advantage and helps to improve the performance of the LSTM model for the application of NLP, sentiment analysis in particular. Hence, it can be concluded that replacing standard embeddings by TT-embeddings in the LSTM model, improves the test accuracy of the model. Further, the by choosing more number of factors in the embedding matrix, better performance is obtained. The test accuracy of the model is sensitive to changes in TT-ranks that are linked to the TT-core which has TT-rank with the highest maximum possible value linked to it (specifically R_4 in the TT2 model and R_5 in the TT3 model).

Chapter 5

Order of factorization and its effects

In this chapter, the second research question will be answered. It is important to note that in TT1, TT2 and TT3 models, the factors of the embedding matrix were always arranged in ascending order with respect to the numerical values of the elements per TT-core. These factors also determine the maximum possible value that each TT-rank can achieve in the particular model. Thus, by changing the order in which the factors of each factorization are placed, the TT-ranks also change. This alters the entire structure of the compressed LSTM model. In the work done by Hrinchuk et al. [19], the effect of re-arranging the factors for the same factors in a TT model has not been explored. To my knowledge, this was not addressed netiher theoretically nor empirically in any other work in the field of tensor decompositions in NLP.

5-1 Methodology - research question 2

In the same LSTM model used to perform NLP, when TT-embeddings are used, how does the order of the factors in each factorization effect the compression rate and the test accuracy of the model?

To investigate the effects of re-arranging the factors of the TT-matrix, we consider the same factors that are present in the TT1, TT2 and TT3 models.
For each TT model, two additional factorizations will be investigated. The factors of both vocabulary size I and the N- dimensional vector space J can be re-arranged and placed differently.

Different combinations of arranging these factors exist but only the order of factors, where the maximum possible value for each TT-ranks is different are considered. The maximum possible TT-ranks are calculated using the same formula mentioned in section 3-2-3. For example, in the original TT1 model, the maximum possible values for rank R_1 and R_2 are 100 and 280. So, we only place the factors in such a way that the ranks of the other type of factorization

Model	Embedding Shape
TT1	(25,30,35) × (4,4,8)
TT1-1	(35,30,25) × (4,8,4)
TT1-2	(25,35,30) × (4,8,4)
TT2	(5,5,7,10,15)) × (2,2,2,4,4)
TT2-1	(5,7,10,15,5) × (2,2,4,4,2)
TT2-2	(5,7,15,10,5) × (2,4,4,2,2)
TT3	(2,3,5,5,5,5,7) × (2,2,2,2,2,2,2)
TT3-1	(5,5,3,5,5,2,7) × (2,2,2,2,2,2,2)
TT3-2	(5,2,5,5,5,3,7) × (2,2,2,2,2,2,2)

Table 5-1: Different factorizations considered to investigate the effect of re-arranging the factors in the TT-matrix. Two additional factorizations for each TT model are considered.

have R_1 and R_2 different from 100 and 280. This is done for all the 3 TT-models.
After the order of factors is decided, the effect of TT-ranks on the maximum test accuracy and the compression rate is investigated. The same procedure that was followed to answer research question 1 is followed to answer research question 2. The TT-ranks for each model is varied to achieve eight different compression rates between 30% and 95%. The maximum test accuracy along with the number of parameters will also be reported.
Different combinations for the TT-ranks will be considered and the combinations which result in the highest test accuracy % for each compression rate will be chosen.

Table 5-1 provides details about the models and the placements of the factors that will be investigated to understand the effects of re-arranging the factors. It can be seen from table 5-1 that the factors are the same for each model, but the order in which those factors are arranged is not in ascending order anymore.

5-2 TT1-1 and TT1-2 models

The TT1-1 has the same factors that the TT1 model has, but the factors are arranged differently. In the case of the TT1-1 model, the ranks of the model are calculated as:

$$R_1 \leq 140, R_2 \leq 100.$$

For the TT1-2 model, the ranks of the model are calculated as:

$$R_1 \leq 100, R_2 \leq 120.$$

The final combination of TT-ranks chosen for various compression rates along with the number of parameters (M) and maximum test accuracy (%) for the TT1-1 model are given in table 5-2. It can be observed from table 5-2 that the combination of R_1 and R_2 as 90 and 100 respectively gives a compression rate of 30.53%. Keeping that as a reference, the TT-ranks are varied.
From table 5-2, it can be observed that at a compression rate of 90.18% with only 0.327105 M,

TT1-1 Ranks	Parameters	Compression %	Max Acc %
(90,100)	2.314825 M	30.53%	78.47%
(76,100)	1.976865 M	40.67%	79.30%
(62,100)	1.638905 M	50.81%	79.30%
(70,70)	1.325025 M	60.23%	79.33%
(35,100)	0.987125 M	70.37%	79.78%
(90,24)	0.665625 M	80.02%	78.77%
(28,28)	0.327105 M	90.18%	79.47%
(1,1)	0.132705 M	96.01%	64.98%

Table 5-2: Results obtained for the TT1-1 model. The TT1-1 ranks are reported as (R_1,R_2), 'Max Acc' is the maximum accuracy % that is achieved in 10 epochs by the LSTM model. The compression rates are defined with respect to the full model parameters (3.332225 M).

TT1-2 Ranks	Parameters	Compression %	Max Acc %
(100,78)	2.335585 M	29.90%	79.16%
(84,78)	1.984545 M	40.44%	78.30%
(79,68)	1.652445 M	50.41%	77.83%
(54,78)	1.326345 M	60.19%	79.54%
(38,78)	0.975305 M	70.73%	79.24%
(100,18	0.648385 M	80.54%	78.88%
(24,26)	0.312465 M	90.62%	79.45%
(1,1)	0.132725 M	96.01%	62.17%

Table 5-3: Results obtained for the TT1-2 model. The TT1-2 ranks are reported as (R_1,R_2), 'Max Acc' is the maximum accuracy % that is achieved in 10 epochs by the LSTM model. The compression rates are defined with respect to the full model parameters (3.332225 M).

the TT1-1 model achieves a maximum test accuracy of 79.47%. However, the test accuracy drops drastically to 64.98% when the compression rate is 96.01%.
All the other combinations of TT-ranks considered for the TT1-1 model, along with the maximum test accuracy in each case, are reported in section C-1 of the appendix.

The final combination of TT-ranks chosen for various compression rates along with the number of parameters (M) and maximum test accuracy (%) for the TT1-2 model are given in table 5-3. It can be observed from table 5-3 that the combination of R_1 and R_2 as 100 and 78 respectively gives a compression rate of 29.90%. Keeping that as a reference, the TT-ranks are varied.
From table 5-3, it can be observed that at a compression rate of 90.62% with only 0.312465 M, the TT1-2 model achieves a maximum test accuracy of 79.45%. However, the test accuracy drops drastically to 62.17% when the compression rate is 96.01%.
All the other combinations of TT-ranks considered for the TT1-2 model along with the maximum test accuracy in each case are reported in section C-2 of the appendix.

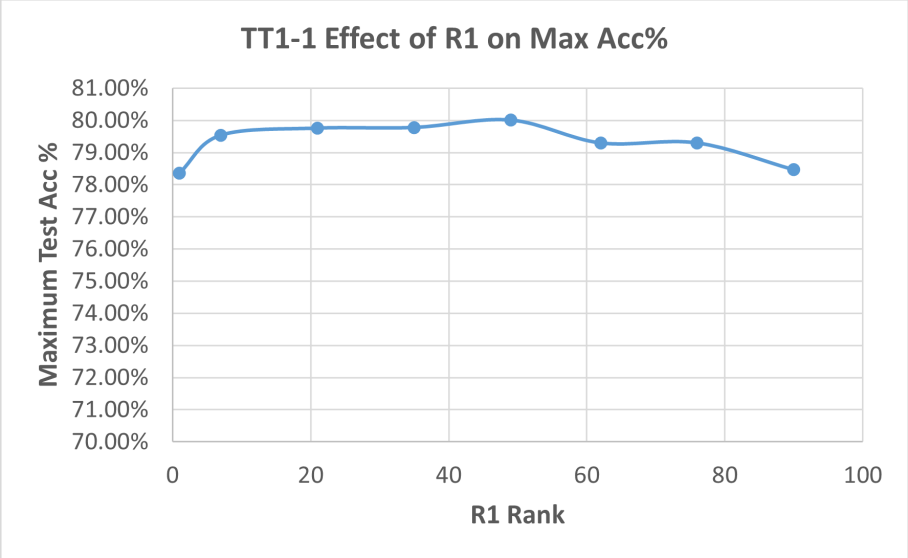


Figure 5-1: Plot between value of rank R_1 and maximum test accuracy % for the TT1-1 model. The rank R_1 is varied from 90 to 1 while rank R_2 is always 100.

5-2-1 Effect of TT-ranks in TT1-1 model

The effect of each rank is investigated by looking at the change in maximum test accuracy with respect to the change in each TT-rank of the model. This is done similarly to that in the case of all the previous TT models. Table C-4 in section C-1 of the appendix shows the change in maximum test accuracy when R_1 and R_2 are varied. The plot that depicts change in R_1 with respect to the maximum test accuracy is given in figure 5-1. In this case, the rank R_1 is varied while R_2 is fixed at 100. The plot that depicts change in rank R_2 with respect to the maximum test accuracy is given in figure 5-2.

When $R_1 = 1$ and $R_2 = 100$, the number of parameters in the model are 0.166365 M. When the $R_2 = 1$ and $R_1 = 90$, the number of parameters in the model are 0.166525 M.
From the plots given in figures 5-1 and 5-2, it can be observed that when rank $R_1 = 1$, the maximum test accuracy is 78.36% and when $R_2 = 1$, the maximum test accuracy is 70.98%. This is an interesting observation because, even when the number of parameters is similar in both cases, the accuracy drops dramatically when the rank R_2 has the lowest value of 1. The TT1-1 model is thus more sensitive to changes in the rank R_2. The rank R_2 in the TT1-1 model connects the TT-cores $\mathbf{G}_{(2)}$ and $\mathbf{G}_{(3)}$.

5-2-2 Effect of TT-ranks in TT1-2 model

Table C-8 in section C-2 of the appendix shows the change in maximum test accuracy when R_1 and R_2 are varied in the TT1-2 model. The plot that shows change of rank R_1 with respect to the maximum test accuracy is given in figure 5-3. In this case, the rank R_1 is varied while R_2 is fixed at 78. The plot that shows change of rank R_2 with respect to the maximum test accuracy is given in figure 5-4.

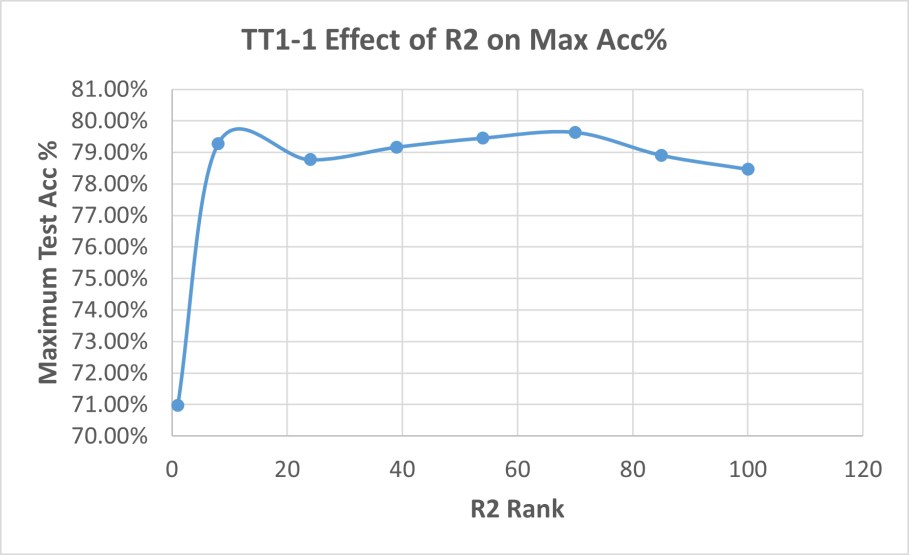


Figure 5-2: Plot between value of rank R_2 and maximum test accuracy % for the TT1-1 model. The rank R_2 is varied from 100 to 1 while rank R_1 is always 90.

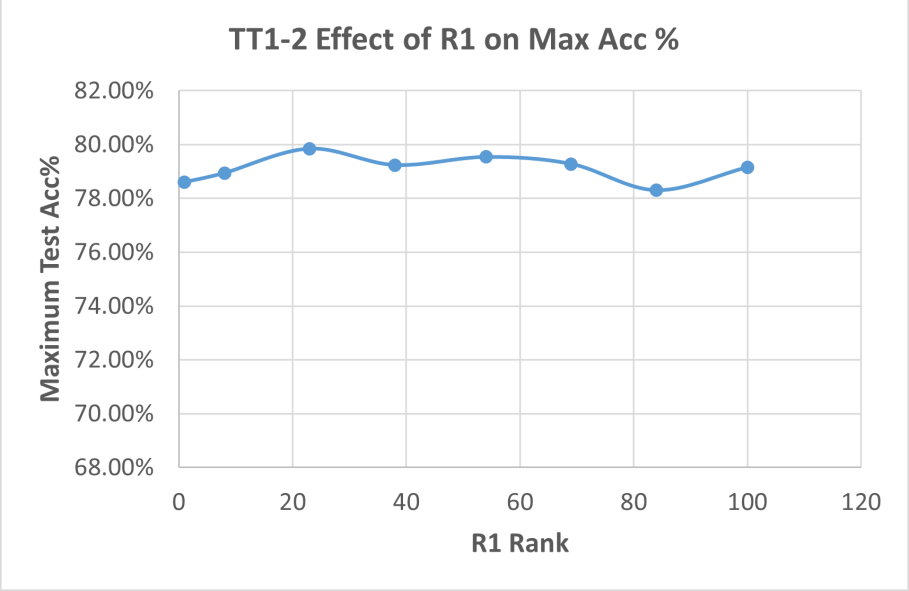


Figure 5-3: Plot between value of rank R_1 and maximum test accuracy % for the TT1-2 model. The rank R_1 is varied from 100 to 1 while rank R_2 is always 78.

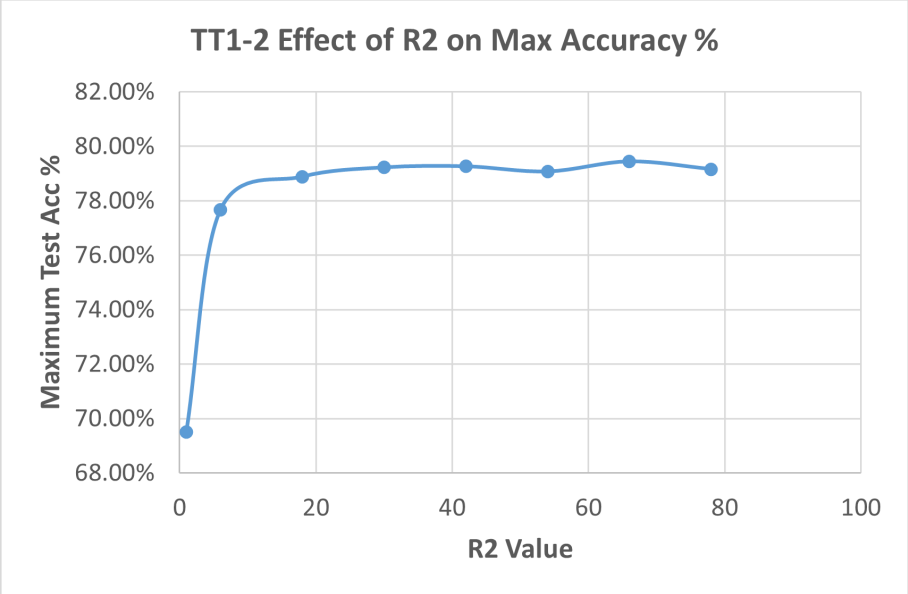


Figure 5-4: Plot between value of rank R_2 and maximum test accuracy % for the TT1-2 model. The rank R_2 is varied from 78 to 1 while rank R_1 is always 100.

When $R_1 = 1$ and $R_2 = 78$, the number of parameters in the model is 0.162525 M. When $R_2 = 1$ and $R_1 = 100$, the number of parameters in the model is 0.170345 M.
From the plots given in figures 5-3 and 5-4, it can be observed that when rank $R_1 = 1$, the maximum test accuracy is 78.60% and when $R_2 = 1$, the maximum test accuracy is only 69.51%. Despite the number of parameters being slightly higher with 0.170345 M, when rank R_2 has the lowest value of 1, the test accuracy drops drastically. This indicates that the TT1-2 model's test accuracy is more sensitive to changes in rank R_2. The same observation was made during the investigation of the effects of TT-ranks on the TT1-1 model.
In the TT1-2 model, the rank R_2 links the TT-cores $\mathbf{G}_{(2)}$ and $\mathbf{G}_{(3)}$.

5-3 Comparison TT1, TT1-1 and TT1-2 models

The results obtained for the TT1-1 and TT1-2 models are compared with the TT1 model to understand if arranging the factors differently has an effect when the number of factors is three. For each model, the maximum test accuracy at each compression rate is plotted.

The final selected combinations from the TT1, TT1-1 and TT1-2 models are used for this comparison, which were reported in the previous sections. The plot that shows the evolution of the maximum test accuracy with respect to the compression rate is given in figure 5-5. From figure 5-5, it can be seen that the change in the maximum test accuracy with respect to the compression rates is similar for all the three models when the compression rates range from 30% to 90%. However, when the compression rate is around 95%, there is a drastic drop in the maximum test accuracy. At 96.01%, the TT1 model achieves the best performance with a maximum test accuracy of 69.50%. The TT1-1 model achieves a test accuracy of

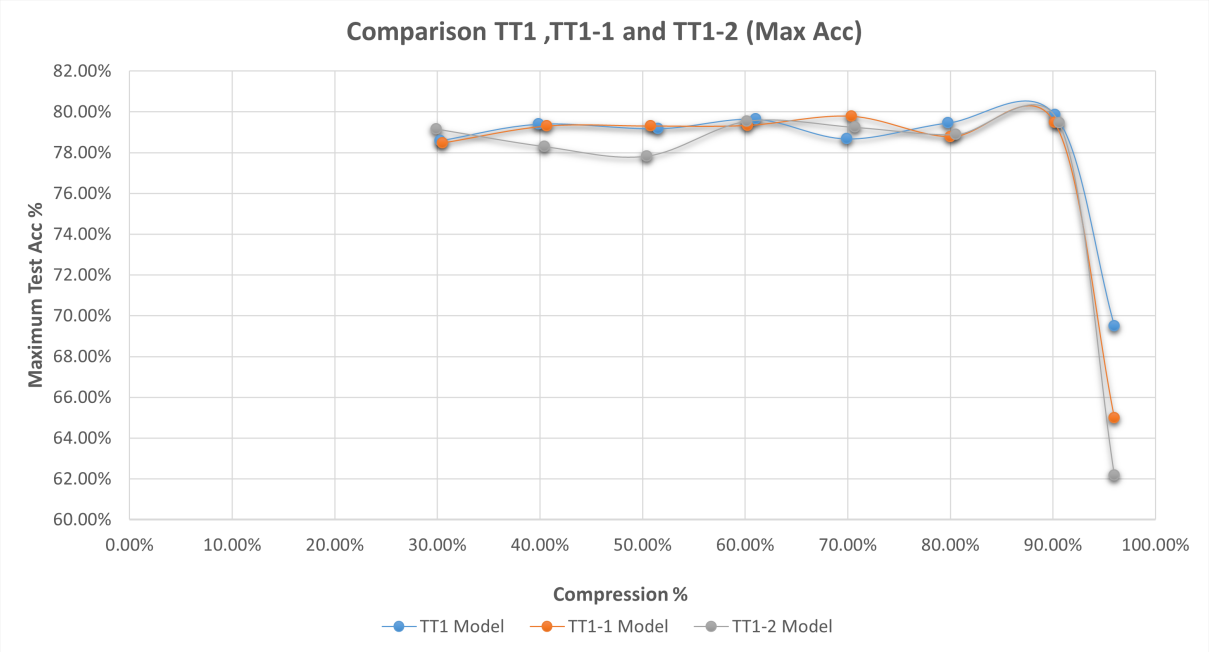


Figure 5-5: Plot that shows the evolution of maximum test accuracy in the TT1, TT1-1 and TT1-2 models with respect to eight different compression rates, ranging from 30% to 96%. The trend for TT1 model is depicted by the blue line, the trend for TT1-1 model is depicted by the orange line and the trend for the TT1-2 model is depicted by the grey line.

64.98%. The TT1-2 model has the lowest test accuracy amongst the three models with only 62.17 %.
It can be concluded that in the case of the TT1 model, changing the order in which the factors are placed does not improve the performance of the LSTM model. However, the effect of TT-ranks in the TT1-1 and TT1-2 seems to be different than that in the case of the TT1 model. In the TT1 model, the maximum test accuracy was not particularly sensitive to changes in either R_1 or R_2.
In the case of the TT1-1 and TT1-2 models, the maximum test accuracy is more sensitive to changes in the rank R_2 which connects the TT-cores $\mathbf{G}_{(2)}$ and $\mathbf{G}_{(3)}$.

5-4 TT2-1 and TT2-2 models

In the case of TT2-1 model, the ranks of the model are calculated as:

$$R_1 \leq 10, R_2 \leq 140, R_3 \leq 600, R_4 \leq 10.$$

For the TT2-2 model, the ranks of the model are calculated as:

$$R_1 \leq 10, R_2 \leq 240, R_3 \leq 200, R_4 \leq 10.$$

The final combination of TT-ranks chosen for various compression rates along with the number of parameters (M) and maximum test accuracy (%) for the TT2-1 model are given in table 5-4.

TT2-1 Ranks	Parameters	Compression %	Max Acc %
(10,140,350,10)	2.322025 M	30.31%	78.02%
(10,116,350,10)	1.982665 M	40.50%	79.92%
(10,92,350,10)	1.643305 M	50.68%	80.14%
(10,68,350,10)	1.303945 M	60.86%	79.54%
(10,100,182,10)	0.983625 M	70.45%	79.34%
(10,90,120,10)	0.649025 M	80.52%	79.24%
(10,60,60,10)	0.320825 M	90.37%	79.39%
(8,1,2,10)	0.133797 M	95.98%	71.94%

Table 5-4: Results obtained for the TT2-1 model. The TT2-1 ranks are reported as (R_1,R_2,R_3,R_4), 'Max Acc' is the maximum accuracy % that is achieved in 10 epochs by the LSTM model. The compression rates are defined with respect to the full model parameters (3.332225 M).

TT2-2 Ranks	Parameters	Compression %	Max Acc %
(10,175,200,10)	2.321425 M	30.22%	78.26%
(10,148,200,10)	1.989865 M	40.28%	79.60%
(10,120,200,10)	1.646025 M	50.60%	79.63%
(10,138,135,10)	1.315865 M	60.51%	78.98%
(10,175,75,10)	0.983925 M	70.47%	80.25%
(10,175,44,10)	0.652225 M	80.42%	79.29%
(10,12,200,10)	0.319785 M	90.40%	79.62%
(10,2,2,10)	0.133625 M	95.98%	72.74%

Table 5-5: Results obtained for the TT2-2 model. The TT2-2 ranks are reported as (R_1,R_2,R_3,R_4), 'Max Acc' is the maximum accuracy % that is achieved in 10 epochs by the LSTM model. The compression rates are defined with respect to the full model parameters (3.332225 M).

It can be observed from table 5-4 that the combination of (R_1,R_2,R_3,R_4) as (10,140,350,10) yields a compression rate of 30.31%. Keeping that as a reference, the TT-ranks are varied. From table 5-4 it can be observed that at a compression rate of 90.37% with only 0.320825 M, the the TT2-1 model achieves a maximum test accuracy of 79.39%. However, the test accuracy drops drastically to 71.94% when the compression rate is 95.98%.All the other combinations of TT-ranks considered for the TT2-1 model, along with the maximum test accuracy in each case, are reported in section C-3 of the appendix.

The final combination of TT-ranks chosen for various compression rates along with the number of parameters (M) and maximum test accuracy (%) for the TT2-2 model are given in table 5-5. It can be observed from table 5-5 that the combination of (R_1,R_2,R_3,R_4) as (10,175,200,10) yields a compression rate of 30.22%. Keeping that as a reference, the TT-ranks are varied. From table 5-5, it can be observed that at a compression rate of 90.40% with only 0.319785 M, the the TT2-2 model achieves a maximum test accuracy of 79.62%. However, the test accuracy drops drastically to 72.74% when the compression rate is 95.98%.
All the other combinations of TT-ranks considered for the TT2-2 model, along with the maximum test accuracy in each case, are reported in section C-4 of the appendix.

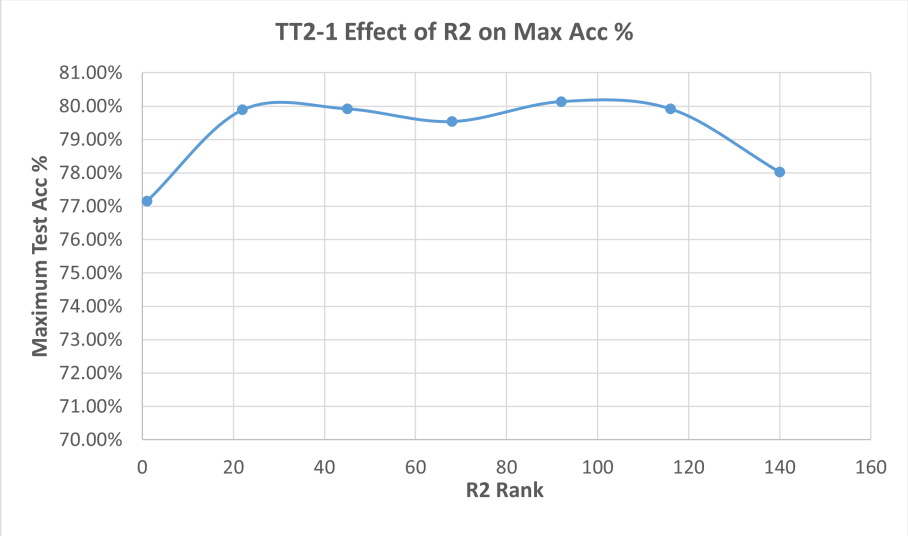


Figure 5-6: Plot between value of rank R_2 and maximum test accuracy % for the TT2-1 model. The rank R_2 is varied from 140 to 1, while R_1, R_3 and R_4 are fixed at 10, 350 and 10 respectively.

5-4-1 Effect of TT-ranks in TT2-1 model

Table C-13 in appendix section C-3 shows the change in maximum test accuracy when R_3 and R_4 are varied. The plot that shows change of rank R_2 with respect to the maximum test accuracy is given in figure 5-6. In this case, the rank R_2 is varied while R_1, R_3 and R_4 are fixed at 10, 350 and 10 respectively. The plot that shows change of rank R_3 with respect to the maximum test accuracy is given in figure 5-7.

The number of parameters when $R_2 = 1$ is 0.356565 M and the number of parameters when $R_3 = 1$ is 0.158225 M. From the plot given in figure 5-6, it can be observed that the maximum test accuracy does not drop drastically when the rank of R_2 reduces to 1. But when R_3 has the lowest value of 1, the maximum test accuracy drops to 70.30%. Even though the number of parameters are less in this case, the drop in the test accuracy to 70.30% suggests that the model is sensitive to changes in R_3. This means that the test accuracy is sensitive to the rank that has the highest possible value ($R_3 \leq 600$). The rank R_3 connects the cores $\mathbf{G}_{(3)}$ and $\mathbf{G}_{(4)}$.

The change in ranks R_1 and R_4 did not have any particular effect on the maximum test accuracy in the TT2-1 model.

5-4-2 Effect of TT-ranks in TT2-2 model

Table C-18 given in section C-4 of the appendix shows the change in maximum test accuracy when R_3 and R_4 are varied in the TT2-2 model. The plot that shows change of rank R_2 with respect to the maximum test accuracy is given in figure 5-8. In this case, the rank R_2 is varied

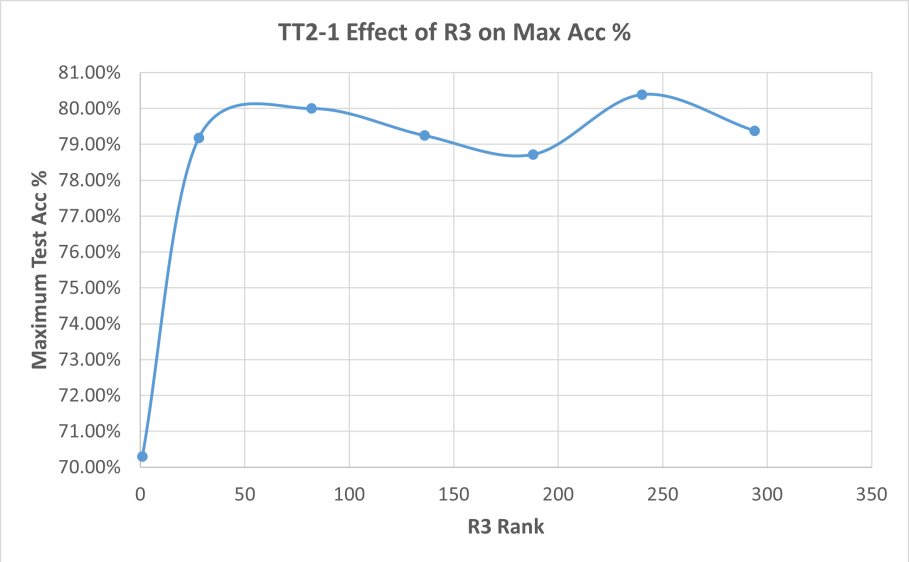


Figure 5-7: Plot between value of rank R_3 and maximum test accuracy % for the TT2-1 model. The rank R_3 is varied from 350 to 1, while R_1, R_2 and R_4 are fixed at 10,140 and 10 respectively.

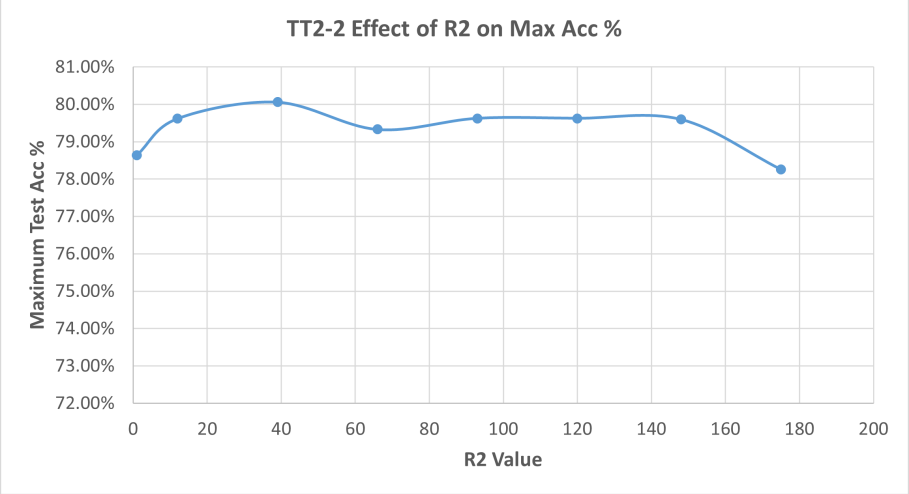


Figure 5-8: Plot between value of rank R_2 and maximum test accuracy % for the TT2-2 model. The rank R_2 is varied from 175 to 1, when R_1, R_3 and R_4 are fixed at 10, 200 and 10 respectively.

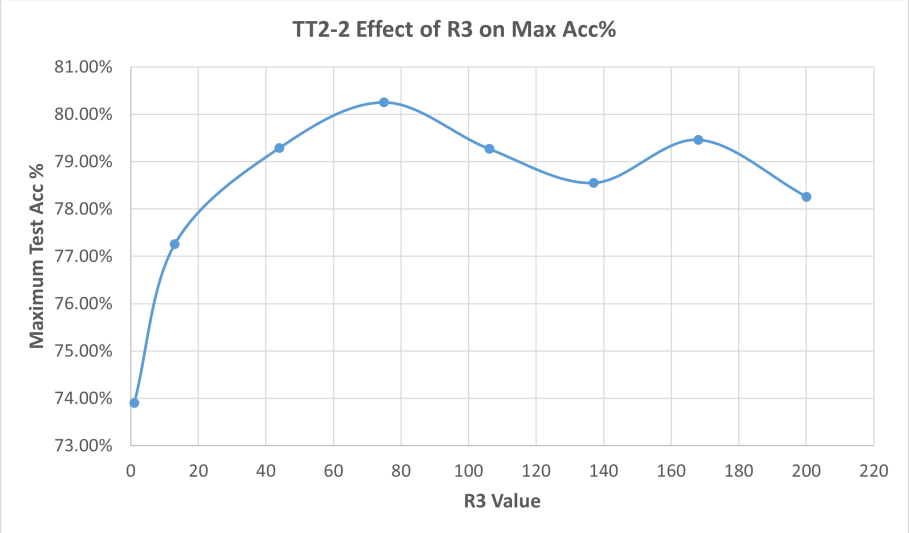


Figure 5-9: Plot between value of rank R_3 and maximum test accuracy % for the TT2-2 model. The rank R_3 is varied from 200 to 1, when R_1, R_2 and R_4 are fixed at 10, 175 and 10 respectively.

while all other TT-ranks are fixed. The plot that shows change of rank R_3 with respect to the maximum test accuracy is given in figure 5-9.

The number of parameters when R_2 is 1 is 0.187405 M. The test accuracy in this case is 78.64% (seen in figure 5-8). When, R_3 is 1, the number of parameters is 0.192125 M. However, the test accuracy is only 73.90% (this is seen in figure 5-9). Despite having more parameters when $R_3 = 1$, the TT2-2 model has lower test accuracy. This indicates that the maximum test accuracy of the TT2-2 model is more sensitive to changes in rank R_3. Hence, in order to make sure that the test accuracy % is better, the value of rank R_3, should not be below a certain value. Additionally, when $R_4 = 1$ and $R_1 = 10$, $R_2 = 175$ and $R_3 = 200$, the maximum test accuracy drops to 75.50% despite the number of parameters in the model being 2.285335 M. This means that the maximum test accuracy is also sensitive to changes in rank R_4.

The TT2-2 model is sensitive to changes in ranks R_3 and R_4. Rank R_3 is linked to the TT-cores $\mathbf{G}_{(3)}$ and $\mathbf{G}_{(4)}$. The TT-core $\mathbf{G}_{(3)}$ is linked to rank R_2 which is has the highest maximum possible rank. Rank R_4 is connected to TT-cores $\mathbf{G}_{(4)}$ and $\mathbf{G}_{(5)}$. In the TT2-2 model, the maximum test accuracy is sensitive to the rank, which is linked to the TT-core, which in-turn is linked to the rank R_2 (highest maximum possible rank). The maximum test accuracy is also affected by changes in R_4 rank. It should be noted that the ranks R_2 and R_3 in the TT2-2 model have maximum values that are very close to each other (240 and 200 respectively).

5-5 Comparison TT2, TT2-1 and TT2-2 models

The results obtained for the TT2-1 and TT2-2 models will be compared with the TT2 model to understand if arranging the factors differently has an effect when the number of factors is

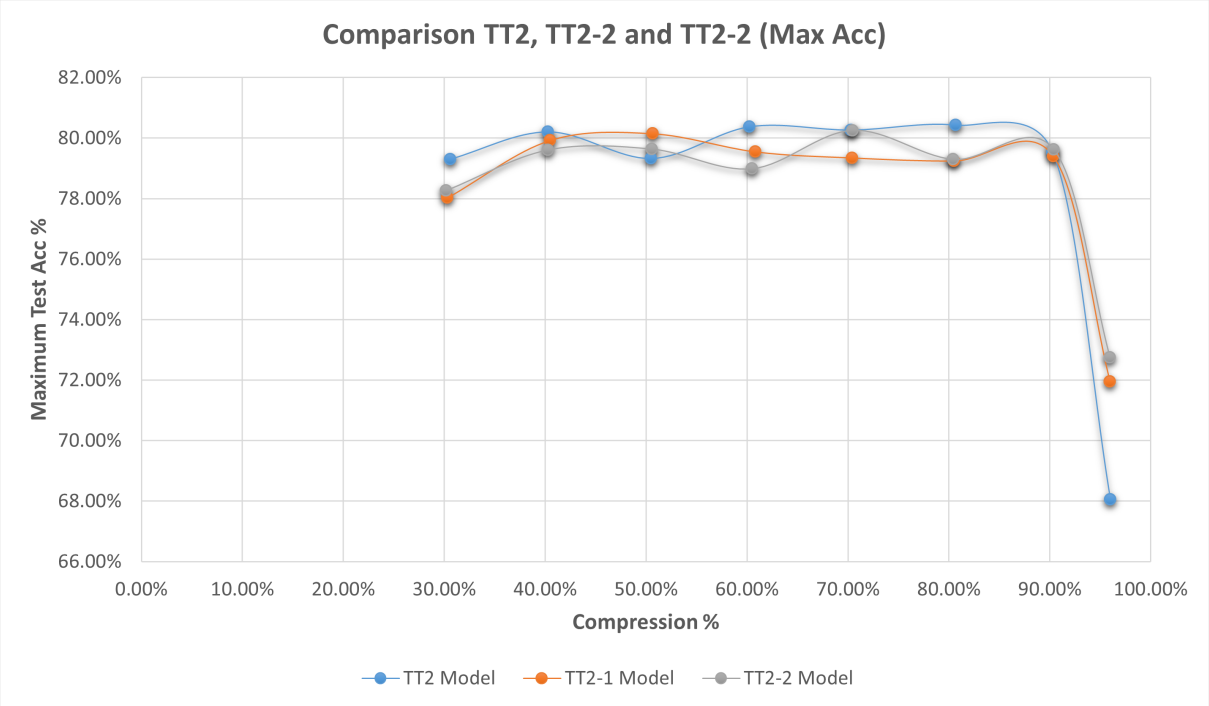


Figure 5-10: Plot that shows the evolution of maximum test accuracy in the TT2, TT2-1 and TT2-2 models with respect to eight different compression rates, ranging from 30% to 95.98%. The trend for TT2 model is depicted by the blue line, the trend for TT2-1 model is depicted by the orange line and the trend for the TT2-2 model is depicted by the grey line.

five. For each model, the maximum test accuracy at each compression rate will be plotted.

The final selected combinations from the TT2, TT2-1 and TT2-2 models will be used for this comparison that were reported in the previous sections. The plot that shows the evolution of the maximum test accuracy with respect to the compression rate is given in figure 5-10. From the plot given in figure 5-10 it can be observed that the evolution of the maximum test accuracy with respect to the compression rate is similar when the compression rate is between 30% and 90%. But when the compression rate is around 95.98%, a clear difference between all three models can be observed. In the case of TT2-1, at 95.98% the maximum test accuracy is 71.94% which is higher than the maximum test accuracy in the case of the TT2 model. The TT2 model achieves a maximum test accuracy of only 68.05% at 95.99% compression rate.
The TT2-2 model achieves the best maximum test accuracy of 72.74% at 95.98% compression rate. This is 0.8% more then that of TT2-1 model and 4.69% more than that of the TT2 model.

Table 5-6 shows the maximum possible TT-ranks for all TT2 models. From table 5-6 it can be observed that the maximum possible TT-ranks for each model depend on the order in which the factors are placed. A possible explanation for TT2-2 model's ability to achieve a higher performance than the TT2-1 and TT2 models can be hypothesised from the structure of the respective models. In the TT2-2 model, the maximum possible TT-ranks have values that are closer to each other than in the case of the TT2 model. In the TT2-2 model, R_2

Model	Maximum TT-ranks
TT2	(10,100,1400,60)
TT2-1	(10,140,600,10)
TT2-2	(10,240,200,10)

Table 5-6: Maximum possible TT-ranks in the TT2, TT2-1 and TT2-2 models. The TT-ranks are placed as (R_1,R_2,R_3,R_4). The factors are arranged differently in each model.

and R_3 ranks can each take a maximum value of 240 and 200. These values are very close to each other. In TT2 model, R_2 and R_3 ranks can each take a maximum value of 100 and 1400, which have a huge gap.

When a particular TT-rank has a higher value it can take, it means that TT-cores that are connected to the TT-rank have a higher influence on the expressive power of the model. An official definition of the term expressive power for a NN based language architecture does not exist.
However, several studies exist wherein, researchers explored the concept of expressive power. Cohen et al. [6] conducted a study on the expressive power of deep learning models and Khrulkov et al. [22] specifically studied the expressive power of RNNs. In a work done by Glasser et al. [12] on the expressive power of NN, the expressive power of a NN is defined as the set of functions that the NN can efficiently represent. The set of functions are calculated by NNs to map the input data to an output. The same definition is stated in a work done by Raghu et al. [35] on understanding the expressive power of deep NNs. With respect to TT-decomposition, in a research study conducted by Chen et al. [5] on support tensor train machines, it is reported that the expressive power of a tensor train increases with its tensor train ranks (number of tensor train ranks).

In the TT2-2 model, due to the arrangement of factors, the TT-cores are connected to the ranks R_2 and R_3 have more control on the expressive power of the model. Both ranks R_2 and R_3 have a higher value that they can take and hence three TT-cores in the TT2-2 model have a higher influence on the expressive power of the model. In the TT2 model, only one rank R_3 has high maximum possible value that it can take, and hence two TT-cores have a higher influence on the expressive power of the model. In the TT2-1 model, the ranks R_2 and R_3 have a maximum value of 140 and 600. This gap between the maximum values of the ranks here is less than that of the TT2 model but a bit higher than that in the case of the TT2-2 model.

A pattern that can be observed here is that when more number of TT-cores have an influence on the expressive power of the model, the model seems to have a higher maximum test accuracy. This can be seen in the case of TT2-2 model, where the ranks R_2 and R_3 have three TT-cores connected to them, that have a similar influence on the expressive power of the model because their maximum possible values are closer to each other. The TT2-2 model achieves the best test accuracy amongst all the three TT2 models. The TT2-1 model, achieves a maximum test accuracy of 71.94%. This is greater than the test accuracy of the TT2 model but less than that of the TT2-2 model. The gap between the maximum values of ranks R_2 and R_3 (140,600) in the TT2-1 is lesser than that in the case of the TT2 model (100,1400).

TT3-1 Ranks	Parameters	Compression %	Max Acc %
(10,100,337,500,56,14)	2.312857 M	30.59%	79.52%
(10,50,320,460,56,14)	1.966257 M	40.99%	79.44%
(10,50,337,360,56,14)	1.656557 M	50.28%	79.83%
(10,50,337,274,56,14)	1.318577 M	60.42%	80.18%
(10,100,100,500,56,14)	0.985657 M	70.42%	80.12%
(10,100,337,78,56,14)	0.654397 M	80.36%	79.65%
(10,50,150,90,56,14)	0.311057 M	90.66%	79.74%
(4,6,12,12,12,10)	0.136437 M	95.90%	77.10%

Table 5-7: Results obtained for the TT3-1 model. The TT3-1 ranks are reported as (R_1,R_2,R_3,R_4,R_5,R_6), 'Max Acc' is the maximum accuracy % that is achieved in 10 epochs by the LSTM model. The compression rates are defined with respect to the full model parameters (3.332225 M).

5-6 TT3-1 and TT3-2 models

The TT3-1 and TT3-2 models have seven factors each just like the TT3 model but the order of factors is different in each model.
The maximum possible values for each TT-rank in the TT3-1 model are calculated as:

$$R_1 \leq 10, R_2 \leq 100, R_3 \leq 600, R_4 \leq 560, R_5 \leq 56, R_6 \leq 14.$$

The maximum possible values for each TT-rank in the TT3-2 model are calculated as:

$$R_1 \leq 10, R_2 \leq 40, R_3 \leq 400, R_4 \leq 840, R_5 \leq 84, R_6 \leq 14.$$

The final combination of TT-ranks for different compression rates along with the number of parameters (M) and maximum test accuracy (%) for the TT3-1 model are given in table 5-7. It can be observed from table 5-7 that the combination of (R_1,R_2,R_3,R_4,R_5,R_6) as (10,100,337,500,56,14) yields a compression rate of 30.59%. Keeping that as a reference, the TT-ranks are varied.
From table 5-7, it can be observed that at a compression rate of 90.66% with only 0.311057 M, the TT3-1 model achieves a maximum test accuracy of 79.74%. The test accuracy does not drastically change in the case of TT3-1. At 95.90% compression rate, the TT3-1 model achieves a maximum test accuracy of 77.10%.
All the other combinations of TT-ranks considered for the TT3-1 model along with the maximum test accuracy in each case are reported in section C-5 of the appendix.

The final combination of TT-ranks for different compression rates along with the number of parameters (M) and maximum test accuracy (%) for the TT3-2 model are given in table 5-8. It can be observed from table 5-8 that the combination of (R_1,R_2,R_3,R_4,R_5,R_6) as (10,40,400,418,84,14) gives a compression rate of 30.24%. Keeping that as a reference, the TT-ranks are varied.

TT3-2 Ranks	Parameters	Compression %	Max Acc %
(10,40,400,418,84,14)	2.34297 M	30.24%	79.55%
(10,40,324,418,84,14)	1.976217 M	40.69%	78.34%
(10,40,400,280,84,14)	1.656377 M	50.29%	79.46%
(10,40,178,418,84,14)	1.307537 M	60.75%	80.36%
(10,40,242,232,84,14)	0.994297 M	70.16%	79.14%
(10,40,34,418,84,14)	0.648017 M	80.55%	79.07%
(10,40,100,125,16,14)	0.320465 M	90.38%	79.75%
(8,8,10,10,10,10)	0.136101 M	95.91%	76.54%

Table 5-8: Results obtained for the TT3-2 model. The TT3-2 ranks are reported as (R_1,R_2,R_3,R_4,R_5,R_6), 'Max Acc' is the maximum accuracy % that is achieved in 10 epochs by the LSTM model. The compression rates are defined with respect to the full model parameters (3.332225 M).

From table 5-8, it can be observed that at a compression rate of 90.38% with only 0.320465 M, the TT3-2 model achieves a maximum test accuracy of 79.75%. Like the TT3-1 model, the test accuracy does not drastically change in the case of TT3-2 either. At 95.91% compression rate, the TT3-2 model achieves a maximum test accuracy of 76.54%.
All the other combinations of TT-ranks considered for the TT3-2 model along with the maximum test accuracy in each case are reported in section C-6 of the appendix.

5-6-1 Effect of TT-ranks in TT3-1 model

Table C-27 given in section C-5 of the appendix shows the change in maximum test accuracy when R_3 and R_4 are varied in the TT3-1 model. The plot that shows change of rank R_3 with respect to the maximum test accuracy is given in figure 5-11. In this case, the rank R_3 is varied while all other TT-ranks are fixed. The plot that shows change of rank R_4 with respect to the maximum test accuracy is given in figure 5-12.

The number of parameters when $R_3 = 1$ is 0.431257 M. The number parameters when $R_4 = 1$ is 0.351787 M. From figures 5-11 and 5-12, it can be observed that the trend for both the plots are very similar. This means that the test accuracy of the model drops when both R_3 and R_4 have a low value. But the model is not particularly sensitive to neither of them.
However, the TT3-1 model is sensitive to the rank R_5. When $R_5 = 1$, the number of parameters is 2.034777 M. But the test accuracy of the model drops to 73.66%. This means that the TT3-1 model is sensitive to rank R_5 which links the TT-cores $\mathbf{G}_{(5)}$ and $\mathbf{G}_{(6)}$. The TT-core $\mathbf{G}_{(5)}$ is linked to the rank R_4 which has a maximum rank of 560.
The TT3-1 model is also sensitive to change is rank R_6. With $R_6 = 1$ and the number of parameters being 2.309763 M, the TT3-1 model achieves a test accuracy of 75.90%. This means that the model is sensitive to changes in rank R_6 too.
In order to obtain the best performance, it is important to make sure that the values of both ranks R_5 and R_6, are not below a certain threshold value.

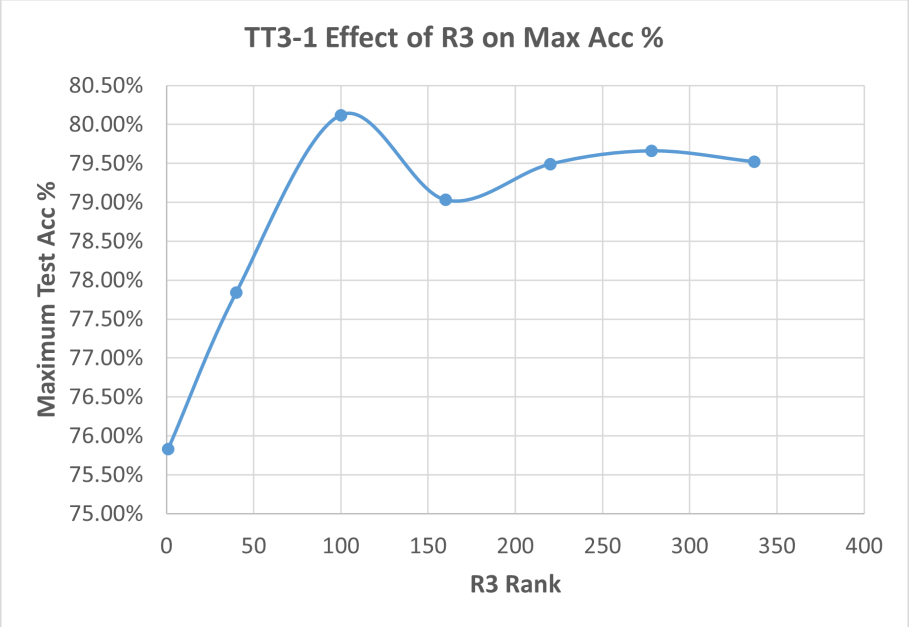


Figure 5-11: Plot between value of rank R_3 and maximum test accuracy % for the TT3-1 model. The rank R_3 is varied from 337 to 1, when R_1, R_2, R_4, R_5 and R_6 are fixed at 10, 100, 500, 56 and 14,

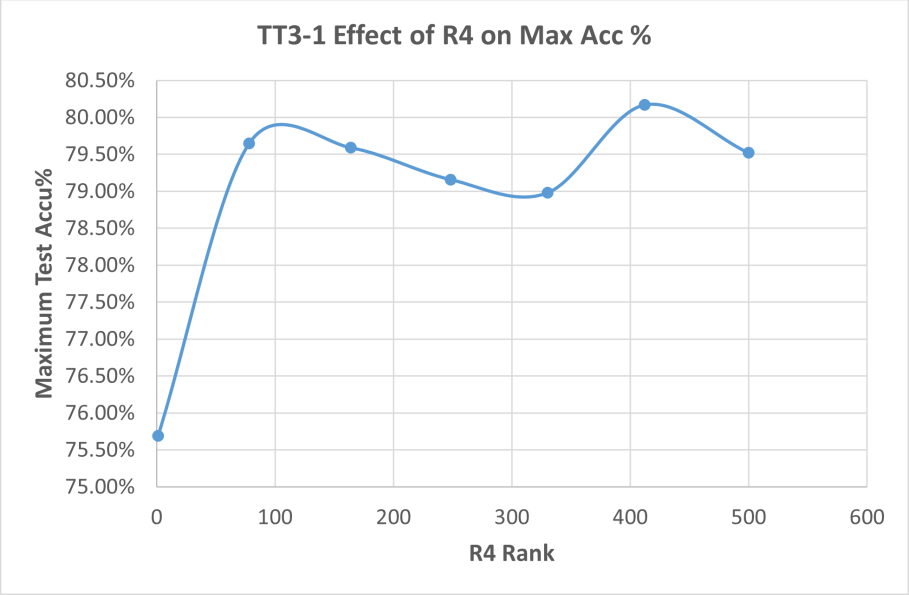


Figure 5-12: Plot between value of rank R_4 and maximum test accuracy % for the TT3-1 model. The rank R_4 is varied from 500 to 1, when R_1, R_2, R_3, R_5 and R_6 are fixed at 10, 100, 337, 56 and 14 respectively.

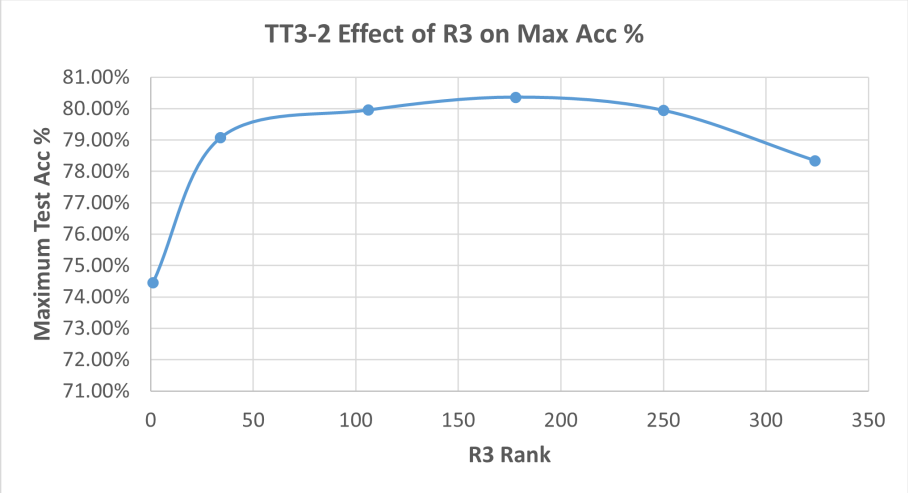


Figure 5-13: Plot between value of rank R_3 and maximum test accuracy % for the TT3-2 model. The rank R_3 is varied from 400 to 1, when R_1, R_2, R_4, R_5 and R_6 are fixed at 10, 40, 418, 84 and 14, respectively.

5-6-2 Effect of TT-ranks in TT3-2 model

Table C-34 given in section C-6 of the appendix shows the change in maximum test accuracy when R_3 and R_4 are varied in the TT3-2 model. The plot that shows change of rank R_3 with respect to the maximum test accuracy is given in figure 5-13. In this case, the rank R_3 is varied while all other TT-ranks are fixed. The plot that shows change of rank R_4 with respect to the maximum test accuracy is given in figure 5-14.

The number of parameters when $R_3 = 1$ is 0.496877 M. The number parameters when $R_4 = 1$ is 0.310857 M. From figures 5-13 and 5-14, it can be observed that the trend for both the plots is not similar. The test accuracy when $R_3 = 1$ is 74.45% and the test accuracy when $R_4 = 1$ is 71.59%. But this could be due to fewer number of parameters when $R_4 = 1$. Hence, it is difficult to say if the model is sensitive to changes in either of these ranks.
However, the TT3-2 model is sensitive to changes in ranks R_5 and R_6. When $R_5 = 1$, the number of parameters is 1.970385 M but the test accuracy is only 74.90%.
When $R_6 = 1$, the number of parameters is 2.317563 M but the test accuracy is only 75.53%.
This suggests that the TT3-2 model is more sensitive to changes in ranks R_5 and R_6. Rank R_5 links TT-cores $\mathbf{G}_{(5)}$ and $\mathbf{G}_{(6)}$ while R_6 links TT-cores $\mathbf{G}_{(6)}$ and $\mathbf{G}_{(7)}$.
TT-core $\mathbf{G}_{(5)}$ is in-turn linked to R_4 which is also the rank with the highest maximum value (840).
To summarise the effect of TT-ranks in the compressed LSTM model, table 5-9 lists the TT-ranks in each TT-model the had affected the maximum test accuracy%. The maximum test accuracy% was sensitive to changes in these TT-ranks in the respective TT-models.

From table 5-9, it can be observed in case of the TT1 models, in both TT1-1 and TT1-2, the maximum test accuracy is sensitive to changes in rank R_2. This suggests that the maximum test accuracy is sensitive to the ranks that are linked to the TT-cores which are connected to the maximum TT-rank.
In case of the TT2 models, ranks R_3 and R_4 seem to affect the maximum test accuracy. Rank R_3 is connected the TT-cores, that are linked to the maximum TT-rank in the model and R_4

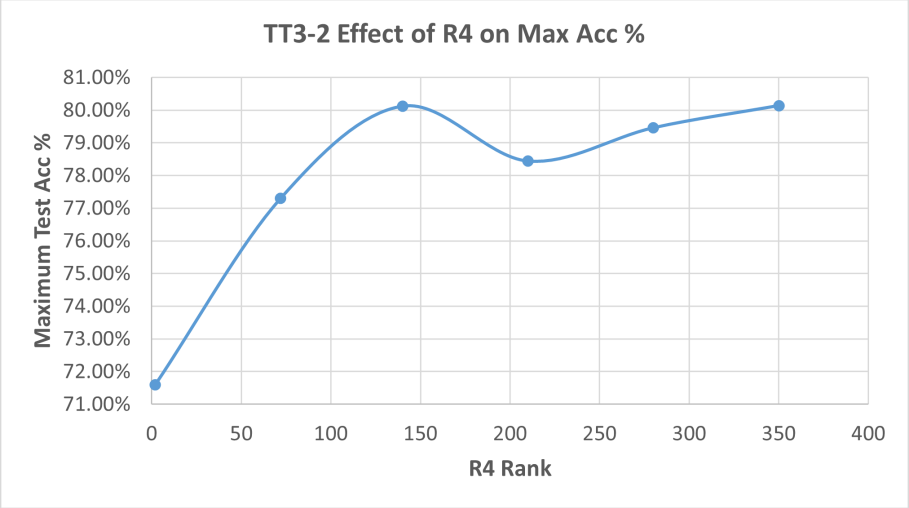


Figure 5-14: Plot between value of rank R_4 and maximum test accuracy % for the TT3-2 model. The rank R_4 is varied from 418 to 1, when R_1, R_2, R_3, R_5 and R_6 are fixed at 10, 40, 400, 84 and 14 respectively.

Model	Sensitive TT-ranks	Maximum TT-Rank
TT1	None	$R_1 \leq 100$, $R_2 \leq 280$
TT1-1	R_2	$R_1 \leq 140$, $R_2 \leq 100$
TT1-2	R_2	$R_1 \leq 100$, $R_2 \leq 120$
TT2	R_4	$R_1 \leq 10$, $R_2 \leq 100$, $R_3 \leq 1400$, $R_4 \leq 60$.
TT2-1	R_3	$R_1 \leq 10$, $R_2 \leq 140$, $R_3 \leq 600$, $R_4 \leq 10$.
TT2-2	R_3, R_4	$R_1 \leq 10$, $R_2 \leq 240$, $R_3 \leq 200$, $R_4 \leq 10$.
TT3	R_5	$R_1 \leq 4$, $R_2 \leq 24$, $R_3 \leq 240$, $R_4 \leq 1400$, $R_5 \leq 140$, $R_6 \leq 14$.
TT3-1	R_5, R_6	$R_1 \leq 10$, $R_2 \leq 100$, $R_3 \leq 600$, $R_4 \leq 560$, $R_5 \leq 56$, $R_6 \leq 14$.
TT3-2	R_5, R_6	$R_1 \leq 10$, $R_2 \leq 40$, $R_3 \leq 400$, $R_4 \leq 840$, $R_5 \leq 84$, $R_6 \leq 14$.

Table 5-9: The TT-ranks that have a dominant effect on the maximum test accuracy in each TT-model are listed under 'Sensitive TT-ranks'. The maximum TT-ranks in each model are also given.

Model	Maximum TT-ranks
TT3	4,24,240,1400,140,14
TT3-1	10,100,600,560,56,14
TT3-2	10,40,400,840,84,14

Table 5-10: Maximum possible TT-ranks in the TT3, TT3-1 and TT3-2 models. The TT-ranks are placed as R_1,R_2,R_3,R_4,R_5,R_6. The factors are arranged differently in each model.

is the TT-rank that connects the last two TT-cores of the model. This suggests that the rank that connects the TT-cores on the right end also affects the maximum test accuracy. In the case of the TT3 models, ranks R_5 and R_6 have an effect on the maximum test accuracy. These ranks are either linked to the TT-cores that are in-turn linked to the maximum TT-rank or they link the last two TT-cores of the model.

Hence, it can be concluded that the TT-ranks that are linked to the last two TT-cores, and the TT-ranks that are linked to the those TT-cores, which are in-turn connected to the maximum TT-rank of the model, have an impact on the performance of the TT-model. The maximum test accuracy does not seem to be sensitive to changes in the TT-ranks that are linked to the first two TT-cores.

5-7 Comparison of TT3, TT3-1 and TT3-2 models

The results obtained for the TT3-1 and TT3-2 models will be compared with the results of the TT3 model. For each model, the maximum test accuracy at each compression rate will be plotted.

The final selected combinations from the TT3, TT3-1 and TT3-2 models will be used for this comparison, which were reported in the previous sections. The plot that shows the evolution of the maximum test accuracy with respect to the compression rate is given in figure 5-15. From the plot given in figure 5-15, it can be observed that the evolution of the maximum test accuracy with respect to the compression rate is similar when the compression rate is between 30% and 90%. When the compression rate is around 95.90%, in TT3-1 and TT3-2 the test accuracy reduces to only 77.10% and 76.54% respectively. The maximum test accuracy does not drastically drop. The TT3-1 model achieves the best test accuracy of 77.10% among all the three models. The next best performance score was achieved by the TT3-2 model. The TT3 model has the lowest test accuracy of 73.09% at a compression rate of 95.93%.

To analyse the results obtained, the maximum TT-ranks for each model is given in table 5-10. As mentioned in the TT2-1 and TT2-2 cases, a performance boost in the TT3-1 and TT3-2 models can also be explained with the concept of expressive power of the model. The factors in the TT3-1 model are arranged in such a way that the TT-ranks R_3 and R_4 have a maximum possible value of 600 and 560 respectively. This means that the TT-cores that are linked to them have a higher influence on the expressive power of the model. In the TT3-1 model, three TT-cores have a higher influence on the expressive power just like that in the

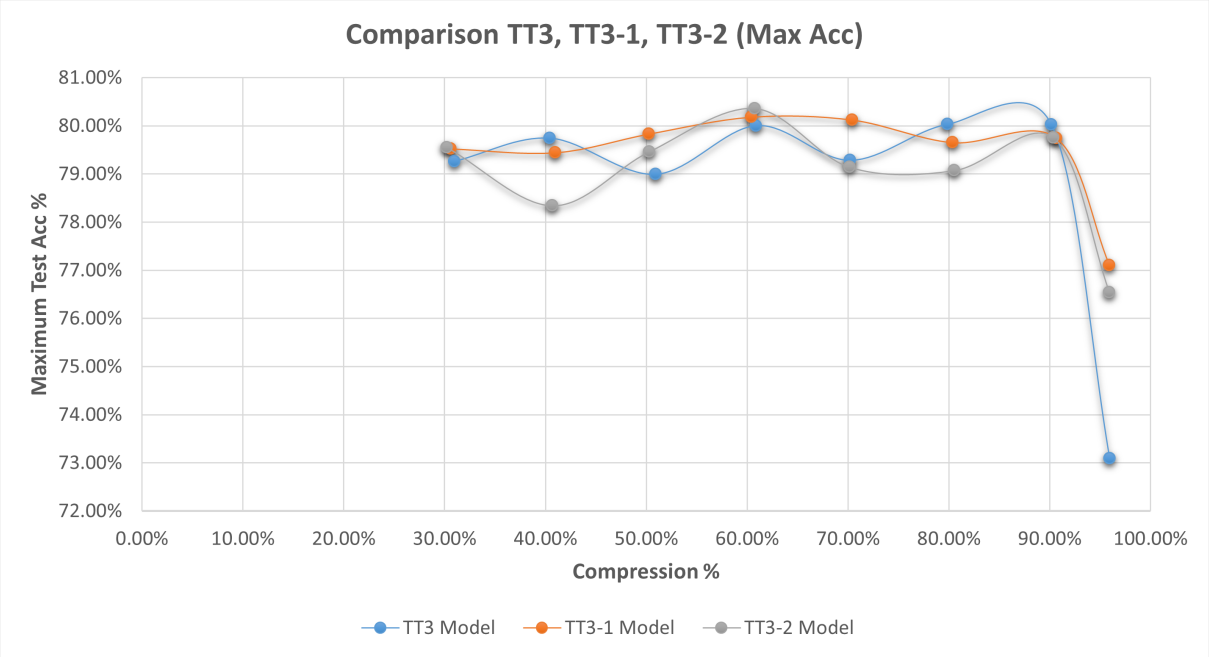


Figure 5-15: Plot that shows the evolution of maximum test accuracy in the TT3, TT3-1 and TT3-2 models with respect to eight different compression rates, ranging from 30% to 95.90%. The trend for TT3 model is depicted by the blue line, the trend for TT3-1 model is depicted by the orange line and the trend for the TT3-2 model is depicted by the grey line.

case of the TT2-2 model. In the case of the TT3-1 model, both ranks R_3 and R_4 can be tuned in order to control the model's performance.
In the case of TT3-2, the ranks R_3 and R_4 have a maximum value of 400 and 840, respectively. Even here, three TT-cores that are linked to both R_3 and R_4 have a higher influence on the expressive power. But the gap between the ranks is slightly higher than that in the case of TT3-1. Hence, the best performance at a compression rate close to 95% is achieved by the TT3-1 model, followed by the TT3-2 model.
In the TT3 model, the ranks R_3 and R_4 have a maximum value of 240 and 1400, respectively. This means that only two TT-cores that are connected to the rank R_4 have a higher influence on the expressive power. This also explains why the TT3 model achieves the lowest test accuracy of 73.09% among these three models.

By looking at the results obtained during the investigation into the second research question, it can be concluded that re-arranging the factors such that the TT-ranks are closer to each other helps to increase the maximum test accuracy of the model. This effect was not visible in the case of TT1-1 and TT1-2 models, even though the TT-ranks were closer to each other. A possible reason for that could be due to the fact that in the original TT1 model, the TT-ranks already had a smaller gap (in the TT1 model, $R_1 \leq 100$ and $R_2 \leq 280$) between them.

The TT3-1 model achieves the best performance with a test accuracy of 77.10% at a compression rate close to 95%. Hence, the TT3-1 model performs the best amongst all the models that were investigated in this thesis work. This means that when the embedding matrix is

factorized and when the factors in the TT-matrix are placed in such a way that more number of TT-cores have a higher influence on the expressive power, the maximum test accuracy of the model can be improved leading to a performance boost.

Chapter 6

Conclusion and future work

This thesis investigated the application of the TT-decomposition method to compress an LSTM model, which can be used for NLP applications. Firstly, the concepts of NLP, deep learning and the working of an LSTM model were explained in chapter 2. The concept of word vectors and the basic idea behind the embedding matrix in a LM, was explained in detail in chapter 3. Few basic concepts of tensors, their properties and the application of the TT-decomposition in the field of NLP and the work done by Hrinchuk et al. [19] were described in chapter 4. Finally, the work done in this thesis was reported in detail in chapters 4 and 5.
The following research questions are answered after the investigation conducted for the thesis.

- *In an LSTM model used to perform NLP, when standard embeddings are replaced by TT-embeddings, how does the factorization of the embedding matrix and subsequently the TT-ranks effect the test accuracy and the compression rate?*

  In the LSTM model, when the embedding matrix was factorized using TT-matrix, the model having a greater number of factors achieved the best performance. The TT3 model achieved a maximum test accuracy of 73.09% at a compression rate of 95.99%. When the effect of TT-ranks on test accuracy was investigated, it was found that the TT-ranks which are linked to the TT-cores that are in-turn linked to the maximum TT-rank had a dominant effect on the test accuracy. The test accuracy was found to be sensitive to changes in these TT-ranks.

- *In the same LSTM model used to perform NLP, when TT-embeddings are used, how does the order of the factors in each factorization effect the compression rate and the test accuracy of the model?*

  In the LSTM model with the same factors used for research question 1, the order of the factors was changed to get different TT-ranks. When the TT-ranks had a smaller gap

in the maximum value, the model attained a performance boost. This can be attributed to more number of TT-cores in the model having a higher influence on the expressive power of the model. It can be hypothesized that when more number of TT-cores in the TT model have a higher influence on the expressive power of the model, the performance of the LSTM model increases. The TT3-1 model, with seven factors and three TT-cores having higher influence on expressive power, attained the best performance at a compression rate of 95.90%. With only 0.136437 M parameters in the model, the TT3-1 model achieved a test accuracy of 77.10%. In this case, the number of parameters was reduced by almost 24.5 times (compared to the full model) and the test accuracy was only reduced by 0.94% (compared to the full model).
During the investigation of the effects of TT-ranks, it was found that the TT-ranks that were linked to the TT-cores which were in-turn connected to the maximum TT-rank, had an effect on the maximum test accuracy. The TT-ranks that are connected to the last two TT-cores also had an effect on the maximum test accuracy.

Based on the empirical results obtained for the above two research questions, the following steps are recommended for future research when the embedding layer of any LM has to be compressed for NLP applications using the TT-decomposition:

- When the embedding layer is represented in the TT-matrix format, choose a factorization with a larger number of factors and arrange the factors in such a way that the TT-ranks (maximum possible values) have a smaller gap in order to attain the best performance in the compressed LM.
- Avoid reducing the values of the last two TT-ranks/ the TT-ranks linked to the TT-core in turn linked to TT-rank which has the highest maximum possible value, below a certain threshold. Otherwise, the performance of the compressed LM might drop.

During this thesis work, a new research question was formed that could be further investigated. It can be hypothesized that when the factors of the embedding matrix are arranged in such a way that the TT-ranks have a smaller gap, the expressive power of the model is distributed amongst more TT-cores. A performance boost was observed in this case, especially in the TT2-1, TT2-2, TT3-1 and TT3-2 models. However, the concept of the expressive power of the TT-cores has not been fully explored. I strongly believe that more investigation and research into the distribution of expressive power among TT-cores can lead to the establishment of a mathematical definition for the expressive power of NNLM when TT-decomposition is used to compress the model. Additionally, a new regularization method for improving performance (based on the values of the maximum value of the TT-ranks) can be developed based on the theoretical proof obtained in future work. Based on the suggestions, a possible research question could then be formulated as:

In the LSTM model or any LM, when the embedding matrix is represented in the TT-matrix format and when the factors are arranged in such a way that the TT-ranks have a maximum possible rank closer to each other, does this provide a boost to the maximum test accuracy? Does the expressive power of the TT model increase in this case?

Apart from the expressive power of the TT model, the generalization capability of a TT model can also be investigated. In the case of the TT3 model, it was concluded through empirical results that the TT3 model has better generalization capability than the TT1 and TT2 models. However, a mathematical explanation to this was not provided. A study to formulate a mathematical definition for the generalization capability in a TT model for NLP applications is a promising topic for future work.

Appendix A

Algorithm to compute mapping

To obtain TT-embeddings, the first step to compute the embedding for a word indexed i in the vocabulary is to map the row index i into the N-dimensional vector index. The pseudo code for the mapping of i into $(i_1, i_2, ..., i_N)$ is given in figure A-1.

Require: I – vocabulary size, $\{I_k\}_{k=1}^{N}$ – an arbitrary factorization of I,
i – index of the target word in vocabulary.
Returns: $\mathcal{I}(i) = (i_1, \ldots, i_N)$ – N-dimensional index.
Initialize: $L = \{1, I_1, I_1 I_2, \ldots, I_1 I_2 \ldots I_{N-1}\}$
for $k = N$ **to** 1 **do**
  $i_k \leftarrow \texttt{floor}(i / L[k])$
  $i \leftarrow i \mod L[k]$
end for

Figure A-1: Algorithm to compute mapping to N-dimensional vector index [19].

Appendix B

Additional results - TT1,TT2 and TT3 models

In this appendix, the different combinations that were chosen to achieve different compression rates for the TT1, TT2 and TT3 models are reported.

B-1 TT1 model

The tables B-1, B-2 and B-3 shows the test accuracy after 10 epochs ('Test Acc') maximum test accuracy ('Max Acc'), the compression rate and the number of parameters for each combination of the TT-ranks for the TT1 model.

B-2 TT2 model

The following tables show the test accuracy after 10 epochs ('Test Acc') maximum test accuracy ('Max Acc'), the compression rate and the number of parameters for each combination of the TT-ranks for the TT2 model.

TT1-ranks	Parameters	Compression %	Test Acc	Max Acc
(100,178)	2.328065 M	30.29%	76.38%	78.59%
(84,178)	1.984705 M	40.43%	75.78%	78.78%
(68,178)	1.641345 M	50.74%	77.32%	79.10%
(52,178)	1.297985 M	61.04%	77.91%	79.64%
(38,178)	0.997545 M	70.06%	74.96%	79.12%
(20,178)	0.611265 M	81.65%	75.59%	79.68%
(5,178)	0.289365 M	91.31%	77.53%	79.46%

Table B-1: Results for the TT1 model when R_1 is varied while R_2 is constant.

TT1-ranks	Parameters	Compression %	Test Acc	Max Acc
(100,178)	2.328065 M	30.29%	76.38%	78.59%
(100,160)	1.984225 M	40.45%	77.26%	78.90%
(100,120)	1.615825 M	51.51%	77.99%	79.18%
(100,95)	1.308825 M	60.72%	77.86%	78.50%
(100,70)	1.001825 M	69.93%	78.53%	78.68%
(100,45)	0.694825 M	79.14%	76.10%	79.18%
(100,15)	0.326425 M	90.20%	77.40%	79.86%
(100,1)	0.154505 M	95.36%	72.85%	72.85%

Table B-2: Results for the TT1 model when R_2 is varied while R_1 is constant.

TT1-ranks	Parameters	Compression %	Test Acc	Max Acc
(100,178)	2.328065 M	30.29%	76.38%	78.59%
(90,168)	2.002665 M	39.90%	78.35%	79.40%
(90,110)	1.360025 M	59.18%	76.52%	79.46%
(73,93)	0.980245 M	70.58%	79.22%	79.22%
(35,120)	0.673325 M	79.79%	79.46%	79.46%
(30,50)	0.329225 M	90.11%	76.98%	79.46%
(10,10)	0.148025 M	95.55%	78.66%	79.22%
(1,1)	0.132725 M	96.01%	69.50%	69.50%

Table B-3: Results for the TT1 model when both ranks R_1 and R_2 are varied.

TT2-ranks	Parameters	Compression %	Test Acc	Max Acc
(10,100,570,60)	2.311925 M	30.61%	75.92%	79.30%
(10,60,570,60)	1.988725 M	40.31%	78.86%	80.20%
(10,20,570,60)	1.665525 M	50.01%	75.50%	79.98%
(10,1,570,60)	1.512005 M	54.62%	78.39%	78.39%

Table B-4: Results for the TT2 model when R_2 is varied while ranks R_1, R_3 and R_4 are constant.

TT2-ranks	Parameters	Compression %	Test Acc	Max Acc
(10,100,570,60)	2.311925 M	30.61%	75.92%	79.30%
(10,100,485,60)	1.988925 M	40.31%	77.70%	79.65%
(10,100,400,60)	1.665925 M	50.00%	76.94%	79.69%
(10,100,310,60)	1.323925 M	60.26%	78.77%	80.37%
(10,100,222,60)	0.989525 M	70.30%	78.46%	80.26%
(10,100,135,60)	0.658925 M	80.22%	77.46%	79.56%
(10,100,47,60)	0.324525 M	90.26%	78.15%	79.53%
(10,100,10,60)	0.183925 M	94.48%	77.46%	78.95%
(10,100,1,60)	0.149725 M	95.50%	72.69%	72.69%

Table B-5: Results for the TT2 model when R_3 is varied while ranks R_1, R_2 and R_4 are constant.

TT2-ranks	Parameters	Compression %	Test Acc	Max Acc
(10,100,570,31)	1.648985 M	50.51%	79.10%	79.32%
(10,100,570,17)	1.329845 M	60.11%	76.34%	79.31%
(10,100,570,1)	0.963185 M	71.09%	74.65%	74.65%

Table B-6: Results for the TT2 model when R_4 is varied while ranks R_1, R_2 and R_3 are constant.

TT2-ranks	Parameters	Compression %	Test Acc	Max Acc
(10,100,570,60)	2.311925 M	30.61%	75.92%	79.30%
(1,60,570,60)	1.983235 M	40.48%	78.71%	80.06%
(1,15,570,60)	1.623685 M	51.27%	77.58%	79.90%
(1,1,570,60)	1.511825 M	54.63%	78.42%	79.39%

Table B-7: Results for the TT2 model when R_1 and R_2 are varied while ranks R_3 and R_4 are constant.

TT2-ranks	Parameters	Compression %	Test Acc	Max Acc
(10,100,570,60)	2.311925 M	30.61%	75.92%	79.30%
(10,72,540,60)	1.983445 M	40.47%	76.75%	79.57%
(10,50,500,60)	1.690925 M	49.25%	76.37%	79.53%
(10,50,380,60)	1.318925 M	60.41%	75.90%	79.66%
(10,50,270,60)	0.977925 M	70.65%	76.82%	80.00%
(10,40,170,60)	0.643125 M	80.69%	80.30%	80.42%
(10,30,70,60)	0.336325 M	89.90%	77.46%	79.61%
(10,10,10,60)	0.162325 M	95.12%	78.89%	79.22%
(10,1,1,60)	0.138439 M	95.84%	73.59%	73.59%

Table B-8: Results for the TT2 model when R_2 and R_3 are varied while ranks R_1 and R_4 are constant.

TT2-ranks	Parameters	Compression %	Test Acc	Max Acc
(10,100,570,60)	2.311925 M	30.61%	75.92%	79.30%
(10,90,560,50)	1.969925 M	40.88%	77.70%	77.70%
(10,60,530,50)	1.646525 M	50.62%	77.53%	80.55%
(10,50,500,50)	1.490325 M	55.27%	75.94%	79.14%
(10,50,450,50)	1.355325 M	59.32%	77.06%	79.58%
(10,40,400,40)	1.002725 M	69.90%	78.30%	80.21%
(10,30,320,30)	0.655525 M	80.32%	78.92%	79.91%
(10,20,200,20)	0.351525 M	89.45%	76.88%	79.88%
(10,20,20,20)	0.157125 M	95.28%	78.55%	78.92%
(10,10,10,10)	0.139325 M	95.81%	78.21%	78.21%
(10,10,1,1)	0.133565 M	95.99%	68.05%	68.05%
(10,1,1,1)	0.132539 M	96.02%	51.67%	51.67%

Table B-9: Results for the TT2 model when R_2, R_3 and R_4 are varied while rank R_1 is constant.

TT2-ranks	Parameters	Compression %	Test Acc	Max Acc
(1,1,1,1)	0.132359 M	96.02%	52.65%	56.46%

Table B-10: Results for the TT2 model when all the ranks are varied.

TT3-ranks	Parameters	Compression %	Test Acc	Max Acc
(4,24,190,550,140,14)	2.013213 M	39.58%	76.02%	79.04%
(4,24,130,550,140,14)	1.668813 M	49.91%	78.66%	78.66%
(4,24,68,550,140,14)	1.312933 M	60.59%	78.79%	79.22%
(4,24,8,550,140,14)	0.968533 M	70.93%	77.05%	79.49%
(4,24,1,550,140,14)	0.928353 M	72.14%	79.26%	79.26%

Table B-11: Results for the TT3 model when R_3 is varied while R_1, R_2, R_4, R_5 and R_6 are constant.

TT3-ranks	Parameters	Compression %	Test Acc	Max Acc
(4,24,240,550,140,14)	2.300213 M	30.97%	79.13%	79.26%
(4,24,240,465,140,14)	1.977213 M	40.66%	76.42%	78.86%
(4,24,240,375,140,14)	1.635213 M	50.92%	78.99%	78.99%
(4,24,240,290,140,14)	1.3122113 M	60.62%	76.83%	78.70%
(4,24,240,205,140,14)	0.989213 M	70.31%	76.85%	79.66%
(4,24,240,120,140,14)	0.666213 M	80.00%	78.06%	80.51%
(4,24,240,30,140,14)	0.324213 M	90.27%	77.98%	79.42%
(4,24,240,1,140,14)	0.214013 M	93.57%	71.42%	71.84%

Table B-12: Results for the TT3 model when R_4 is varied while R_1, R_2, R_3, R_5 and R_6 are constant.

B-3 TT3 model

The following tables show the test accuracy after 10 epochs ('Test Acc') maximum test accuracy ('Max Acc'), the compression rate and the number of parameters for each combination of the TT-ranks for the TT3 model.

TT3-ranks	Parameters	Compression %	Test Acc	Max Acc
(4,24,240,550,100,14)	2.074613 M	37.74%	78.18%	79.68%
(4,24,240,550,84,14)	1.984373 M	40.44%	79.41%	79.74%
(4,24,240,550,68,14)	1.894133 M	43.15%	76.17%	79.45%
(4,24,240,550,28,14)	1.668533 M	49.92%	78.58%	79.10%
(4,24,240,550,1,14)	1.516253 M	54.49%	71.22%	71.22%

Table B-13: Results for the TT3 model when R_5 is varied while R_1, R_2, R_3, R_4 and R_6 are constant.

TT3-ranks	Parameters	Compression %	Test Acc	Max Acc
(4,24,240,550,140,1)	2.281831 M	31.52%	76.25%	76.40%

Table B-14: Results for the TT3 model when R_6 is varied while other ranks are constant.

TT3-ranks	Parameters	Compression %	Test Acc	Max Acc
(4,20,185,550,140,14)	1.977017 M	40.66%	77.26%	79.40%
(4,15,130,550,140,14)	1.656897 M	50.27%	77.47%	80.78%
(4,10,68,550,140,14)	1.303077 M	60.89%	79.15%	80.00%
(4,10,10,550,140,14)	0.978277 M	70.64%	77.04%	79.22%

Table B-15: Results for the TT3 model when R_2 and R_3 are varied while other ranks are constant.

TT3-ranks	Parameters	Compression %	Test Acc	Max Acc
(4,20,240,470,140,14)	1.986517 M	40.38%	76.38%	78.98%
(4,20,240,380,140,14)	1.644517 M	50.64%	77.99%	77.99%
(4,20,240,300,140,14)	1.340517 M	59.77%	76.87%	78.02%
(4,20,240,210,140,14)	0.998517 M	70.03%	77.94%	77.94%
(4,10,240,130,140,14)	0.670277 M	79.88%	79.27%	80.03%
(4,10,240,40,140,14)	0.328277M	90.14%	79.38%	79.88%
(4,1,240,1,140,14)	0.158261 M	95.25%	72.91%	72.91%

Table B-16: Results for the TT3 model when R_2 and R_4 are varied while other ranks are constant.

TT3-ranks	Parameters	Compression %	Test Acc	Max Acc
(4,20,240,550,85,14)	1.980317 M	40.57%	76.69%	78.66%
(4,10,240,550,30,14)	1.645877 M	50.60%	78.40%	78.98%
(4,1,240,550,1,14)	1.460501 M	56.17%	72.11%	72.11%

Table B-17: Results for the TT3 model when R_2 and R_5 are varied while other ranks are constant.

TT3-ranks	Parameters	Compression %	Test Acc	Max Acc
(4,24,215,500,140,14)	1.979213 M	40.60%	76.88%	77.02%
(4,24,185,450,140,14)	1.659513 M	50.19%	77.35%	78.50%
(4,24,185,350,140,14)	1.334513 M	59.95%	78.93%	79.86%
(4,24,185,245,140,14)	0.993263 M	70.19%	79.45%	79.58%
(4,24,155,160,140,14)	0.661813 M	80.13%	76.46%	79.61%
(4,24,60,80,140,14)	0.327013 M	90.18%	77.68%	78.78%
(4,24,4,4,140,14)	0.159333 M	95.21%	76.28%	76.28%

Table B-18: Results for the TT3 model when R_3 and R_4 are varied while other ranks are constant.

TT3-ranks	Parameters	Compression %	Test Acc	Max Acc
(4,24,220,550,110,14)	2.016213 M	39.50%	75.05%	78.85%
(4,24,155,550,110,14)	1.643113 M	50.69%	78.47%	79.55%
(4,24,110,550,100,14)	1.328413 M	60.13%	77.05%	79%
(4,24,70,550,80,14)	0.986013 M	70.40%	78.84%	79.28%
(4,24,40,550,40,14)	0.644613 M	80.65%	76.22%	76.22%
(4,24,10,550,20,14)	0.303213 M	90.90%	78.15%	79.07%
(4,24,1,550,1,14)	0.144393 M	95.66%	70.65%	70.65%

Table B-19: Results for the TT3 model when R_3 and R_5 are varied while other ranks are constant.

TT3-ranks	Parameters	Compression %	Test Acc	Max Acc
(4,24,220,520,120,14)	1.970613 M	40.86%	77.24%	80.31%
(4,24,200,490,100,14)	1.665013 M	50.03%	78.46%	78.46%
(4,24,200,380,100,14)	1.335013 M	59.93%	77.74%	79.60%
(4,24,190,280,90,14)	0.975213 M	70.73%	76.17%	79.39%
(4,24,180,170,90,14)	0.647813 M	80.55%	76.90%	79.06%
(4,24,100,90,80,14)	0.330213 M	90.09%	77.46%	79.16%
(4,24,24,24,24,14)	0.153653 M	95.38%	79.42%	79.70%

Table B-20: Results for the TT3 model when R_3, R_4 and R_5 are varied while other ranks are constant.

TT3-ranks	Parameters	Compression %	Test Acc	Max Acc
(4,20,190,550,140,1)	1.987135 M	40.36%	74.45%	74.45%
(4,20,130,550,140,1)	1.645135 M	50.62%	73.04%	74.34%
(4,12,76,550,140,1)	1.331063 M	60.00%	75.58%	75.58%
(4,12,12,550,140,1)	0.971383 M	70.84%	77.09%	77.09%

Table B-21: Results for the TT3 model when R_2, R_3 and R_6 are varied while other ranks are constant.

TT3-ranks	Parameters	Compression %	Test Acc	Max Acc
(4,20,240,470,140,1)	1.968135 M	40.93%	74.46%	74.46%
(4,20,240,390,140,1)	1.664135 M	50.05%	74.14%	74.64%
(4,20,240,300,140,1)	1.322135 M	60.32%	74.26%	74.77%
(4,20,240,200,140,1)	0.980135 M	70.58%	69.93%	69.93%

Table B-22: Results for the TT3 model when R_2, R_4 and R_6 are varied while other ranks are constant.

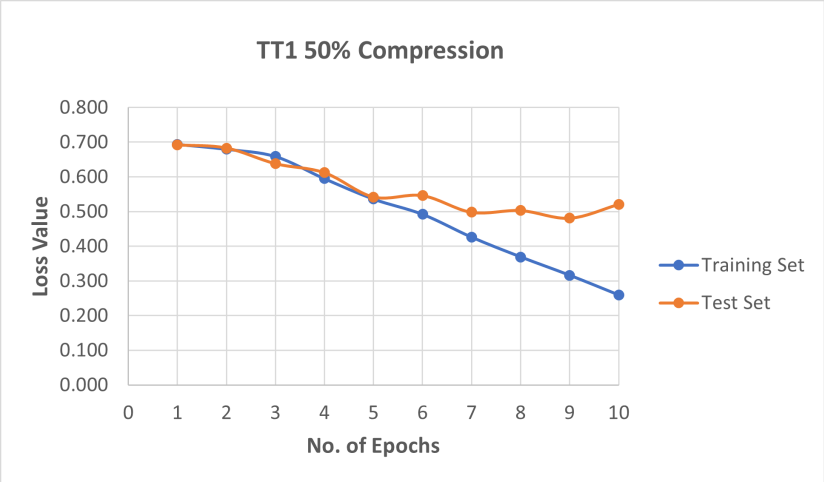
TT3-ranks	Parameters	Compression %	Test Acc	Max Acc
(4,8,8,8,10,8)	0.135425 M	95.93%	73.09%	73.09%

Table B-23: Results for the TT3 model when R_2, R_3 R_4, R_5 R_6 are varied while other rank R_1 is constant.

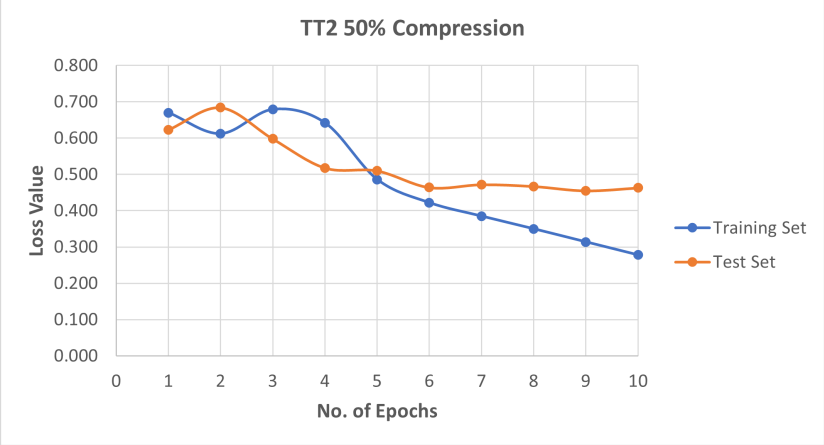
B-4 Evolution of loss in TT1, TT2 and TT3 models

The plots showing evolution of loss for the test loss and training loss when the compression rate is closer to 50% in the TT1, TT2 and TT3 models are given by figures B-1a, B-1b and B-1c respectively.

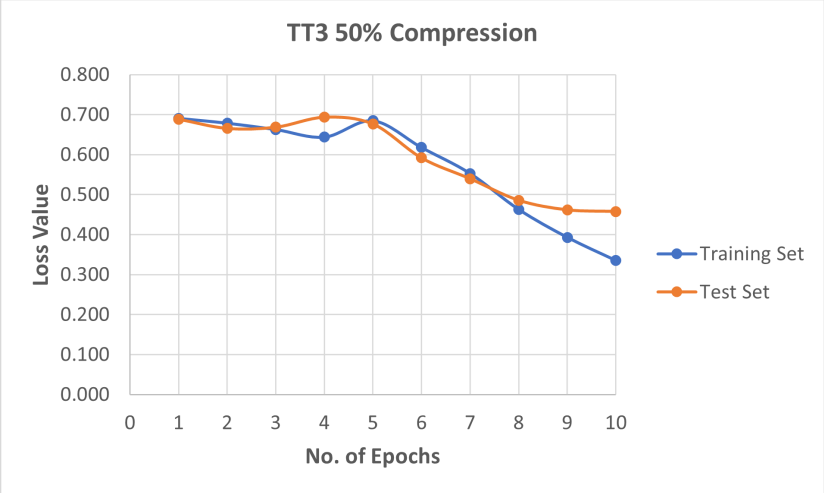
The plots showing evolution of loss for the test loss and training loss when the compression rate is closer to 70% in the TT1, TT2 and TT3 models are given by figures B-2a, B-2b and B-2c respectively.



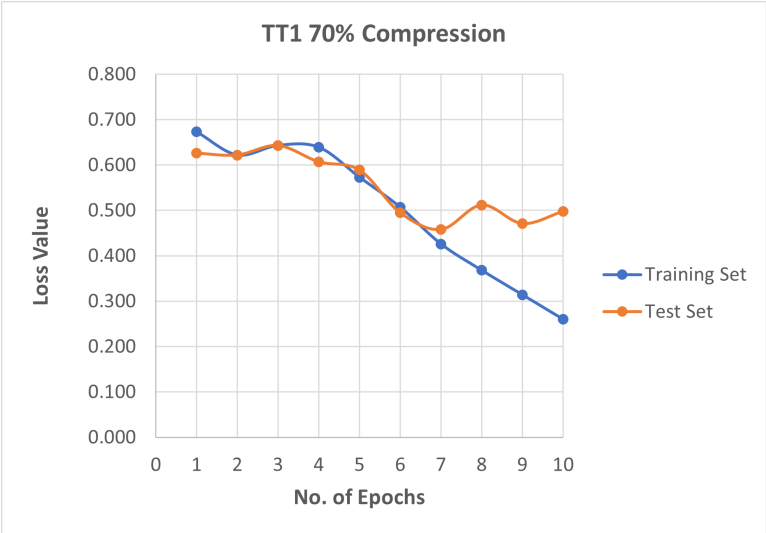
(a) Evolution of loss in the case of both training and test datasets in the TT1 model when the compression rate is close to 50%.



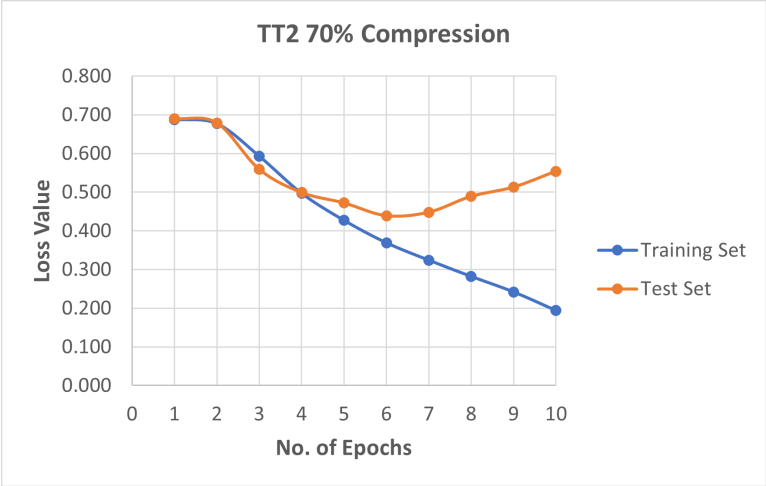
(b) Evolution of loss in the case of both training and test datasets in the TT2 model when the compression rate is close to 50%.



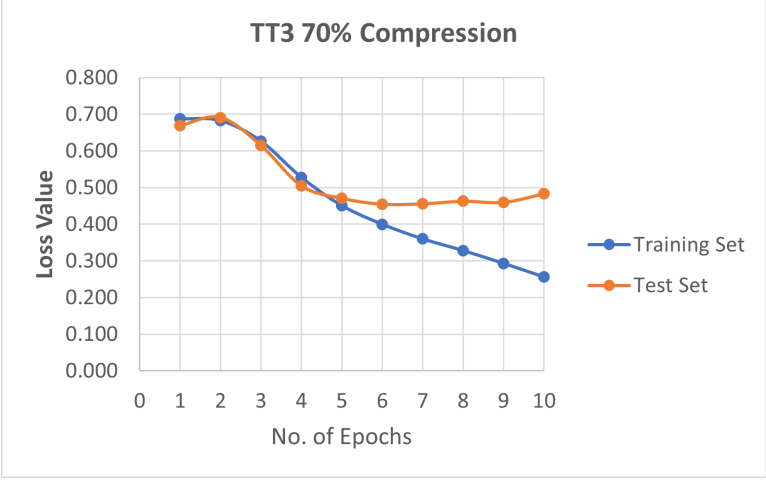
(c) Evolution of loss in the case of both training and test datasets in the TT3 model when the compression rate is close to 50%.



(a) Evolution of loss in the case of both training and test datasets in the TT1 model when the compression rate is close to 70%.



(b) Evolution of loss in the case of both training and test datasets in the TT2 model when the compression rate is close to 70%.



(c) Evolution of loss in the case of both training and test datasets in the TT3 model when the compression rate is close to 70%.

Appendix C

Additional Results - TT1-1, TT1-2, TT2-1, TT2-2, TT3-1, TT3-2 Models

In this appendix, the different combinations that were chosen to achieve different compression rates for the TT1-1, TT1-2, TT2-1, TT2-2, TT3-1 and TT3-2 models.

C-1 Results TT1-1 model

The following tables show the test accuracy after 10 epochs ('Test Acc') maximum test accuracy ('Max Acc'), the compression rate and the number of parameters for each combination of the TT-ranks for the TT1-1 model. The changes in maximum test accuracy with change in ranks R_1 and R_2 in the TT1-1 model are reported in table C-4.

C-2 Results TT1-2 model

The following tables show the test accuracy after 10 epochs ('Test Acc') maximum test accuracy ('Max Acc'), the compression rate and the number of parameters for each combination

TT1-1 ranks	Parameters	Compression %	Test Acc	Max Acc
(90,100)	2.314825 M	30.53%	76.61%	78.47%
(76,100)	1.976865 M	40.67%	79.30%	79.30%
(62,100)	1.638905 M	50.81%	77.81%	79.30%
(49,100)	1.325085 M	60.23%	76.00%	80.01%
(35,100)	0.987125 M	70.37%	77.04%	79.78%
(21,100)	0.649165 M	80.51%	77.73%	79.76%
(7,100)	0.311205 M	90.66%	76.66%	79.54%
(1,100)	0.166365 M	95.00%	78.36%	78.36%

Table C-1: Results for the TT1-1 model when R_1 is varied while R_2 is fixed.

TT1-1 ranks	Parameters	Compression %	Test Acc	Max Acc
(90,85)	1.989325 M	40.30%	77.76%	78.91%
(90,70)	1.663825 M	50.03%	76.71%	79.64%
(90,54)	1.316625 M	60.48%	76.54%	79.46%
(90,39)	0.991125 M	70.25%	76.40%	79.17%
(90,24)	0.665625 M	80.02%	77.98%	78.77%
(90,8)	0.318425 M	90.44%	77.01%	79.28%
(90,1)	0.166525 M	95.00%	70.48%	70.98%

Table C-2: Results for the TT1-1 model when R_2 is varied while R_1 is fixed.

TT1-1 ranks	Parameters	Compression %	Test Acc	Max Acc
(80,95)	1.976925 M	40.67%	74.12%	79.14%
(76,82)	1.646745 M	50.58%	77.66%	79.38%
(70,70)	1.325025 M	60.23%	76.78%	79.33%
(59,60)	0.996085 M	70.10%	76.66%	79.63%
(46,46)	0.651105 M	80.46%	76.40%	79.58%
(28,28)	0.327105 M	90.18%	77.06%	79.47%
(8,8)	0.149505 M	95.51%	77.80%	78.10%
(1,1)	0.132705 M	96.01%	64.98%	64.98%

Table C-3: Results for the TT1-1 model when both R_1 and R_2 are varied.

R1 Rank	Max Acc%	R2 Rank	Max Acc%
90	78.47%	100	78.47%
76	79.30%	85	78.91%
62	79.30%	70	79.64%
49	80.01%	54	79.46%
35	79.78%	39	79.17%
21	79.76%	24	78.77%
7	79.54%	8	79.28%
1	78.36%	1	70.98%

Table C-4: Effects of ranks R_1 and R_2 on maximum test accuracy 'Max Acc %' of the TT1-1 model. The rank R_1 is changed from 90 to 1, when R_2 is fixed at 100. The rank R_2 is changed from 100 to 1, when R_1 is fixed at 90.

TT1-2 ranks	Parameters	Compression %	Test Acc	Max Acc
(100,78)	2.335585 M	29.90%	74.82%	79.16%
(84,78)	1.984545 M	40.44%	78.30%	78.30%
(69,78)	1.655445 M	50.32%	76.09%	79.28%
(54,78)	1.326345 M	60.19%	77.92%	79.54%
(38,78)	0.975305 M	70.73%	76.93%	79.24%
(23,78)	0.646205 M	80.60%	77.30%	79.85%
(8,78)	0.317105 M	90.48%	78.28%	78.94%
(1,78)	0.162525 M	95.09%	78.60%	78.60%

Table C-5: Results for the TT1-2 model when R_1 is varied while R_2 is fixed.

TT1-2 ranks	Parameters	Compression %	Test Acc	Max Acc
(100,78)	2.335585 M	29.90%	74.82%	79.16%
(100,66)	1.998145 M	40.03%	76.31%	79.44%
(100,54)	1.660705 M	50.16%	75.14%	79.07%
(100,42)	1.323265 M	60.28%	75.02%	79.26%
(100,30)	0.985825 M	70.41%	75.82%	79.22%
(100,18)	0.648385 M	80.54%	78.31%	78.88%
(100,6)	0.310945 M	90.66%	77.25%	77.66%
(100,1)	0.170345 M	94.88%	69.51%	69.51%

Table C-6: Results for the TT1-2 model when R_2 is varied while R_1 is fixed.

of the TT-ranks for the TT1-2 model.

The changes in maximum test accuracy with respect to change in ranks R_1 and R_2 in the TT1-2 model are reported in table C-8.

C-3 Results TT2-1 model

The following tables show the test accuracy after 10 epochs ('Test Acc') maximum test accuracy ('Max Acc'), the compression rate and the number of parameters for each combination of the TT-ranks for the TT2-1 model. Table C-13 shows the change in maximum test accuracy when R_3 and R_4 are varied.

TT1-2 ranks	Parameters	Compression %	Test Acc	Max Acc
(91,72)	1.984525 M	40.44%	76.55%	78.61%
(79,68)	1.652445 M	50.41%	77.83%	77.83%
(68,63)	1.326945 M	60.17%	76.32%	79.36%
(56,54)	0.991025 M	70.25%	72.50%	78.56%
(42,44)	0.659145 M	80.21%	77.28%	77.84%
(24,26)	0.312465 M	90.62%	78.66%	79.45%
(10,10)	0.162425 M	95.13%	78.27%	79.12%
(1,1)	0.132725 M	96.01%	61.61%	62.17%

Table C-7: Results for the TT1-2 model when both R_1 and R_2 are varied.

R1 Rank	Max Acc%	R2 Rank	Max Acc%
100	79.16%	78	79.16%
84	78.30%	66	79.44%
69	79.28%	54	79.07%
54	79.54%	42	79.26%
38	79.24%	30	79.22%
23	79.85%	18	78.88%
8	78.94%	6	77.66%
1	78.60%	1	69.51%

Table C-8: Effects of ranks R_1 and R_2 on the maximum test accuracy 'Max Acc %' of the TT1-2 model. The rank R_1 is changed from 100 to 1, when R_2 is fixed at 78. The rank R_2 is changed from 78 to 1, when R_1 is fixed at 100.

TT2-1 ranks	Parameters	Compression %	Test Acc	Max Acc
(10,140,350,10)	2.322025 M	30.31%	77.76%	78.02%
(10,116,350,10)	1.982665 M	40.50%	79.92%	79.92%
(10,92,350,10)	1.643305 M	50.68%	79.31%	80.14%
(10,68,350,10)	1.303945 M	60.86%	78.62%	79.54%
(10,45,350,10)	0.978725 M	70.62%	76.86%	79.92%
(10,22,350,10)	0.653505 M	80.38%	76.12%	79.89%
(10,1,350,10)	0.356565 M	89.29%	77.14%	77.15%

Table C-9: Results for the TT2-1 model when R_2 is varied while other ranks are constant.

TT2-1 ranks	Parameters	Compression %	Test Acc	Max Acc
(10,140,350,10)	2.322025 M	30.31%	77.76%	78.02%
(10,140,294,10)	1.974825 M	40.73%	78.38%	79.38%
(10,140,240,10)	1.640025M	50.78%	77.58%	80.39%
(10,140,188,10)	1.317625 M	60.45%	76.26%	78.72%
(10,140,136,10)	0.995225 M	70.13%	76.46%	79.25%
(10,140,82,10)	0.660425 M	80.18%	76.74%	80.00%
(10,140,28,10)	0.325625 M	90.22%	77.31%	79.18%
(10,140,1,10)	0.158225 M	95.25%	70.30%	70.30%

Table C-10: Results for the TT2-1 model when R_3 is varied while other ranks are constant.

TT2-1 ranks	Parameters	Compression %	Test Acc	Max Acc
(10,140,350,1)	2.132935 M	35.99%	77.10%	77.34%

Table C-11: Results for the TT2-1 model when R_4 is varied while other ranks are constant.

TT2-1 ranks	Parameters	Compression %	Test Acc	Max Acc
(10,130,314,10)	1.971825 M	40.82%	75.50%	77.48%
(10,120,278,10)	1.650425 M	50.47%	76.70%	78.85%
(10,110,236,10)	1.327825 M	60.15%	73.77%	73.77%
(10,100,182,10)	0.983625 M	70.45%	78.45%	79.34%
(10,90,120,10)	0.649025 M	80.52%	78.46%	79.24%
(10,60,60,10)	0.320825 M	90.37%	77.72%	79.39%
(10,1,2,10)	0.133845 M	95.98%	73.63%	73.63%

Table C-12: Results for the TT2-1 model when R_2 and R_3 are varied while other ranks are constant.

R2 Rank	Max Acc%	R3 Rank	Max Acc%
140	78.02%	294	79.38%
116	79.92%	240	80.39%
92	80.14%	188	78.72%
68	79.54%	136	79.25%
45	79.92%	82	80.00%
22	79.89%	28	79.18%
1	77.15%	1	70.30%

Table C-13: Effects of ranks R_2 and R_3 on the maximum test accuracy 'Max Acc %' of the TT2-1 model. The rank R_2 is changed from 140 to 1, when R_1, R_3 and R_4 are fixed at 10, 350 and 10 respectively. The rank R_3 is changed from 350 to 1, when R_1, R_2 and R_4 are fixed at 10, 140 and 10.

TT2-2 ranks	Parameters	Compression %	Test Acc	Max Acc
(10,148,200,10)	1.989865 M	40.28%	76.91%	79.60%
(10,120,200,10)	1.646025 M	50.60%	77.72%	79.63%
(10,93,200,10)	1.314465 M	60.55%	77.91%	79.63%
(10,66,200,10)	0.982905 M	70.50%	76.22%	79.33%
(10,39,200,10)	0.651345 M	80.45%	76.17%	80.06%
(10,12,200,10)	0.319785 M	90.40%	79.09%	79.62%
(10,1,200,10)	0.184705 M	94.45%	78.63%	78.64%

Table C-14: Results for the TT2-2 model when R_2 is varied while other ranks are constant.

TT2-2 ranks	Parameters	Compression %	Test Acc	Max Acc
(10,175,168,10)	1.979025 M	40.60%	77.36%	79.46%
(10,175,137,10)	1.647325 M	50.56%	77.47%	78.55%
(10,175,106,10)	1.315625 M	60.51%	75.63%	79.27%
(10,175,75,10)	0.983925 M	70.47%	78.48%	80.25%
(10,175,44,10)	0.652225 M	80.42%	76.74%	79.29%
(10,175,13,10)	0.320525 M	90.38%	76.88%	77.26%
(10,175,1,10)	0.192125 M	94.24%	73.90%	73.90%

Table C-15: Results for the TT2-2 model when R_3 is varied while other ranks are constant.

C-4 Results TT2-2 model

The following tables show the test accuracy after 10 epochs ('Test Acc') maximum test accuracy ('Max Acc'), the compression rate and the number of parameters for each combination of the TT-ranks for the TT2-2 model. Table C-18 shows the change in maximum test accuracy when R_3 and R_4 are varied in the TT2-2 model.

C-5 Results TT3-1 model

The following tables show the test accuracy after 10 epochs ('Test Acc') maximum test accuracy ('Max Acc'), the compression rate and the number of parameters for each combination of the TT-ranks for the TT3-1 model.

Table C-27 shows the change in maximum test accuracy when R_3 and R_4 are varied in the TT3-1 model.

TT2-2 ranks	Parameters	Compression %	Test Acc	Max Acc
(10,175,200,1)	2.285335 M	31.41%	75.50%	75.50%

Table C-16: Results for the TT2-2 model when R_4 is varied while other ranks are constant.

TT2-2 ranks	Parameters	Compression %	Test Acc	Max Acc
(10,165,178,10)	1.976425 M	40.68%	76.58%	78.64%
(10,155,155,10)	1.648325 M	50.53%	76.41%	78.38%
(10,138,135,10)	1.315865 M	60.51%	78.21%	78.98%
(10,116,115,10)	0.988305 M	70.34%	76.68%	79.13%
(10,90,90,10)	0.661625 M	80.14%	75.60%	79.31%
(10,52,52,10)	0.319625 M	90.40%	77.46%	79.59%
(10,2,2,10)	0.133625 M	95.98%	72.41%	72.74%

Table C-17: Results for the TT2-2 model when R_2 and R_3 are varied while other ranks are constant.

R2 Rank	Max Acc%	R3 Rank	Max Acc%
100	79.16%	78	79.16%
84	78.30%	66	79.44%
69	79.28%	54	79.07%
54	79.54%	42	79.26%
38	79.24%	30	79.22%
23	79.85%	18	78.88%
8	78.94%	6	77.66%
1	78.60%	1	69.51%

Table C-18: Effects of ranks R_2 and R_3 on the maximum test accuracy 'Max Acc %' of the TT2-2 model. The rank R_2 is changed from 175 to 1, when R_1, R_3 and R_4 are fixed at 10, 200 and 10 respectively. The rank R_3 is changed from 200 to 1, when R_1, R_2 and R_4 are fixed at 10, 175 and 10 respectively.

TT3-1 ranks	Parameters	Compression %	Test Acc	Max Acc
(10,100,337,500,56,14)	2.312857 M	30.59%	78.89%	79.52%
(10,100,278,500,56,14)	1.982457 M	40.50%	78.20%	79.66%
(10,100,220,500,56,14)	1.657657 M	50.25%	78.37%	79.49%
(10,100,160,500,56,14)	1.321657 M	60.33%	79.03%	79.03%
(10,100,100,500,56,14)	0.985657 M	70.42%	79.38%	80.12%
(10,100,40,500,56,14)	0.649657 M	80.50%	77.84%	77.84%
(10,100,1,500,56,14)	0.431257 M	87.05%	75.83%	75.83%

Table C-19: Results for the TT3-1 model when R_3 is varied while other ranks are constant.

TT3-1 ranks	Parameters	Compression %	Test Acc	Max Acc
(10,100,337,500,56,14)	2.312857 M	30.59%	78.89%	79.52%
(10,100,337,412,56,14)	1.967017 M	40.96%	77.92%	80.17%
((10,100,337,330,56,14)	1.644757 M	50.64%	77.62%	78.98%
(10,100,337,248,56,14)	1.322497 M	60.31%	78.97%	79.16%
(10,100,337,164,56,14)	0.992377 M	70.21%	77.50%	79.59%
(10,100,337,78,56,14)	0.654397 M	80.36%	79.16%	79.65%
(10,100,337,1,56,14)	0.351787 M	89.44%	75.69%	75.69%

Table C-20: Results for the TT3-1 model when R_4 is varied while other ranks are constant.

TT3-1 ranks	Parameters	Compression %	Test Acc	Max Acc
(10,100,337,500,1,14)	2.034777 M	38.93%	73.66%	73.66%

Table C-21: Results for the TT3-1 model when R_5 is varied while other ranks are constant.

TT3-1 ranks	Parameters	Compression %	Test Acc	Max Acc
(10,100,337,500,56,1)	2.309763 M	30.68%	75.90%	75.90%

Table C-22: Results for the TT3-1 model when R_6 is varied while other ranks are constant.

TT3-1 ranks	Parameters	Compression %	Test Acc	Max Acc
(10,50,337,442,56,14)	1.978817 M	40.61%	77.28%	79.32%
(10,50,337,360,56,14)	1.656557 M	50.28%	79.39%	79.83%
(10,50,337,274,56,14)	1.318577 M	60.42%	79.84%	80.18%
(10,50,337,190,56,14)	0.988457 M	70.33%	76.50%	79.51%
(10,50,337,106,56,14)	0.658337 M	80.24%	77.14%	79.45%
(10,30,337,30,56,14)	0.317217 M	90.48%	77.62%	79.50%
(10,1,337,1,56,14)	0.141709 M	95.74%	71.91%	71.91%

Table C-23: Results for the TT3-1 model when R_2 and R_4 are varied while other ranks are constant.

TT3-1 ranks	Parameters	Compression %	Test Acc	Max Acc
(10,100,310,450,56,14)	1.978657 M	40.62%	77.29%	79.28%
(10,100,300,370,56,14)	1.642857 M	50.69%	77.80%	79.81%
(10,100,280,300,56,14)	1.321657 M	60.33%	78.76%	79.64%
(10,100,240,230,56,14)	0.970457 M	70.87%	78.81%	79.80%
(10,100,180,160,56,14)	0.653257 M	80.39%	77.18%	79.04%
(10,100,80,90,56,14)	0.316057 M	90.51%	76.96%	79.41%
(10,100,5,5,56,14)	0.151707 M	95.44%	75.58%	75.58%

Table C-24: Results for the TT3-1 model when R_3 and R_4 are varied while other ranks are constant.

TT3-1 ranks	Parameters	Compression %	Test Acc	Max Acc
(10,50,320,460,56,14)	1.966257 M	40.99%	78.85%	79.44%
(10,50,300,400,56,14)	1.654657 M	50.34%	78.46%	80.23%
(10,50,265,340,56,14)	1.311557 M	60.64%	79.26%	79.78%
(10,50,225,280,56,14)	0.994957 M	70.14%	78.98%	78.98%
(10,50,190,180,56,14)	0.640457 M	80.77%	78.94%	78.94%
(10,50,150,90,56,14)	0.311057 M	90.66%	78.76%	79.74%
(10,16,16,16,56,14)	0.150313 M	95.48%	79.26%	79.42%

Table C-25: Results for the TT3-1 model when R_2, R_3 and R_4 are varied while other ranks are constant.

TT3-1 ranks	Parameters	Compression %	Test Acc	Max Acc
(4,6,12,12,12,10)	0.136437 M	95.90%	76.90%	77.10%

Table C-26: Results for the TT3-1 model when R_2, R_3, R_4 and R_5 are varied while other ranks are constant.

R3 Rank	Max Acc%	R4 Rank	Max Acc%
337	79.52%	500	79.52%
278	79.66%	412	80.17%
220	79.49%	330	78.98%
160	79.03%	248	79.16%
100	80.12%	164	79.59%
40	77.84%	78	79.65%
1	75.83%	1	75.69%

Table C-27: Effects of ranks R_3 and R_4 on the maximum test accuracy 'Max Acc %' of the TT3-1 model. The rank R_3 is changed from 337 to 1, when R_1, R_2, R_4, R_5 and R_6 are fixed at 10, 100, 500, 56 and 14, respectively. The rank R_4 is changed from 500 to 1, when R_1, R_2, R_3, R_5 and R_6 are fixed at 10, 100, 337, 56 and 14 respectively.

TT3-2 ranks	Parameters	Compression %	Test Acc	Max Acc
(10,40,400,418,84,14)	2.34297 M	30.24%	76.02%	79.55%
(10,40,324,418,84,14)	1.976217 M	40.69%	78.31%	78.34%
(10,40,250,418,84,14)	1.637297 M	50.86%	77.30%	79.94%
(10,40,178,418,84,14)	1.307537 M	60.75%	79.72%	80.36%
(10,40,106,418,84,14)	0.977777 M	70.65%	77.43%	79.95%
(10,40,34,418,84,14)	0.648017 M	80.55%	79.07%	79.07%
(10,40,1,418,84,14)	0.496877 M	85.08%	74.45%	74.45%

Table C-28: Results for the TT3-2 model when R_3 is varied while other ranks are constant.

C-6 Results TT3-2 model

The following tables show the test accuracy after 10 epochs ('Test Acc') maximum test accuracy ('Max Acc'), the compression rate and the number of parameters for each combination of the TT-ranks for the TT3-2 model.

Table C-34 shows the change in maximum test accuracy when R_3 and R_4 are varied in the TT3-2 model.

TT3-2 ranks	Parameters	Compression %	Test Acc	Max Acc
(10,40,400,350,84,14)	1.995177 M	40.12%	77.74%	80.14%
(10,40,400,280,84,14)	1.656377 M	50.29%	79.27%	79.46%
(10,40,400,210,84,14)	1.317577 M	60.45%	78.44%	78.44%
(10,40,400,140,84,14)	0.978777 M	70.62%	77.62%	80.12%
(10,40,400,72,84,14)	0.649657 M	80.79%	77.30%	77.30%
(10,40,400,2,84,14)	0.310857 M	90.67%	71.59%	71.59%

Table C-29: Results for the TT3-2 model when R_4 is varied while other ranks are constant.

TT3-2 ranks	Parameters	Compression %	Test Acc	Max Acc
(10,40,400,418,6,14)	1.991705 M	40.22%	77.71%	77.71%
(10,40,400,418,1,14)	1.970385 M	40.86%	74.37%	74.90%

Table C-30: Results for the TT3-2 model when R_5 is varied while other ranks are constant.

TT3-2 ranks	Parameters	Compression %	Test Acc	Max Acc
(10,40,400,418,84,1)	2.317563 M	30.44%	75.26%	75.53%

Table C-31: Results for the TT3-2 model when R_6 is varied while other ranks are constant.

TT3-2 ranks	Parameters	Compression %	Test Acc	Max Acc
(10,40,370,370,84,14)	1.968977 M	40.91%	78.22%	79.38%
(10,40,330,328,84,14)	1.631097 M	51.05%	77.98%	79.98%
(10,40,290,282,84,14)	1.311857 M	60.63%	79.03%	80.00%
(10,40,242,232,84,14)	0.994297 M	70.16%	78.45%	79.14%
(10,40,172,182,84,14)	0.675897 M	79.71%	78.45%	78.45%
(10,40,90,85,84,14)	0.325077 M	90.24%	77.62%	79.73%
(10,40,16,16,84,14)	0.163577 M	95.09%	79.01%	79.23%
(10,40,1,1,84,14)	0.142427 M	95.72%	72.01%	72.01%

Table C-32: Results for the TT3-2 model when R_3 and R_4 are varied while other ranks are constant.

TT3-2 ranks	Parameters	Compression %	Test Acc	Max Acc
(10,40,365,370,6,14)	1.653325 M	50.38%	77.86%	79.09%
(10,40,326,320,6,14)	1.327425 M	60.16%	77.75%	77.75%
(10,40,260,270,16,14)	0.984665 M	70.45%	77.06%	79.23%
(10,40,190,210,16,14)	0.644065 M	80.67%	78.70%	80.34%
(10,40,100,125,16,14)	0.320465 M	90.38%	79.04%	79.75%
(10,40,16,16,16,14)	0.142809 M	95.71%	77.94%	77.94%

Table C-33: Results for the TT3-2 model when R_3, R_4 and R_5 are varied while other ranks are constant.

R3 Rank	Max Acc%	R4 Rank	Max Acc%
324	78.34%	350	80.14%
250	79.94%	280	79.46%
178	80.36%	210	78.44%
106	79.95%	140	80.12%
34	79.07%	72	77.30%
1	74.45%	2	71.59%

Table C-34: Effects of ranks R_3 and R_4 on the maximum test accuracy 'Max Acc %' of the TT3-1 model. The rank R_3 is changed from 400 to 1, when R_1, R_2, R_4, R_5 and R_6 are fixed at 10, 40, 418, 84 and 14, respectively. The rank R_4 is changed from 418 to 1, when R_1, R_2, R_3, R_5 and R_6 are fixed at 10, 40, 400, 84 and 14 respectively.

Bibliography

[1] Dan Klein Adam Pauls. Faster and smaller n-gram language models. *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference*, 2011.

[2] Peter T. Pham Andrew L. Maas, Raymond E. Daly. Learning word vectors for sentiment analysis. *Association for Computational Linguistics*, 2011.

[3] Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. *International Conference on Machine Learning.*, 2018.

[4] Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. A neural probabilistic language model. *The Journal of Machine Learning Research*, 2003.

[5] Cong Chen, Kim Batselier, Ching-Yun Ko, and Ngai Wong. A support tensor train machine. *2019 International Joint Conference on Neural Networks (IJCNN)*, 2019.

[6] Nadav Cohen, Or Sharir, and Amnon Shashua. On the expressive power of deep learning: A tensor analysis. *Journal of Machine Learning Research.*, 2016.

[7] Ronald J. Williams D. Rumelhart, Geoffrey E. Hinton. Learning representations by back-propagating errors. *Computer Science*, 1986.

[8] Jiang Qian Dan Li. Text sentiment analysis based on long short-term memory. *International Conference on Computer Communication and the Internet*, 2016.

[9] Michael I. Jordan David M.Blei, Andrew Y. Ng. Latent dirichlet allocation. *Journal of Machine Learning Research 3*, 2003.

[10] Susan T. Dumais. Latent semantic analysis. *Annual Review of Information Science and Technology*, 2004.

[11] Jack FitzGerald, Shankar Ananthakrishnan, Konstantine Arkoudas, Davide Bernardi, Abhishek Bhagia, Claudio Delli Bovi, Jin Cao, RAKESH CHADA, Amit Chauhan,

Luoxin Chen, Anurag Dwarakanath, Satyam Dwivedi, Turan Gojayev, Karthik Gopalakrishnan, Thomas Gueudre, Dilek Hakkani-Tur, Wael Hamza, Jonathan Hueser, Kevin Martin Jose, Haidar Khan, Beiye Liu, Jianhua Lu, Alessandro Manzotti, Pradeep Natarajan, Karolina Owczarzak, Gokmen Oz, Enrico Palumbo, Charith Peris, Chandana Satya Prakash, Stephen Rawls, Andy Rosenbaum, Anjali Shenoy, Saleh Soltan, Mukund Harakere, Liz Tan, Fabian Triefenbach, Pan Wei, Haiyang Yu, Shuai Zheng, Gokhan Tur, and Prem Natarajan. Alexa teacher model: Pretraining and distilling multi-billion-parameter encoders for natural language understanding systems. *KDD 2022*, 2022.

[12] Ivan Glasser, Ryan Sweke, Nicola Pancotti, Jens Eisert, and Ignacio Cirac. Expressive power of tensor-network factorizations for probabilistic modeling. *Advances in neural information processing systems*, 2019.

[13] Joel Greeenberg. The enigma machine, the turing guide. *Oxford University Press*, 2017.

[14] Masato Hagiwara. *Real-World Natural Language Processing: Practical Applications with Deep Learning*. Simon and Schuster, 2021.

[15] GE Hinton, JL McClelland, and DE Rumelhart. Distributed representations. *The Philosophy of Artificial Intelligence*, 1986.

[16] Frank L Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 1927.

[17] Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural Computation*, 1997.

[18] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 1997.

[19] Oleksii Hrinchuk, Valentin Khrulkov, Leyla Mirvakhabova, Elena Orlova, and Ivan Oseledets. Tensorized embedding layers. *Association for Computational Linguistics*, 2020.

[20] James Martens Ilya Sutskever. Generating text with recurrent neural networks. *Proceedings of the 28th International Conference on Machine Learning, ICML*, 2011.

[21] Karen Sparck Jones. Natural language processing: A historical review. *Current Issues in Computational Linguistics: In Honour of Don Walker*, 1994.

[22] Valentin Khrulkov, Alexander Novikov, and Ivan Oseledets. Expressive power of recurrent neural networks. *International Conference on Learning Representations.*, 2018.

[23] Jingling Li, Yanchao Sun, Jiahao Su, Taiji Suzuki, and Furong Huang. Understanding generalization in deep learning via tensor methods. *International Conference on Artificial Intelligence and Statistics*, 2020.

[24] Elizabeth D Liddy. Natural language processing. *Encyclopedia of Library and Information Science, 2nd Edition*, 2001.

[25] Xindian Ma, Peng Zhang, Shuai Zhang, Nan Duan, Yuexian Hou, Ming Zhou, and Dawei Song. A tensorized transformer for language modeling. *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS 32 (NIPS 2019)*, 2019.

[26] Mark Neumann Matthew E. Peters. Deep contextualized word representations. *Association for Computational Linguistics*, 2018.

[27] A. Moody Md. Zahangir Alom. Effective quantization approaches for recurrent neural networks. *International Joint Conference on Neural Networks (IJCNN)*, 2018.

[28] Eva Memmel, Clara Menzen, Jetze Schuurmans, Frederiek Wesel, and Kim Batselier. Towards green ai with tensor networks–sustainability and innovation enabled by efficient algorithms. *arXiv preprint arXiv:2205.12961*, 2022.

[29] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *Proceedings of the International Conference on Learning Representations*, 2013.

[30] Bing Liu Nitin Jindal. Review spam detection. *WWW '07: Proceedings of the 16th international conference on World Wide Web*, 2007.

[31] Alexander Novikov, Dmitrii Podoprikhin, Anton Osokin, and Dmitry P Vetrov. Tensorizing neural networks. *NIPS*, 2015.

[32] Ivan V Oseledets. Tensor-train decomposition. *Society for Industrial and Applied Mathematics*, 2011.

[33] Roger Penrose. Applications of negative dimensional tensors. *Combinatorial mathematics and its applications*, 1971.

[34] Tao Qin. Deep learning basics. *In: Dual Learning, Springer*, 2020.

[35] Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. *ICML'17: Proceedings of the 34th International Conference on Machine Learning*, 2017.

[36] Robel Daniel Robin Cheong. transformers.zip : Compressing transformers with pruning and quantization. 2019.

[37] Xin Rong. word2vec parameter learning explained. *arXiv*, 2014.

[38] Jeff Pool Song Han and John Tran. Learning both weights and connections for efficient neural networks. *NIPS'15: Proceedings of the 28th International Conference on Neural Information Processing Systems*, 2015.

[39] W. Dally Song Han, Huizi Mao. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *ICLR*, 2016.

[40] Y. Choe Taehyeon Kim, Jieun Lee. Tensor train decomposition for efficient memory saving in perceptual feature-maps. *International Conference on Computer Vision Workshop (ICCVW)*, 2019.

[41] Brett W. Bader Tamara G. Kolda. Tensor decompositions and applications. *Society for Industrial and Applied Mathematics*, 2009.

[42] Alan Turing. Computing machinery and intelligence. *Oxford Academic - Mind Volume LIX*, 1950.

[43] John Liu Uday Kamath. Basics of deep learning. *In: Deep Learning for NLP and Speech Recognition, Springer*, 2019.

[44] Yuli Vasiliev. Natural language processing with python and spacy: A practical introduction. *No Starch Press*, 2020.

[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, 2017.

[46] Liaqat Ali Yakubu Imrana, Yanping Xiang. A bidirectional lstm deep learning approach for intrusion detection. *Expert Systems with Applications, Volume 185*, 2021.