A comprehensive evaluation of stacked autoencoders for text embedding

MSc Research Project
MSc. in Data Analytics

Sarthak Bhatnagar
Student ID: X21185352

School of Computing
National College of Ireland

Supervisor: Dr. Giovani Estrada

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Sarthak Bhatnagar		
Student ID:	X21185352		
Programme:	MSc. in Data Analytics	**Year:**	2022-23
Module:	Research in Computing		
Supervisor:	Dr. Giovani Estrada		
Submission Due Date:	18-09-2023		
Project Title:	A comprehensive evaluation of stacked autoencoders for text embedding.		
Word Count:	8492	**Page Count:**	25

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Sarthak Bhatnagar

Date: 16-09-2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

A comprehensive evaluation of stacked autoencoders for text embedding

Sarthak Bhatnagar
X21185352

Abstract

Artificial Intelligence is significantly expanding in the current age of Information Technology, with many people utilizing the potential of AI to make machines intelligent enough to perform repetitive tasks creatively and more simply. With the advancement of technologies, AI research, and development have narrowed to two research areas: Machine Learning and Deep Learning. Machine learning algorithms are trained on data to learn how to perform a task. However, they are stringent for domain knowledge, while deep learning (a subset of machine learning) has an excellent capability to achieve flexibility in computational tasks and accuracy and offers to learn through neural network architecture. In this research paper, we examine the effect of a stacked autoencoder architecture on text embedding by employing Bidirectional Long Short-Term Memory (BiLSTM) and Bidirectional Gated Recurrent Unit (BiGRU) models. The stacked autoencoder architecture can capture complex patterns and parameters within the dataset due to multiple layers and Bidirectional LSTMs and Bidirectional GRUs, known for their capability to capture context from both past and future sequences, are used as components to study autoencoder architectures. By leveraging the strengths of both, we will show that there is indeed a trade-off between model complexity and accuracy using layered architecture. Our results showcased that a three-layered bidirectional GRU autoencoder has the best accuracy. Moreover, the higher number of layers has a negligible impact on the accuracy, while potentially taking more computing resources.

1 Introduction

The research area of artificial intelligence has proved to be a boon in the big data era. Artificial intelligence is at the core of computer and human interaction. The motivation behind creating AI technologies is their ability to make machines think like human beings, mimic their behaviour, and perform tasks as humans do. AI-integrated machines learn from their surroundings, perform specific tasks, and aim to maximize the probability of success in each problem. Many researchers are utilizing AI techniques to solve real-world, domain-specific problems that started with conventional approaches, ranging from automating repetitive human tasks to optimization techniques and much more. In the research field of AI, Machine learning is a subset of AI techniques that enable computers to learn from historical data and apply what they learn by making intelligent decisions (Simon, 2013). The benefit of utilizing machine approaches is its ability to extract the significant features out of the data while training (LeCun, Bengio and Hinton, 2015). Moreover, the machine learning techniques are classified into three approaches: supervised, unsupervised, and reinforcement learning methods. Supervised learning is about training the model on a labelled dataset to make predictions. There are several algorithms that use this type of learning, such as SVM (Support Vector Machine), etc. Supervised learning, being powerful, faced some challenges, including overfitting, biases, and difficulty scaling the data quality. Unsupervised learning is another approach to learning in which the algorithms are asked to find out the hidden patterns and significant features based on unlabelled data. The algorithms used in the learning are aimed to capture the relationships in the data, clusters, and characteristics of the data. These types of techniques are basically used for clustering, dimensionality reduction, etc. (Thomas and Gupta, 2020) Last, but not least, the Semi-supervised learning approach which possesses a combined method of both the other approaches, the algorithms in this approach are trained on a small amount of labelled dataset and a large amount of unlabelled dataset so that the training model will utilize the labeled data to improve and its accuracy and aim to uncover the hidden patterns to predict the unseen data.

Traditional machine learning approaches are a broad term covering various methods and algorithms to make predictions. Although machine learning seems complex, these stringent techniques require human intervention and domain expertise. To overcome the problems associated with machine learning, the concept of deep learning is introduced, which promises excellent flexibility and power by learning the hierarchy concepts, which can be further defined to achieve more straightforward tasks with more accuracy and less human intervention. Deep learning is a subset of machine learning that comes under the unsupervised learning category and works on multi-layer structures. The multi-layer structures have several benefits over single layers as they can comprehend complex and non-linear relationships of the input data. Moreover, the multi-layers networks generalize the data effectively and utilize their ability to extract abstract features even from small datasets (Li, Pei and Li, 2023). Each layer in a structure performs low-level to high-level feature extraction, calling it a feature selection process. Among the myriad of deep learning techniques, autoencoders (Rumelhart, Hinton and Williams, 1985) have played a crucial role in feature extraction, dimensionality reduction, and data representation. A detailed introduction to autoencoders is given by (Bourlard and Kamp, 1988). It is a specific type of neural network that promises the potential to convert raw data into compact and meaningful representations. As a result, they can aid in the understanding and manipulation of complex data. Autoencoders belong to a category of unsupervised machine-learning models that aim to learn compressed versions of representations of input data by transforming it into a latent space with reduced dimensions. The principal architecture of autoencoders involves two key components: an encoder and a decoder. The encoder's job is to compress and encode the input data, also known as the encoding phase. As the name suggests, it transforms or maps the input data with low-dimensional latent space or extracts meaningful features from input data. The purpose of the decoder is to reconstruct the original input data from the encoded representations, and it is known as the decoding phase. This process of encoding and decoding techniques in the model facilitates the acquisition of salient characteristics present in the input data which is why autoencoders is a valuable model architecture in deep learning for various tasks like dimensionality reduction and feature extraction.

One of the remarkable aspects of autoencoders is their ability to learn efficient data representations without requiring explicit labelling. These characteristics have gained significant attention in the big data era. By utilizing the potential of deep neural networks, autoencoders can uncover hidden patterns and relationships within the data, which can be the basis of improved decision-making in domain-specific tasks such as sentiment analysis and anomaly detection. Autoencoders have been employed not just for the purpose of reducing dimensionality and representing data, but they have also gained popularity as generative models. Variational Autoencoders (Pinheiro Cinelli et al., 2021) and Generative Adversarial Networks (Goodfellow et al., 2014) are the techniques that work on the principle of autoencoder architectures and elevate the power and creativity of AI to a new level. These generative models have been employed in many domain specific platforms such as image synthesis, drug discovery, etc. due to their ability to learn and acquire knowledge from the underlying probability distribution of data. The primary objective of this thesis is to examine the multifaceted domain of deep learning, with a specific focus on autoencoders. The exploration commences with an examination of the theoretical concepts and basic frameworks of autoencoders, in order to establish a robust groundwork for our subsequent inquiry. We undertake a critical experiment on the stack-layered architecture of autoencoders utilizing Bidirectional-LSTMs and Bidirectional-GRUs. The multiple hidden layers in autoencoders allow us to grasp the complex features of the input data and contribute to improving the outcomes (Bengio, 2009). However, some challenges are encountered due to multiple hidden layers while training the data, increasing training time and model complexity. This experimental study aims to examine the effect of the stack-layered architecture of autoencoder in text embedding using the Bidirectional LSTMs and Bidirectional GRUs as a combination.

Following that, we have mentioned the literature study of research papers in the context to the topic. In Section 3, set the base for suitable methodology. Then Section 4 discusses design requirements and specifications, and Section 5 discusses how the model was implemented. Subsequently, the Result and evaluation obtained related to the model are covered, and in the end, the Conclusion and suggestions for future work is given.

Research Question

The research questions originated from the lack of detailed studies on the trade-offs between parametric complexity and accuracy in stack-layered autoencoders. The specific questions we want to address are:

Question 1: What effect do Bi-LSTM and Bi-GRU layers have on the autoencoders for Text Embedding?

Question 2: What effect does stacking layers have on the accuracy of autoencoders for Text Embedding?

2 Related Work

The essence of any scholarly endeavour lies in including a literature review, as it provides an all-encompassing assessment of prior research in the field. It briefly outlines the problem, the ongoing investigations, and the relevant theories to explore. In the context of text summarization, extensive studies have been undertaken in the past, delving into diverse NLP-based methods for textual summarization. This study delves into a comprehensive evaluation of various algorithmic models, focusing on exploring the impact of Deep Learning techniques in this domain.

2.1 A Historical Journey through Autoencoders in Literature

The concept of Autoencoders was first introduced in the "Learning Internal Representation by back propagating error" research paper, which provides simple two-layer architecture networks. The architecture has two input layers directly mapped to patterns in the output layer, with no hidden units involved. The author (Rumelhart, Hinton and Williams, 1985) introduced the backpropagation algorithm which is a technique to automatically adjust the weights of neural networks using gradient descent optimization and the delta rule. But the author mentioned that the loss function can be reduced (namely, the problem of local maxima or minima) but could not explicitly guarantee the execution period. Furthermore, a proper continuous, non-linear activation function for experiments is carried out, as the linear threshold function is discontinuous and hence does not suffice for delta generalized rules. A pioneering role in demonstrating the significant use of autoencoders in dimensionality reduction in neural networks and deep learning is studied by (Hinton and Salakhutdinov, 2006) The author utilized the efficient adaptive, multilayer encoder network of autoencoders to convert high-dimensional data to low-dimensional code and reconstruct it with the help of a decoder. Facing the problem of optimizing the weights of a non-linear autoencoder, the author introduced a "pretraining" procedure for binary data for the hidden units and generalized it to real-valued data. In contrast, linear units replace the visible units with Gaussian noise, which allows low-dimensional code to make good use of continuous variables to facilitate comparisons with PCA. The pre-trained algorithms allowed us to fine-tune networks and minimize the cross-entropy error. The autoencoder consists of layer size (28*28) with a symmetric decoder trained on 20,000 images and tested on 10,000 new photos. This showed that pretraining improved the model architecture and outperformed PCA as it has lower reconstruction loss in less training time.

The non-linear functions at the hidden layer of the autoencoder are not necessary, as proposed by (Bourlard and Kamp, 1988). Removing non-linear functions does not affect the performance of the autoencoder network; even linear output weight values can be calculated using standard linear algebra techniques such as singular value decomposition (SVD). This finding was proven to be the discovery of its time, as training the autoencoders with a large number of units in the hidden layer is pretty expensive. Also, the optimal weight values for autoencoders could be achieved by minimizing the mean squared error between the input and output of the network. Moreover, it suggests that it is an efficient alternative technique for error backpropagation algorithms but does not directly evaluate the methods.

To build a deep neural network based on Stack-layered architecture for denoising autoencoders, a new method is proposed by (Vincent et al., 2010), which yields lower classification errors bridging the gap utilizing the concept of a deep belief network. The author aims to introduce a straightforward denoising autoencoder architecture trained to reconstruct the input from the corrupted version of it. This is done by first corrupting the input data using the masking noise, and geometric mapping between input and corrupted inputs is achieved using the stochastic operator's so-called manifold assumptions. Several techniques are compared for the Classification performance problem in which stack-layered denoising autoencoder architecture outperformed SVM, ordinary autoencoder pretraining (SAE) architecture.

2.2 A Critical survey of Techniques in Text Embedding

A lot of significant change has come to deep learning through its evolution and Artificial Intelligence. The progress observed in recent years in deep learning techniques, namely in the domain of text analysis and Natural Language Processing, is groundbreaking due to the emergence and improvement of neural network structures. These Structures comprised stacked autoencoders, LSTM networks, bidirectional LSTM networks, convolutional neural networks (CNNs), and bidirectional GRU. The structure of these architectures has been specifically designed to tackle numerous obstacles of different kinds in text embedding and sequence-to-sequence learning. This literature review section examines and analyzes various approaches the researchers utilize in developing efficient hierarchical text embedding through stack-layered autoencoders.

The multilayered deep Long Short-Term Memory (LSTM) outperformed shallow LSTM in the study proposed by (Sutskever, Vinyals and Le, 2014). The model consists of two main components: an encoder LSTM, which reads the input sequences and generates the fixed dimensional vector, and a decoder LSTM, which uses the vector to create the output sequences, trained to maximize the conditional probability. The author trained the dataset on the WMT'14 English to French dataset, which consists of 12 million sentences[1]. The possible reason behind the high performance of multilayered LSTM could be a large number of hidden layers and (input, forget, and output) gates that control the flow of information in each cell. The input and forget gates utilized the sigmoid and tanh functions, respectively. Moreover, the output from the last LSTM is fed into the naive softmax layer to generate the outcome. We implemented the same approach in our work, but it didn't work well with the dataset we have. However,

[1] https://www.kaggle.com/datasets/dhruvildave/en-fr-translation-dataset

regarding limitations, the author assesses the model evaluation of only one language pair and utilizes naive softmax, which is computationally expensive.

A novel approach to Bidirectional LSTM with Conditional Random Field (CRF) is proposed by (Huang, Xu and Yu, 2015) for sequence tagging. Various LSTM-based models are utilized, including LSTM networks, Bidirectional LSTM, and BiLSTM-CRF. The BiLSTM-CRF technique outperformed the other two approaches and achieved great accuracy (close to SOTA techniques) on POS, chunking, and NER datasets. The novel approach of BiLSTM-CRF utilizes the power of BiLSTM (efficiently using both past and future input features) and CRF (sentence-level tag information). In the model's architecture, the LSTM layer processes the input sequence and generates the hidden state sequence fed to the CRF layer; then, it models the dependencies between the adjacent tags to predict the probable tags for the input sequences.

In this proposed study, the authors presented an LSTM (Long Short-Term Memory) Encoder-Decoder model for text simplification. The model comprises an encoder that generates fixed-length sequences from the input vector representations and a decoder that decodes and generates output sequences by predicting the following words based on given inputs and vector representations. Moreover, (Wang et al., 2016) also utilized attention-based LSTM to generate sequences that focus on the relevant part of the input sequences. Both models are trained and evaluated for text simplification tasks. The evaluation is based on the model's ability to generate simpler sentences from their complex counterparts. The results showed that the Attention-based LSTM Encoder-Decoder model outperformed the basic LSTM Encoder-Decoder model and other existing methods for text simplification tasks. However, The author mentioned several limitations to the proposed experiment, such as the dataset's quality and the model needing help with sentences with syntactically complex structures.

A novel approach introduced in this paper for text representation and (Zhang, Liu and Song, 2018) proposed a new LSTM structure for encoding text named Sentence-LSTM (S-LSTM) and argued that the traditional bidirectional LSTM has limitations in capturing long-term dependencies and contextual information exchange due to sequential which we have observed in our stacked-layer architecture experiment. The S-LSTM addresses these limitations by introducing the concept of a sentence-state vector, which captures the global context of the sentence. The sentence vector updates each timestep, relying on the input sequence and previous hidden states. The author evaluated the novel approach based on accuracy, F1-Score, and even training time and compared it with different architectures such as Bi-directional LSTMs, CNNs, Bi-directional LSTMs+Attention mechanism, and Transformers. Moreover, The S-LSTM is trained and evaluated in several benchmark datasets, such as Sentiment Analysis, NER, and POS tagging, and the model outperformed all of State-Of-The-Art models and achieved a reward in the field of Text Representation.

Improved and effective Variational Autoencoders (VAEs) for text modeling are proposed in the study by (Yang et al., 2017). The author introduced a new decoder architecture that uses dilated convolutions to increase its contextual capacity while effectively using encoding information and mentioned that dilated convolutions and the use of Gaussian prior for the latent variable are two significant components of their model. The author evaluated the models on two datasets, Yahoo Answer and Yelp 15 reviews, and highlighted the model's potential for semi-supervised and unsupervised labeling tasks. The author's model evaluation is based on perplexity for language modeling in which dilated convolutional VAE outperformed standard LSTM models. Moreover, several limitations are also discussed, such as the model is unsuitable for the generation of long sequences, not computationally efficient as it requires a large number of parameters, and is slow to train.

In comparison to the performance of LSTM, GRU, and Bidirectional RNN deep learning models for generating new conversations and scenario-based scripts, LSTM performed the best, then GRU and bidirectional RNN in the end. The study proposed by (Mangal, Joshi and Modak, 2019) uses the script of famous TV series, dialogues, and descriptions of the dialog to train the models, with the motive that the trained model will generate new scripts and will assist writers in developing new content. The author divided the architecture into five modules for each model: an embedding layer, a neural layer, a dropout layer, a dense layer, and an output layer. The embedding layer maps the vocabulary with 256 units; the neural layer can be either LSTM, GRU, or Bi-RNN and could be any combinations such as bi, tri, or quad layers. The dropout layer generalizes the learning process and prevents overfitting, and the dense layer connects all the neurons and produces the output from the layers which generate the text. On evaluation, the LSTM performed better in developing text in the least time than GRU and bidirectional RNN.

The paper proposes a unified structure of neural networks to enhance the accuracy of text classification by combining the techniques of word embedding and Gated Recurrent network. (Zulqarnain et al., 2019) claims that this approach outperformed other RNN approaches and can help organizations manage and exploit meaningful information from large amounts of online data. The GRU model is a Recurrent neural network capable of processing sequential data over its RNN architecture, and it is designed in a way that overcomes the issue of vanishing gradient that existed in other RNNs. In this approach, the online posts were first converted into vector representation with the help of the word embedding technique and later fed to GRU to extract the contextual semantics between words. This approach is compared with other RNNS: Recursive Neural Networks, Matrix-Vector Recurrent Neural Networks (MV-RNN), and Long Short Term Memory (LSTM), in which they observed that GRUs are effective in the Text classification task. Moreover, the author utilized the dropout strategy and L2 regularization to deal with the problem of overfitting.

The paper study about the learning effect of different hidden layers in stacked autoencoders. (Xu et al., 2016) utilizes the MNIST dataset to train the model on 60,000 images and to test the algorithm with 10,000 examples. Each image has a dimension of 28*28 pixels which converts into 784 numbers. The author introduces stacked autoencoders, a neural network consisting of multiple layers in the encoder and decoder. The stacked autoencoder is trained layer by layer, with each layer learning a new representation of the data. The author evaluated the stacked autoencoder using training error and validation error in which training error measures the distance between the predicted out and the actual out of the training data. For validation error, it measures the difference between the predicted and actual output of the validation set. In conclusion, the author tests the effect of different hidden layers of numbers on the learning capability of neural networks, and the result showed that increasing the number of hidden layers can improve the performance of the autoencoder up to a certain point, beyond a certain number of hidden layers it starts to degrade.

The research study proposes a new method for learning mapping in the embedding space of an autoencoder which allows the generation of long and complex sentences called Bag-of-Vectors Autoencoders (BoV-AEs). BoV-AEs encode the text into a variable-size bag of vectors that grows with the size of the text, which is similar to the attention-mechanism method, and proposed a novel regularization technique that helps in learning meaningful patterns in the latent space. (Mai and Henderson, 2021) evaluated the effectiveness of BoV-AEs on three unsupervised learning tasks, such as style transfer, sentiment transfer, and sentence summarization with different datasets. The proposed method is compared based on the tasks with standard autoencoder, fixed-size autoencoder, and seq-to-seq model and outperformed previous methods on all three tasks.

A comparative study on classic and contextualized word embeddings in deep learning was conducted by (Wang, Nulty and Lillis, 2020) for text classification tasks. The author utilized various datasets to train the models (word2vec, GloVe, and FastText) for classic word embeddings and models (ELMo and BERT) for contextualized embeddings. Moreover, the Bidirectional LSTM is used as a downstream encoder in contextualized word embedding on SST-2 dataset. Similarly, CNN is also used in classic word embeddings in AAPD, Reuters, and 20NewsGroup datasets. According to the neural network architectural viewpoint, the embedding layer passes the text representation to the downstream encoder, which could be either Bidirectional LSTM or CNN; the Bidirectional LSTM processes the representation forward and backward to seize the contextual information, whereas CNN applies a series of convolutional filters to capture the essential local features. The following fully connected layer maps the input sequence with fixed vector representations. It passes it to the classification layer, which could be either a single softmax or multi-label sigmoid layer to predict the label of input text. Overall, the study utilized different architectures and evaluated the models' performances on various datasets to learn the effective representations of text in classification tasks. It was found that (ELMo and BERT) performed better than (word2vec, GloVe, and FastText) on SST-2 dataset. However, the result is vice versa on the rest of the datasets.

With the motive to understand pattern recognition for adaptive user interfaces, (Umer et al., 2023) proposed a study focussed explicitly on text classification tasks using machine learning and deep learning models. The author investigated the impact of convolutional neural network (CNN) and FastText embedding and compared it with machine learning approaches, including random forest, logistic regression, extra tree classifier, gradient boosting machine, and stochastic gradient descent. Several preprocessing steps are performed on every dataset which deals with missing or inconsistent values, and stops word is removed. Then tokenized and converted to vector representations. FastText is a popular word embedding technique used to extract high-quality features represented in high-dimensional space. These vector representations are passed to the dropout layer to prevent over-fitting and passed to a 3-layered convolutional neural network. These layers will slide multiple filters on input data along with convolutional operations to generate feature maps. The generated maps are then passed through the max-pooling layer for dimensionality reduction, and the maps will be moved further to fully connected layers which predict the output. The results showed that FastText embedding with the CNN approach is potentially effective and surpasses all other machine-learning models.

Table 1: Summary of Related work.

Reference	Work
(Rumelhart, Hinton and Williams, 1985)	Learning Representations by back Propagating error.
(Hinton and Salakhutdinov, 2006)	Dimensionality reduction techniques.
(Bourlard and Kamp, 1988)	Standard Autoencoders.
(Vincent et al., 2010)	Standard Stacked Autoencoders.
(Sutskever, Vinyals and Le, 2014)	Multilayered LSTM, Shallow LSTM
(Huang, Xu and Yu, 2015)	Bi-LSTM with CRF. LSTM
(Wang et al., 2016)	LSTM, LSTM+Attention Mechanism
(Zhang, Liu and Song, 2018)	LSTM, Bi-LSTM, Bi-LSTM+Attention, CNN
(Yang et al., 2017)	CNN-VAE, LSTM.
(Mangal, Joshi and Modak, 2019)	LSTM, GRU, Bidirectional RNN.
(Zulqarnain et al., 2019)	GRU, RNN, MV-RNN, LSTM.
(Xu et al., 2016)	Stacked Autoencoders.
(Mai and Henderson, 2021)	Bag-of-Vectors Autoencoders.
(Wang, Nulty and Lillis, 2020)	CNN, Bidirectional LSTM, FastText, GloVE.
(Umer et al., 2023)	FastText with CNN model.

3 Research Methodology

The section research methodology is a systematic and structured approach used in the experiment of hierarchical text embedding using stack-layered autoencoders. The selection of the most suitable data mining approach is a crucial part of crafting a model to generate precise and accurate results. With the careful examination of various data mining techniques, we have selected the KDD (Knowledge Discovery in Databases) approach, which starts with data collection, preparation, and preprocessing, generating meaningful insights from the massive corpora, and model development. The rationale judgement behind selecting this approach for this study is derived from its comprehensive framework, which allows a detailed explanation of each component. Figure 1 depicts thorough sequence-wise steps involved in the process.

3.1 Data Collection and Data Description

For the purpose of data collection in this study, the well-known Brown Corpus from the Natural Language Toolkit (NLTK)[2] library was used to collect data. The Brown Corpus is a crucial dataset that has been around since the 1960s and is extensively utilized in the field of natural processing language. It is the first million words corpora in the English language, which was collected electronically. The corpus consists of 500 samples and is categorized into 15 genres, including politics, books, sports, culture, and even news, which make up the corpus as a whole. In this research project, the Brown Corpus is used as a foundational dataset for hierarchical text embedding techniques utilizing autoencoders.

3.2 Data Preprocessing and Visualizations

In this study, comprehensive data processing was performed to prepare the Brown Corpus, which is a collection of sentences in the English language. The corpora are rich textual data categorized into 15 types of various genres and are utilized for subsequent analysis and text embedding using stack-layered autoencoders. Initially, the Brown Corpus was assessed using the Natural Language Toolkit (NLTK) library, which is a powerful and famous Python library specially designed to work with human language data in the field of Natural Language Processing. To gain insights into the language and frequency distribution from the Brown corpus, The text was tokenized into words, and the frequency distribution of the words in the dataset was calculated to reveal the top 30 most common words, which shows the prominence of specific words in the dataset.

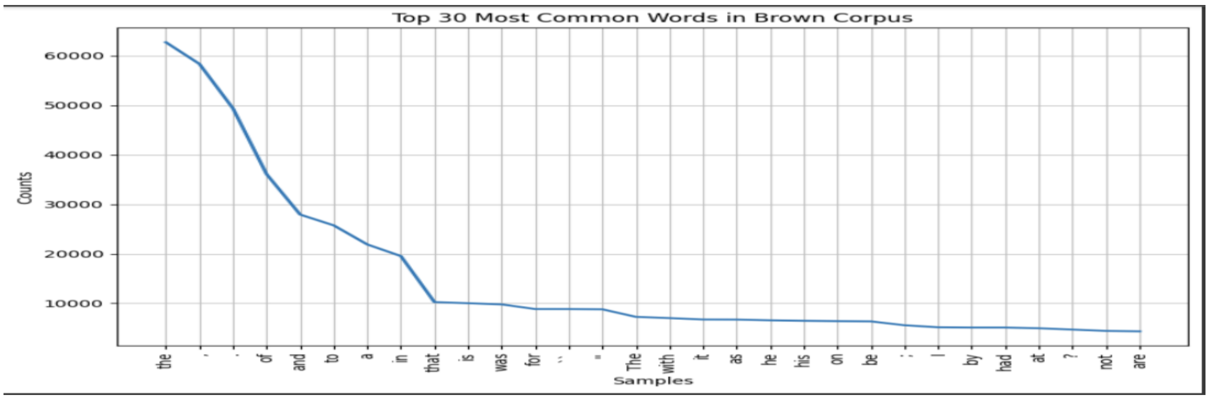

Fig.1: Top 30 common words in the Brown corpora.

POS tagging or Part-of-Speech tagging is a crucial step of data preprocessing in the domain of Natural Language Processing, which involves assigning a grammatical category to each word in a given sentence. The words in the Brown corpus are analyzed and assigned POS tags to

[2] https://www.nltk.org/book/ch02.html

each word. The frequency distribution of resulting POS tags is shown in Fig. 2, which provides insights into the text structure.

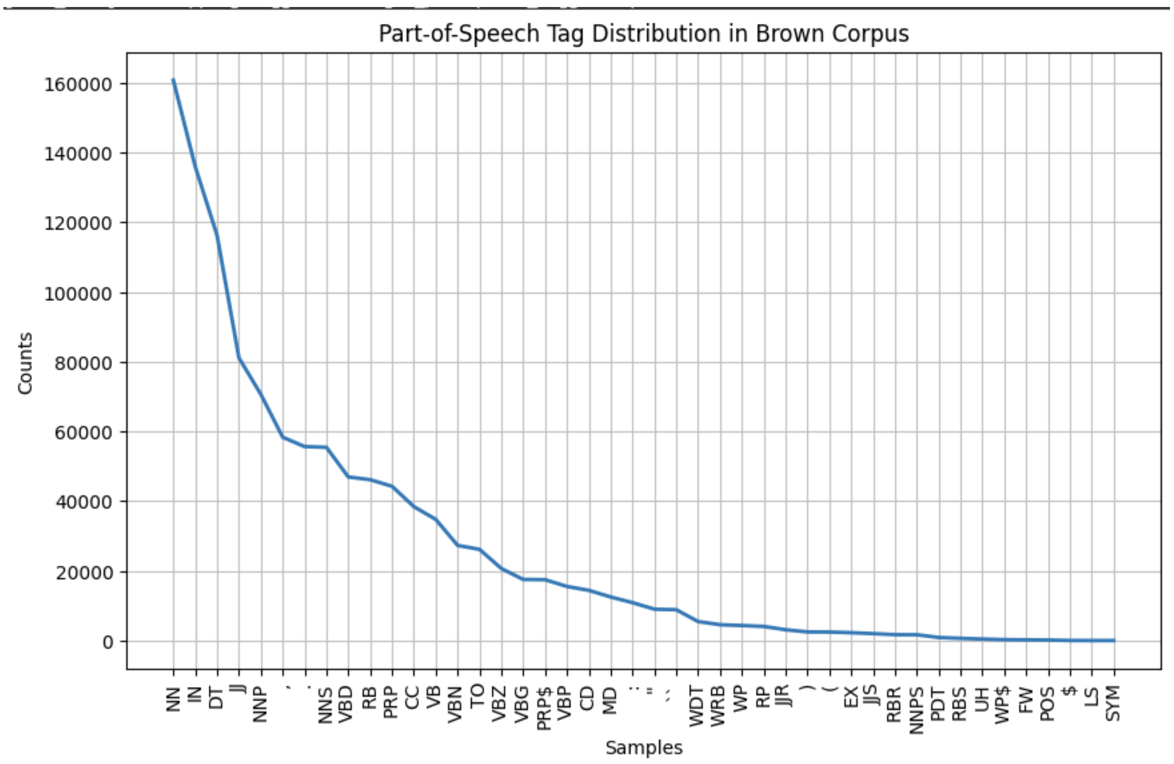

Fig. 2: Top 30 POS Tag Distribution.

A word cloud is generated to visually understand the frequency of the most common words. The frequency influences the size and color of the words; the most common words will be more significant in size. Furthermore, the maximum length of a sentence and the number of sentences from the corpus are calculated and analyzed. Thus, a resulting distribution graph for sentence lengths was plotted, which helps to understand the variability and structure of sentences within the corpus.

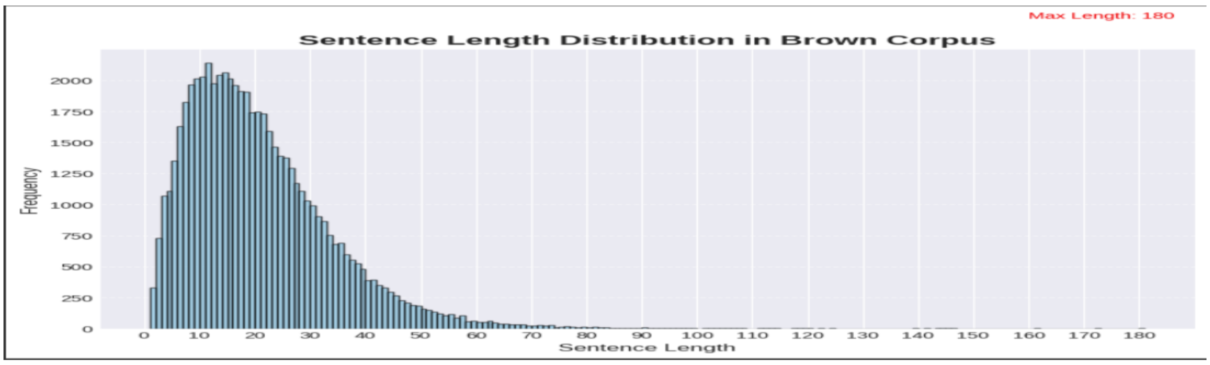

Fig. 3: Sentence length distribution graph in the Brown corpus.

In the context of text embedding, the vocabulary size refers to the number of unique words in the corpus. Furthermore, the data is preprocessed for tokenization and padding. Tokenization is the process of breaking down text or a sequence of characters into smaller units known as tokens. They are units of text that are used as input for natural processing tasks. The nltk tokenizer class (version 3.8) from Keras Library was employed to tokenize the sentences into sequences of integers, padding was employed to ensure tokenized sequence should be used in fixed length. The resulting padded sequences are the basis for text embedding and modeling processes.

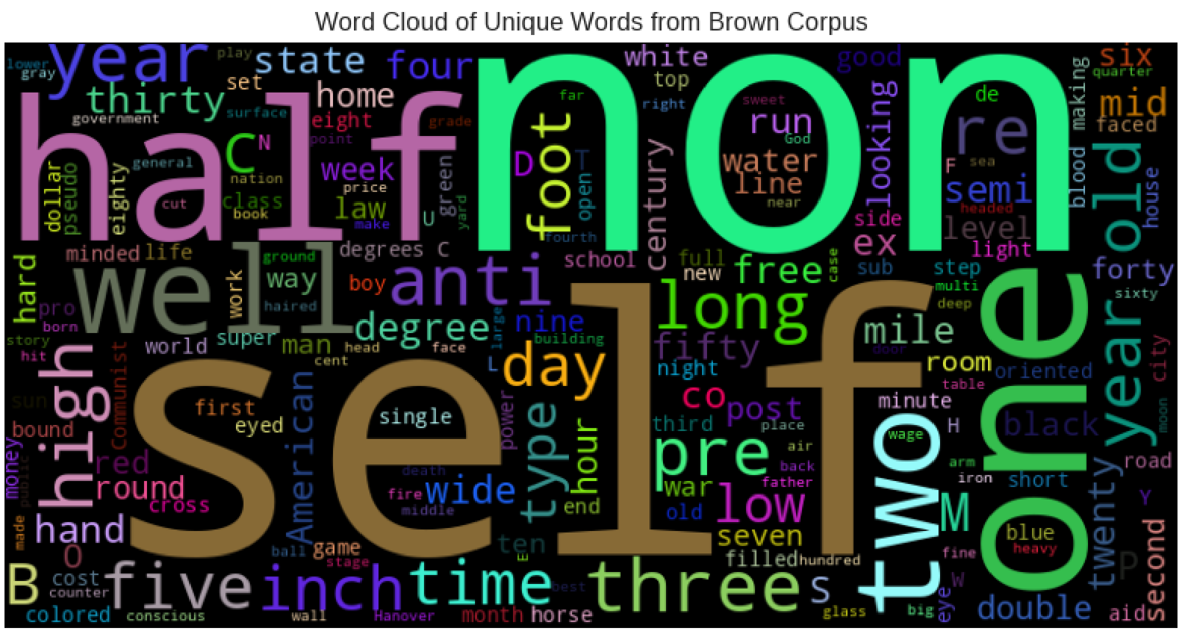

Fig. 4: This word cloud represents the frequency of words within the Brown Corpus, it is a comprehensive collection of American English texts comprised of various genres and published between 1961 and 1972. In this visualization, larger and bolder words indicate higher frequency.

4 Design Specification

This design specification outlines the techniques, architecture, and framework for implementing hierarchical Text embedding using stack-layered autoencoders. The primary goal of this section is to investigate the influence of Bidirectional LSTMs and Bidirectional GRUs on stack-layered autoencoders, with a focus on capturing intricate hierarchical features within textual data. The proposed architecture in this study comprised eight models: four models using Bidirectional LSTMs (Long-short-term memory) and four others Bidirectional GRUs (Gated Recurrent unit). According to (Tan et al., 2000), Text is a rich source of information and gives us the opportunity to gain valuable insights which cannot be achieved using quantitative methods. The main aim of different natural language processing methods is to get a human-like understanding of the text (Wang, Nulty and Lillis, 2020). Several approaches are available to carry out information from vast amounts of text from the corpora; one of them is autoencoder. Autoencoders are employed in unsupervised learning techniques to reduce the dimensions of the data which non-linear to describe relationships between dependent and independent features. Thus, effectively used for feature extraction. However, feature extraction for datasets having complex relationships is not a small feat. That is why an autoencoder is not sufficient. A single autoencoder might not be able to capture all the intrinsic features. Therefore, for such cases, we study the effect of stack-layered autoencoders with Bidirectional LSTMs and Bi-directional GRUs. The Bidirectional LSTM has distinctive characteristics to process input sequences in both forward and backward directions making it an asset for valuable tasks where patterns are complex and required to capture relationships between the elements of sequences. Thus, it is very crucial to feature extraction tasks. Bidirectional GRUs has almost the same abilities as Bi-LSTM but has an advantage in terms of less training time and operate on fewer parameters. Therefore, Bi-LSTM and Bi-GRU are the well-suited layer for stacked autoencoders and for our research endeavour.

In our study, the proposed architecture involves building four incremental models of autoencoders (Bidirectional LSTMs and Bidirectional GRUs). Each model has two significant components: an encoder and a decoder. The encoder is constructed using Bidirectional LSTMs and Bidirectional GRUs in a stack-layered manner. The input sequential data is processed through successive layers of these recurrent units to extract meaningful hierarchical features from the input text sequences. The encoder is responsible for encoding or capturing the intricate features from the input text, and the decoder aims to reconstruct the input text from the encoded representation while preserving the hierarchical features. The encoder is responsible for transforming input sequences into latent representations by using Bidirectional LSTMs and Bidirectional GRUs. It starts with the embedding layer, which maps words to continuous vectors, Bi-LSTM layers follow the embedding layer, capturing the context in both forward and backward directions at each layer to uncover the intricate dependencies present in the text. Similarly, Bidirectional-GRU is applied in the same manner in other models, which provides an alternative mechanism to capture temporal patterns. The decoder's objective is to reconstruct the input sequences from the encoded representations keeping the hierarchical information intact. A RepeatVector layer is used to duplicate the encoded representation across the sequence length, facilitating reconstruction. Moreover, the decoder includes layers similar to the encoder's architecture to generate the output sequences.

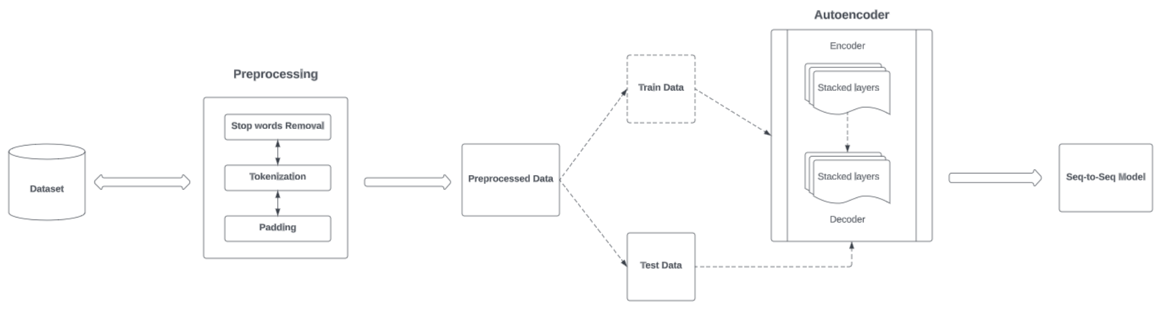

Fig. 5: Architecture for Design Specification.

Figure. 5 illustrates the architecture for design specification utilized in the study and also demonstrates the processes carried out in this research from the beginning to the execution. The dataset is downloaded and is made available from the nltk library to preprocess. Python is used to perform cleaning, visualize, or to gain insights from the corpus. Necessary calculations such as vocab size, maximum length of sentences, and count of sentences, and words are carried out in order to understand the variability of sentences in the corpus. Subsequently, the data was pre-processed for tokenization and padded up to a fixed length before feeding to the autoencoder. The model received the prepared data for training and testing. The models are evaluated on the basis of training time, Training and Testing Accuracy, and Loss functions.

5 Implementation of the Models

The methods used to complete the assignment for the study are briefly discussed in this section.

5.1 Experimental Setup

Utilizing the advantages of readily accessible library modules, Python programming language is employed for the completion of this task, specifically version 3.6.9. Both the local workstation and Google web services are used for the execution of tasks. The local workstation with the hardware configuration of 64-bit Windows 11 OS, 11th Gen Intel(R) Core(TM) i5-11300H @ 3.10GHz Processor and 16GB of RAM was initially used. The first model training was done on the local workstation, but as we started training the model, it required massive amount of training time. Because of this, we switched to Google Clab Pro services. The Google Colab Platform is based on IaaS, which utilizes the Google Compute engine for computing operations. It offers a platform to write and execute code collaboratively and has a Jupyter Notebook interface specially designed for tasks like data analysis, machine learning, and deep learning. The High-end CPU, 25GB of RAM, and 100 computing units were configured to execute the task.

5.2 Implementation of Stack-Layered Autoencoders

The implementation process involved several key steps in preprocessing and analyzing the text data using Python programming language (version 3.6.9) along with essential libraries such as keras, numpy, pandas, matplotlib, etc. The Natural Language Toolkit (NLTK) and Keras libraries were employed, requiring installation through pip commands at the beginning. These libraries are required to facilitate work with text processing and deep learning operations.
To begin, the Brown Corpus was downloaded using the NLTK library. The corpus consists of text documents in the English language categorized by 15 genres, making it suitable for several NLP tasks. The frequency distribution of words is calculated using the FreqDist function, and a visualization was created to plot the top 30 most common words in the corpus utilizing the matplotlib library. In the next step, we conducted a POS tagging analysis in which tags or grammatical categories were assigned to each word in a sentence, the frequency distribution of POS tags was determined, and the corresponding plot was generated. For further analysis, the number of sentences in the corpus and the length of sentences were calculated with the help of the number of words in a sentence. The distribution of the maximum length of sentences was plotted using a histogram which offered insights into the structure of text data in the corpus. A vocabulary set was created to store unique words from the Brown Corpus. The number of unique words and the count of sentences are calculated and printed to provide an overview of

the corpus characteristics. Unique words from the vocabulary were combined into a single string and to add the element of randomness, a function was defined to generate random colors, and a visually appealing word cloud was created to get more insights. Following these preliminary steps, parameters were defined for the subsequent data processing. These included the number of words in the vocabulary (num_words), the maximum length of the sequence (maxlien), and other hyperparameters required for the models. Furthermore, Tokenization, is a crucial step in natural language processing, was performed to convert the text data into sequences of integers or tokens. The Tokenizer class from Keras was utilized to tokenize the sentences into the sequence of tokens. The tokenizer was fitted on the input text data, and sequences were generated by mapping each word to its corresponding integer. The tokenized sequences were then padded to fixed uniform length using the pad_sequences function, ensuring consistent input dimensions for subsequent processing. These steps laid the groundwork for further processing and analysis using the autoencoder architecture with bidirectional LSTM and GRU layers.

Table 2: Description of Hyperparameters.

Hyperparameters	**Description**	**Value**
Hidden Layers	Intermediate layers between the input layer and the output layer of a neural network.	(1,2,3,4)
Neural Layers	All layers within a neural network.	(1-8)
Embedding_Dimension	The embedding dimensions in encoder and decoder.	128, 256
Loss_Function	A measure of the difference between the predicted values and actual target values.	Sparse categorical crossentropy
Optimizer	A method for minimizing the loss function.	Adam
Activation	To capture complex relationships between input features and model predictions.	SoftMax, ReLu
Early_Stopping	It Stops the training process once the model's performance starts to degrade.	1
Workers	Speed up data processing by performing tasks concurrently.	16
Epochs	Number of times the entire training dataset will be used to train the model	15

5.2.1 Implementation of Stack-Layered Autoencoders using Bidirectional LSTMs

To build a hierarchical text embedding model using a stack-layered autoencoder architecture, we utilized Bidirectional LSTMs, sequentially in the incremental form in different models. We first started off with the input layer in the encoder component. It is designed to accept input sequences, where each sequence can have a maximum length of 'maxlen'. Then, we utilized the embedding layer to convert input sequences into dense vector representations. These vectors capture the semantic and syntactic relationships between words. The possible words in a sequence can be represented 'num_words' which is associated with the embedding dimension provided for the layers. A series of Bidirectional Long Short-Term Memory (LSTMs) are employed as per the model architecture, constituting the essence of the encoder phase. The benefit of utilizing the LSTM is its ability to capture temporal dependencies within sequential data. The term 'Bidirectional' denotes its capability to perform concurrent analysis and capture semantic and contextual nuances in both forward and backward directions. The purpose of employing stacked bidirectional LSTMs hierarchically is to construct representations at each stage, which are modified from the previous one to enhance their capacity to comprehend the context in a much better way. Each layer encompasses 128 units and employs the Rectified Linear Unit (ReLU) activation function to introduce non-linearity. Each layer passes its output sequences to the next layer and the following layer tries to capture the output sequences and produce its own sequences with the modified version of meaningful representations. The last layer processes the output sequence of the previous layer and generates a singular output sequence. An encoder model is designed to take the input sequences from the last layer and generates its output sequence, which encapsulates the encoder component of the hierarchical text embedding architecture.

The outcome of the encoder model serves as the input for the decoder segment of the architecture and the RepeatVector layer is employed to replicate the encoder output sequence and to match the sequence length 'maxlen' which prepares the input for the Decoder Bidirectional LSTM layers. Similar to the encoder architecture, the decoder employs a comparable number of Bidirectional layers. However, the role of these is to reconstruct the original sequences from the hierarchical embeddings learned by the encoder. The multiple decoder LSTM layers are duplicated for each layer in the encoder. The output from the decoder bidirectional LSTM layers is directed to the dense layer, designed with the softmax activation function to predict the most probable word from the vocabulary for each position in the sequence. The number of neurons in this dense layer corresponds to "num_words".

The conclusive output from the dense layer signifies the regenerated sequences founded on the acquired hierarchical embeddings. This methodology holds the potential to comprehend intricate patterns and relationships present within text data.

5.2.2 Implementation of Stack-Layered Autoencoders using Bidirectional GRUs

The same architecture and implementation are utilized for stack-layered autoencoders using bidirectional GRUs. However, GRU has a simpler architecture as compared to LSTMs Which require fewer parameters and can lead to efficient memory utilization. Moreover, GRUs address the vanishing gradient problem more efficiently than LSTMs and have faster convergence rates due to simpler architecture and gating mechanisms.

6 Evaluation.

In the result and evaluation section, we have shown the trade-off between complexity and accuracy in stacked autoencoder architecture. A number of experiments are proposed with incremental layers from one to four and evaluate every single bidirectional LSTM and bidirectional GRU layer to show their efficacy in the model.

6.1 One-Bidirectional LSTM In Stack-Layered Autoencoder.

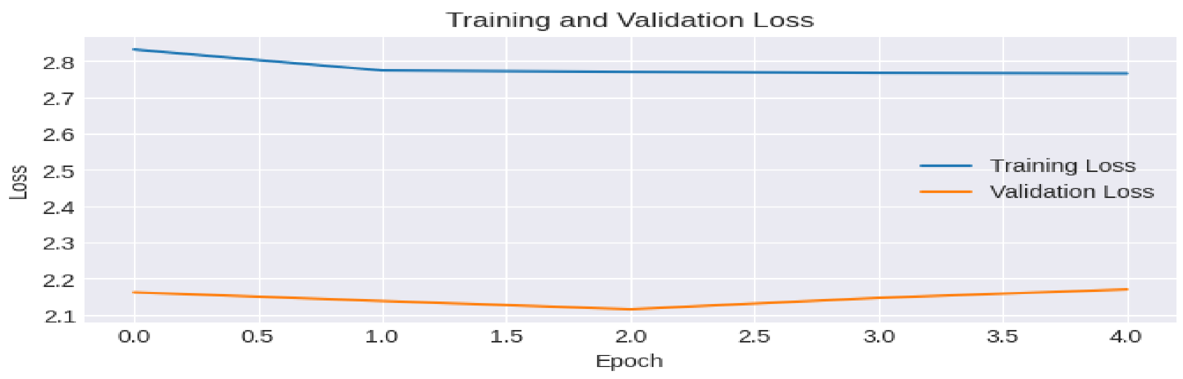

Fig. 6: Loss Graph for One-Bidirectional LSTM in Stack-Layered Autoencoder.

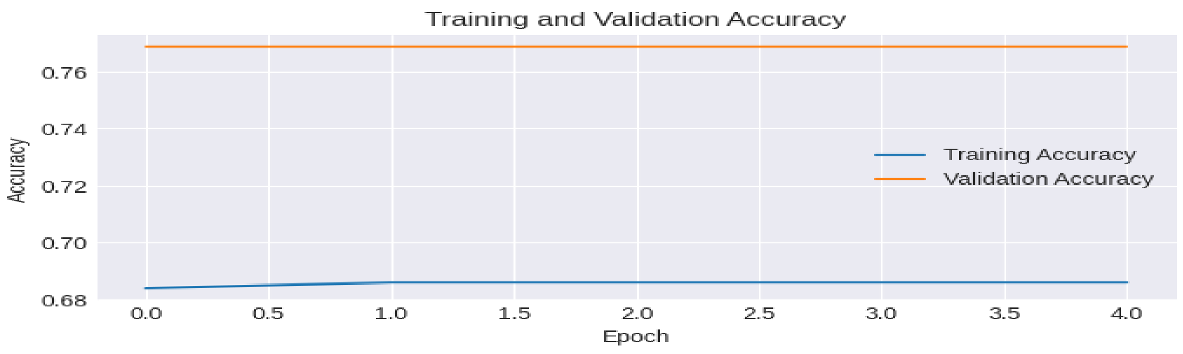

Fig. 7: Accuracy Graph for One-Bidirectional LSTM in Stack-Layered Autoencoder.

6.2 Two-Bidirectional LSTM In Stack-Layered Autoencoder.

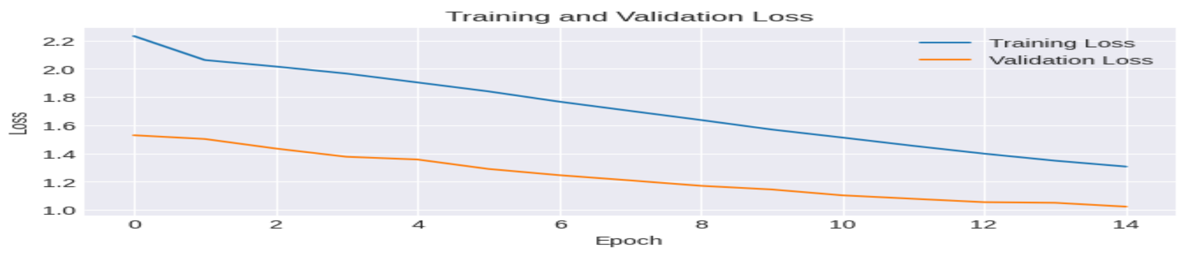

Fig. 8: Loss Graph for Two-Bidirectional LSTM in Stack-Layered Autoencoder.

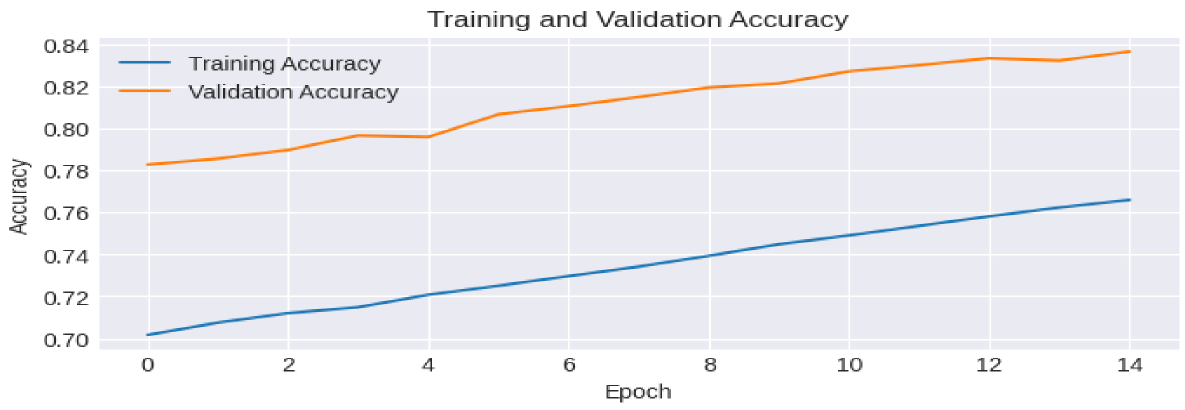

Fig. 9: Accuracy Graph for Two-Bidirectional LSTM in Stack-Layered Autoencoder.

6.3 Three-Bidirectional LSTM In Stack-Layered Autoencoder.

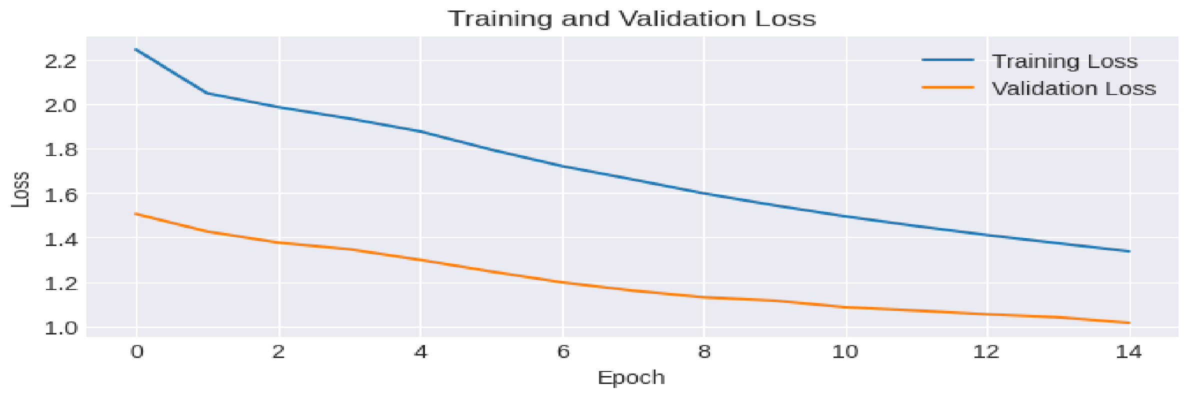

Fig. 10: Loss Graph for Three-Bidirectional LSTM in Stack-Layered Autoencoder.

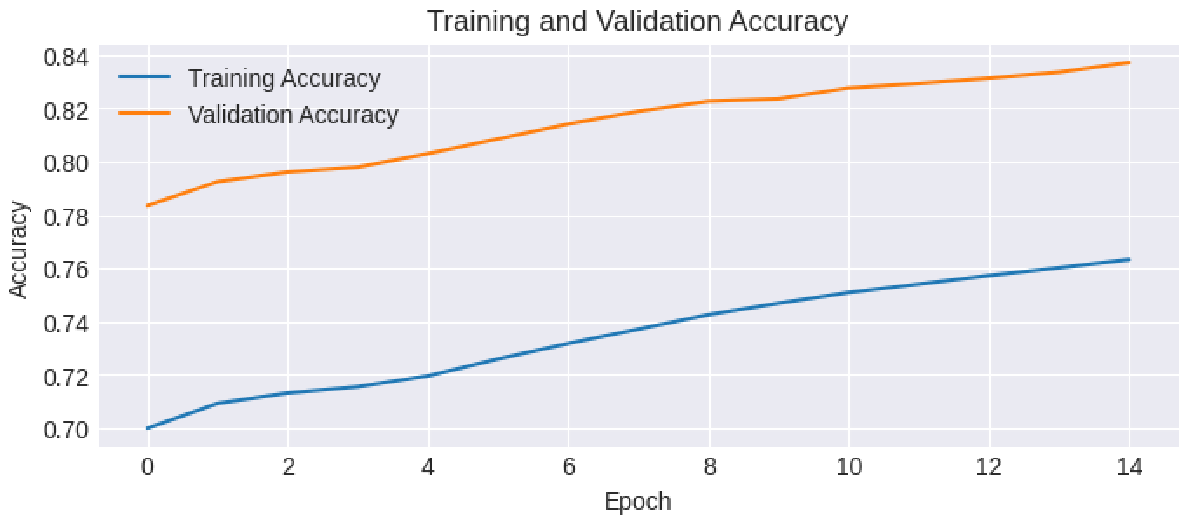

Fig. 11: Accuracy Graph for Three-Bidirectional LSTM in Stack-Layered Autoencoder.

6.4 Four-Bidirectional LSTM In Stack-Layered Autoencoder.

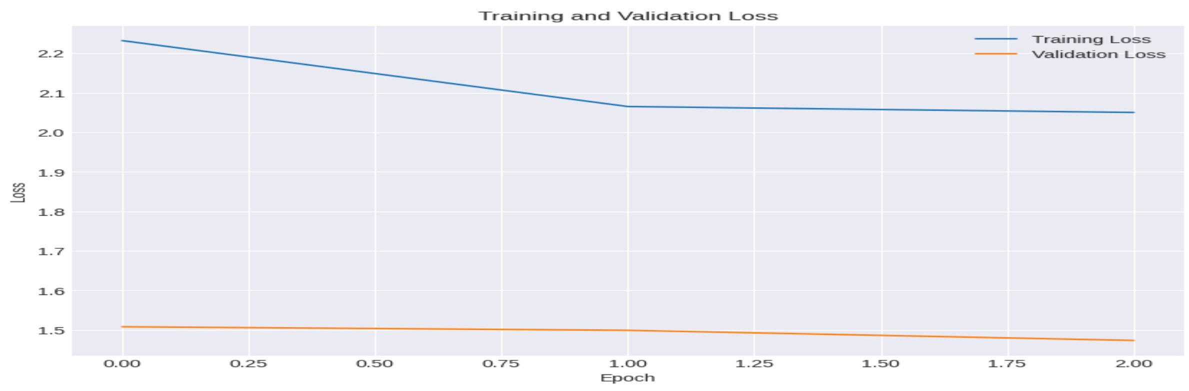

Fig. 12: Loss Graph for Three-Bidirectional LSTM in Stack-Layered Autoencoder.

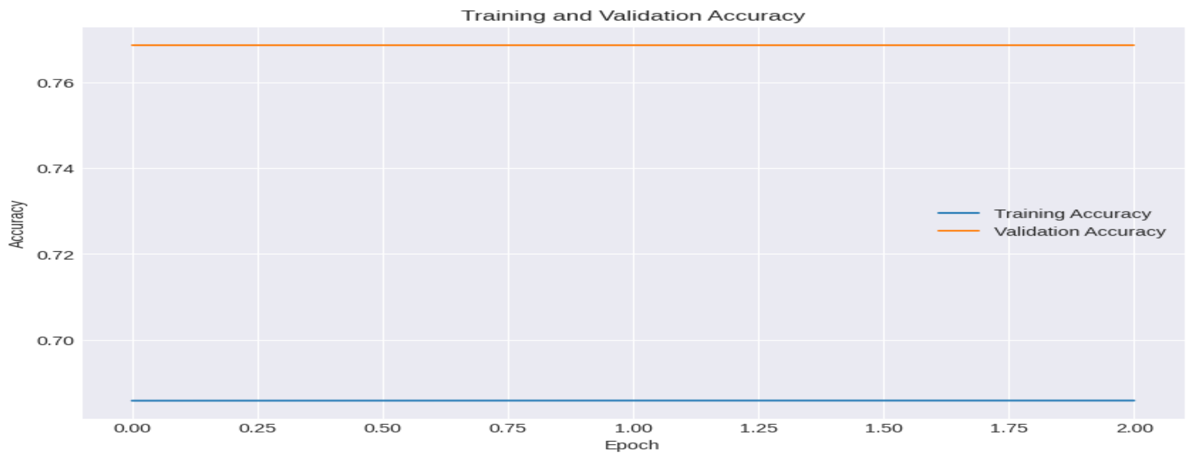

Fig. 13: Accuracy Graph for One-Bidirectional LSTM in Stack-Layered Autoencoder.

6.5 One-Bidirectional GRU In Stack-Layered Autoencoder.

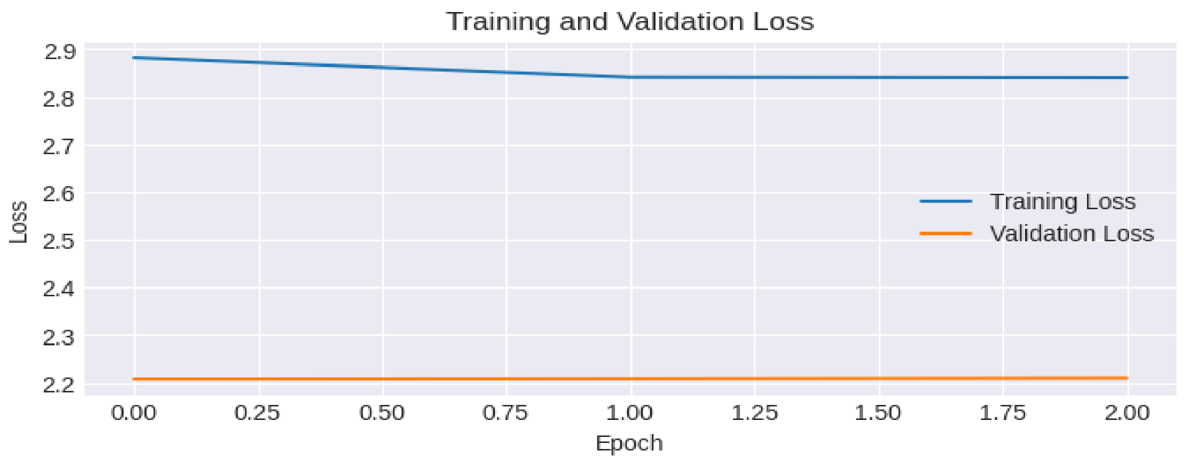

Fig. 14: Loss Graph for One-Bidirectional GRU in Stack-Layered Autoencoder.

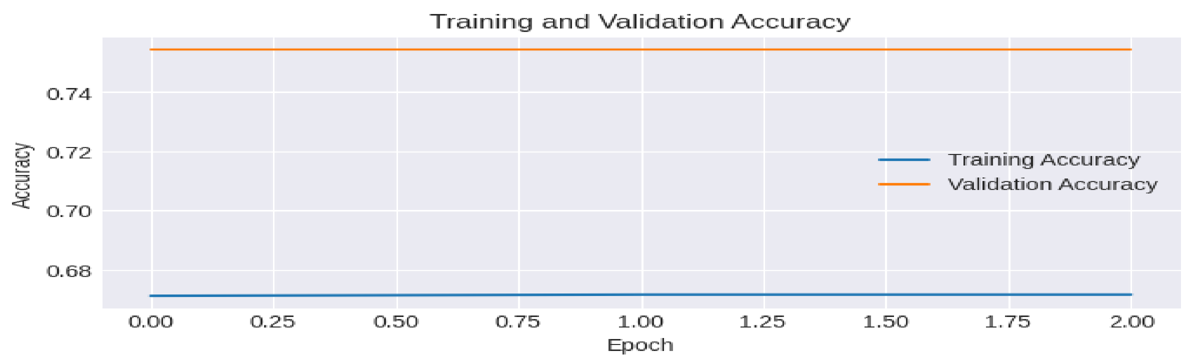

Fig. 15: Accuracy Graph for One-Bidirectional GRU in Stack-Layered Autoencoder.

6.6 Two-Bidirectional GRU In Stack-Layered Autoencoder.

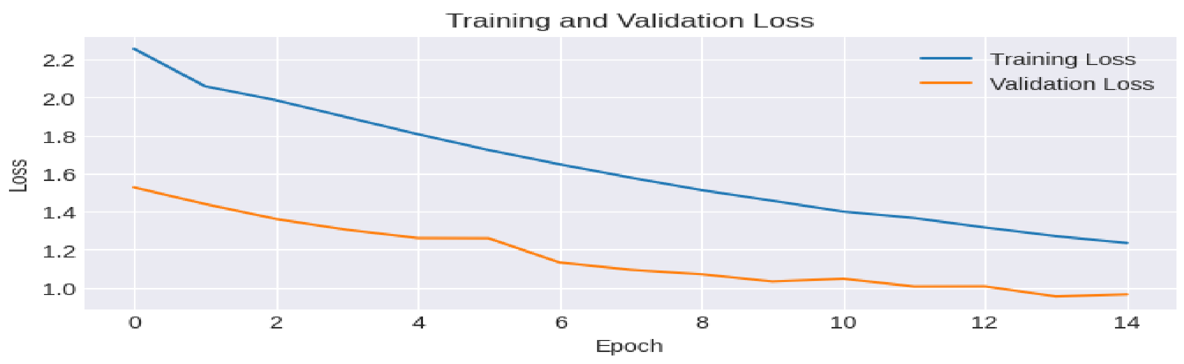

Fig. 16: Loss Graph for Two-Bidirectional GRU in Stack-Layered Autoencoder

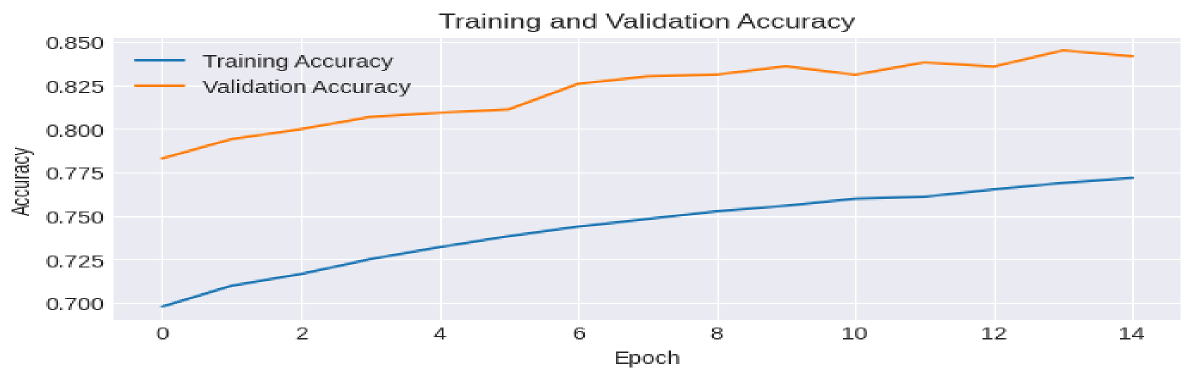

Fig. 17: Loss Graph for Two-Bidirectional GRU in Stack-Layered Autoencoder

6.7 Three-bidirectional GRU In Stack-Layered Autoencoder.

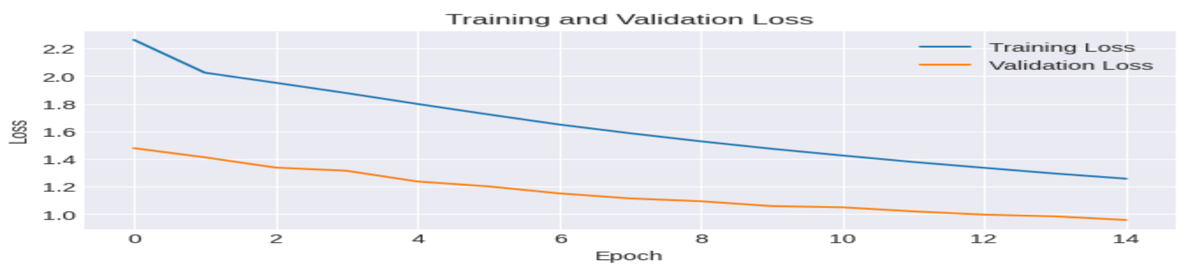

Fig. 18: Loss Graph for Three-Bidirectional GRU in Stack-Layered Autoencoder.

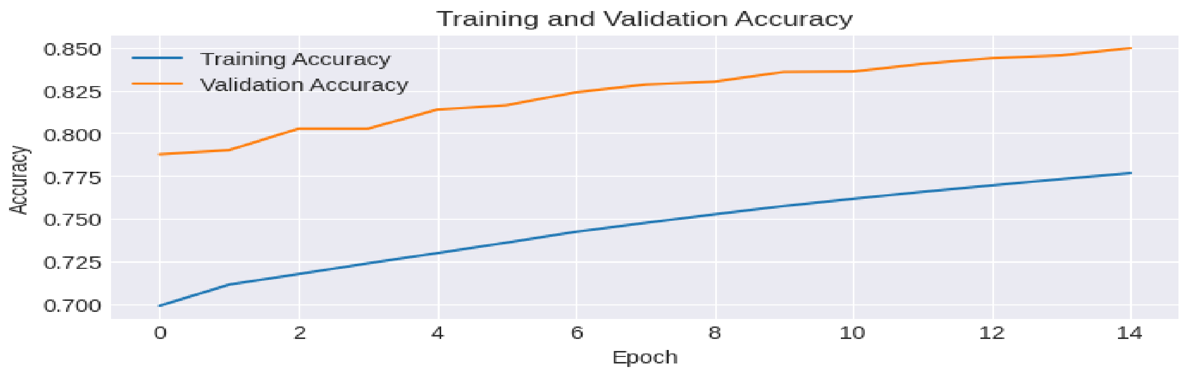

Fig. 19: Accuracy Graph for Three-Bidirectional GRU in Stack-Layered Autoencoder.

6.8 Four-Bidirectional GRU In Stack-Layered Autoencoder.

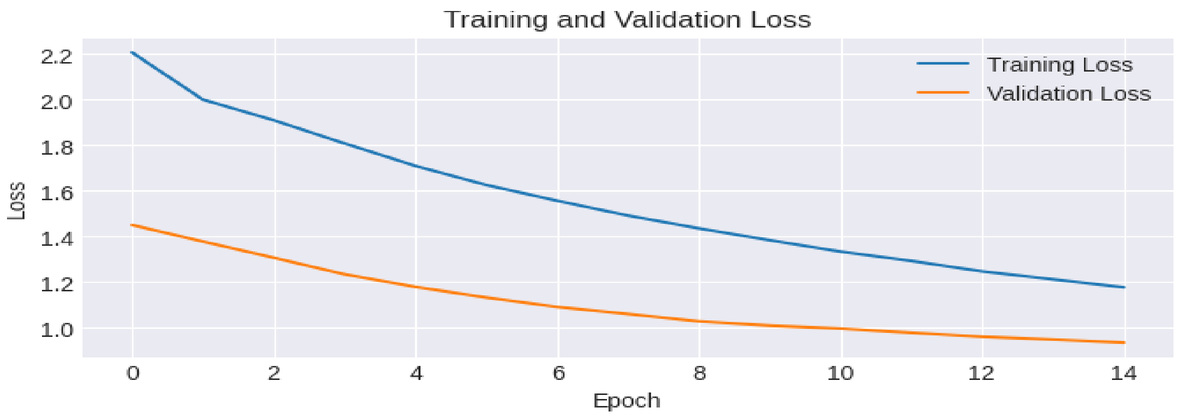

Fig. 20: Loss Graph for Four-Bidirectional GRU in Stack-Layered Autoencoder.

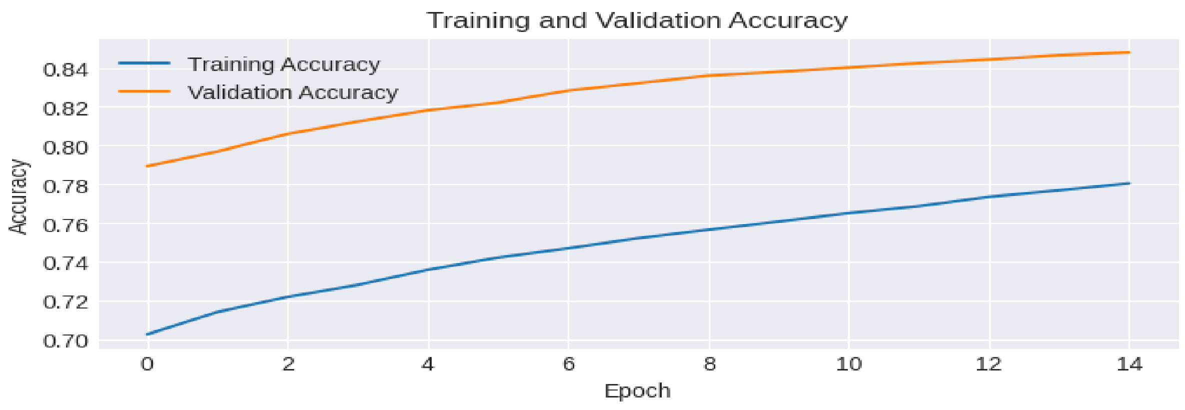

Fig. 21: Accuracy Graph for Four-Bidirectional GRU in Stack-Layered Autoencoder

6.9 Discussion

In the bidirectional LSTM architecture of the stacked autoencoders, the models with increasing layers consistently decrease the loss function and improve accuracy. With Two-BiLSTM and Three-BiLSTM models demonstrate enhancing accuracy and reducing loss. However, the Four Bi-LSTM model encounters converging issues. On the other hand, Bidirectional GRUs architecture performance benefited from the increased number of layers, resulting in improved accuracy, decreased training loss, and efficient training time. Moreover, it can be noted that Four Bi-GRU has outperformed all other models with its performance in accuracy in a reasonable amount of time.

Table 3: Result and Evaluation table for Bidirectional LSTMs in Stack-Layered Autoencoder.

Layers	Loss	Accuracy	Training Time	Figure Reference
Bi-LSTM	2.6530	70.25	3hrs 7mins	Fig. 6 and Fig. 7
Two_Bi-LSTM	1.2044	78.33	6hrs 1mins	Fig. 8 and Fig. 9
Three_Bi-LSTM	1.2277	78.18	7hrs 11mins	Fig. 10 and Fig. 11
Four_Bi-LSTM	nan	68.81	2hrs 36mins	Fig. 12 and Fig. 13

Table 4: Result and Evaluation table for Bidirectional LSTMs in Stack-Layered Autoencoder.

Layers	Loss	Accuracy	Training Time	Figure Reference
Bi-GRU	2.7098	68.81	1hr 56mins	Fig. 14 and Fig. 15
Two_Bi-GRU	1.1583	78.58	6hrs 20mins	Fig. 16 and Fig. 17
Three_Bi-GRU	1.1466	79.74	7hrs 17mins	Fig. 18 and Fig. 19
Four_Bi-GRU	1.0873	79.75	6hrs 29mins	Fig. 20 and Fig. 21

This section will discuss the results obtained from the experiments as part of our research study. Evaluating the results will challenge the effectiveness and design of our models. The goal of this study is to investigate the effect of stack-layered autoencoder on hierarchical embedding utilizing bidirectional LSTM and GRU layers. The obtained outcomes from the result of experiments provide valuable insights into the effectiveness of different architectures of the autoencoder. The incremental layered architecture from a single Bi-LSTM or GRU layer to multiple layers sheds light on the layer depth's impact on loss and accuracy. The increase in the layer in the autoencoder showcased the improvement in its accuracy and decrementing loss values. However, there are cases where more complex architectures lead to diminishing results. Moreover, as we move up the hierarchy of layers, the trade-off between complexity and accuracy becomes clear. The training time of a model has great significance in the design and specification of architectures. As the model evolves in complexity, the associated training time changes considerably. It drives a crucial factor in organizations while implementing large models for research and development.

Though our model design has been diligently executed, there are some areas for potential improvements. The bidirectional LSTM model with four layers fails to converge even though the same model with two and three layers displayed good accuracy and lower training loss values. The models' convergence and performance stabilization could be further optimized by tuning the hyperparameters, such as exploring adaptive learning rate, and utilizing regularization methods to mitigate overfitting. Additionally, the presence of Nan values in training loss indicates potential instabilities which could be solved through modified training approaches. Relating to our findings, we could have also evaluated the hierarchical embedding representations downstream to several tasks such as machine language translation, Sentiment analysis, Q/A system, etc. Comparison with other similar studies shows how the architecture could be improved with better efficiency.

7 Conclusion and Future Work

In summary, this research investigated the effects of bidirectional LSTM and GRU layers in a stack-layered autoencoder architecture for hierarchical text embedding. The research question revolves around finding the impact of these layers on text-embedding techniques and model performance. Our objectives successfully addressed the understanding of hierarchical meaningful representations, construction of experimental setups, and analysis results. Our study clearly indicates the impact of layer depth in an autoencoder on both loss and accuracy metrics. Our models have shown promising results with incrementing layers in the autoencoders. From the results, we have shown that a three-layered bidirectional GRU autoencoder architecture has the best trade-off between model complexity and accuracy. The higher number of layers such as in four GRU layered architecture could increase the learning time, but as seen in the work, the early stopping function can actually reduce the learning time. However, the accuracy is marginally higher than the three-layered GRU architecture. Moreover, several inconsistencies are observed in the model architecture of four bidirectional LSTM layers. The model displayed a complexity-performance trade-off, and the optimization algorithm struggled to find the optimal set of weights that minimizes the loss function, resulting in convergence issues. These convergence issues might arise due to improper initialization of weights, inadequate learning rates, or vanishing gradients problems. These issues can be addressed by carefully performing hyper-tuning the parameters or using L2 regularization methods.

We have also observed that stack-layered autoencoder architecture captures meaningful representation hierarchically, but we needed to evaluate the model by downstream the tasks to machine translation or Q/A system. Furthermore, we analysed the parameters we selected in strict conditions. This is because these models are computationally expensive and require considerable training time to execute diligently. Moreover, customizing the model to achieve better results using attention mechanisms in stack-layered autoencoders may deliver better results utilizing bidirectional LSTM, and GRU is still an open question. Ultimately, our study contributes to a deeper comprehension of these models while revealing untapped research directions.

References

Simon, P. (2013) Too big to ignore: The business case for big data. Available at: http://ci.nii.ac.jp/ncid/BB14252363.

LeCun, Y., Bengio, Y. and Hinton, G.E. (2015) “Deep learning,” Nature, 521(7553), pp. 436–444. Available at: https://doi.org/10.1038/nature14539.

N, T.R. and Gupta, R. (2020) “A survey on machine learning approaches and its techniques:,” 2020 IEEE International Students’ Conference on Electrical, Electronics and Computer Science (SCEECS) [Preprint]. Available at: https://doi.org/10.1109/sceecs48394.2020.190.

Li, P., Pei, Y. and Li, J. (2023) “A comprehensive survey on design and application of autoencoder in deep learning,” Applied Soft Computing, 138, p. 110176. Available at: https://doi.org/10.1016/j.asoc.2023.110176.

Rumelhart, D.E., Hinton, G.E. and Williams, R.J., 1985. Learning Internal Representations by Error Propagation: [online] Fort Belvoir, VA: Defense Technical Information Center. https://doi.org/10.21236/ADA164453.

Bourlard, H. and Kamp, Y. (1988) “Auto-association by multilayer perceptrons and singular value decomposition,” Biological Cybernetics, 59(4–5), pp. 291–294. Available at: https://doi.org/10.1007/bf00332918.

Vincent, P. et al. (2010) “Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion,” Journal of Machine Learning Research, 11(110), pp. 3371–3408. Available at: https://www.jmlr.org/papers/volume11/vincent10a/vincent10a.pdf.

Bengio, Y. (2009) “Learning deep architectures for AI,” Foundations and Trends in Machine Learning, 2(1), pp. 1–127. Available at: https://doi.org/10.1561/2200000006.

Huang, Z., Xu, W. and Yu, K. (2015) “Bidirectional LSTM-CRF models for sequence tagging,” arXiv (Cornell University) [Preprint]. Available at: https://arxiv.org/pdf/1508.01991.

Vinyals, O. and Le, Q.V. (2015) “A neural conversational model,” arXiv (Cornell University) [Preprint]. Available at: http://cs224d.stanford.edu/papers/ancm.pdf.

Sutskever, I., Vinyals, O. and Le, Q.V. (2014) “Sequence to Sequence Learning with Neural Networks,” Neural Information Processing Systems, 27, pp. 3104–3112. Available at: http://cs224d.stanford.edu/papers/seq2seq.pdf

Wang, T. et al. (2016) “An experimental study of LSTM Encoder-Decoder model for text simplification,” arXiv (Cornell University) [Preprint]. Available at: https://www.arxiv.org/pdf/1609.03663.

Xu, Q. et al. (2016) “The Learning Effect of Different Hidden Layers Stacked Autoencoder,” 8th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC) [Preprint]. Available at: https://doi.org/10.1109/ihmsc.2016.280.

Mai, F. and Henderson, J. (2021) “Bag-of-Vectors autoencoders for unsupervised conditional text generation,” arXiv (Cornell University) [Preprint]. Available at: https://doi.org/10.48550/arxiv.2110.07002

Wang, C., Nulty, P. and Lillis, D. (2020) “A Comparative Study on Word Embeddings in Deep Learning for Text Classification,” N Proceedings of the 4th International Conference on Natural Language Processing and Information Retrieval (NLPIR ’20) [Preprint]. Available at: https://doi.org/10.1145/3443279.3443304.

Zhang, Y., Liu, Q. and Song, L. (2018) “Sentence-State LSTM for Text Representation,” Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)" [Preprint]. Available at: https://doi.org/10.18653/v1/p18-1030.

Yang, Z. et al. (2017) “Improved variational autoencoders for text modeling using dilated convolutions,” International Conference on Machine Learning, pp. 3881–3890. Available at: http://proceedings.mlr.press/v70/yang17d/yang17d.pdf.

Mangal, S., Joshi, P. and Modak, R. (2019) “LSTM vs. GRU vs. Bidirectional RNN for script generation.,” arXiv (Cornell University) [Preprint]. Available at: https://arxiv.org/pdf/1908.04332.pdf.

Zulqarnain, M. et al. (2019) “Efficient processing of GRU based on word embedding for text classification,” JOIV : International Journal on Informatics Visualization, 3(4). Available at: https://doi.org/10.30630/joiv.3.4.289

Umer, M. et al. (2022) “Impact of convolutional neural network and FastText embedding on text classification,” Multimedia Tools and Applications, 82(4), pp. 5569–5585. Available at: https://doi.org/10.1007/s11042-022-13459-x.

Xu, Q. and Zhang, L. (2015) “The effect of different hidden unit number of sparse autoencoder,” The 27th Chinese Control and Decision Conference (2015 CCDC) [Preprint]. Available at: https://doi.org/10.1109/ccdc.2015.7162335.

Tan, A.-H., Ridge, K., Labs, D. and Terrace, H., 2000. Text Mining: The state of the art and the challenges.

Naseem, U. et al. (2021) “A comprehensive survey on word representation models: From Classical to State-of-the-Art Word Representation Language Models,” ACM Transactions on Asian and Low-resource Language Information Processing, 20(5), pp. 1–35. Available at: https://doi.org/10.1145/3434237.

Goodfellow, I. et al. (2017) “GAN(Generative Adversarial Nets),” Journal of Japan Society for Fuzzy Theory and Intelligent Informatics, 29(5), p. 177. Available at: https://doi.org/10.3156/jsoft.29.5_177_2.

Cinelli, L.P. et al. (2021) “Variational Autoencoder,” in Springer eBooks, pp. 111–149. Available at: https://doi.org/10.1007/978-3-030-70679-1_5.