A Comparison of Long Short-Term Memory, Convolutional Neural Network, Transformer, and Mamba Models for Sentiment Analysis

By

Hang Ruan
B.Sc., University of Victoria, 2022

A Project Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master of Engineering

in the Department of Electrical and Computer Engineering

A Comparison of Long Short-Term Memory, Convolutional Neural Network, Transformer, and Mamba Models for Sentiment Analysis

By

Hang Ruan
B.Sc., University of Victoria

Supervisory Committee

Dr. T. Aaron Gulliver
(Department of Electrical and Computer Engineering)

Dr. Mihai Sima
(Department of Electrical and Computer Engineering)

Abstract

Sentiment analysis is a critical task in Natural Language Processing (NLP) that helps decode the emotions and opinions embedded in text. With applications spanning from market research and social media monitoring to political analysis and customer feedback evaluation, sentiment analysis provides invaluable insights into public opinion and consumer behavior. This project studies the evolution of sentiment analysis models, focusing on the advancements made by deep learning techniques such as Long Short-Term Memory (LSTM), Convolutional Neural Networks (CNNs), and transformer-based models like Bidirectional Encoder Representations from Transformers (BERT) [1] and Generative Pre-trained Transformer (GPT) [2]. These models have set new benchmarks for accuracy, efficiency, and versatility. Additionally, this work explores Mamba [3], a recent State Space Model (SSM) designed to overcome the computational challenges of transformers in handling long sequences and demonstrates state-of-the-art performance on language modeling tasks comparable to transformers twice its size [3]. This study examines the strengths and limitations of these models, comparing their performance on sentiment analysis datasets to provide a comprehensive understanding of their applicability and efficacy in various contexts.

Contents

List of Tables

List of Figures

List of Acronyms

NLP	Natural Language Processing
LSTM	Long Short-Term Memory
CNN	Convolutional Neural Network
BERT	Bidirectional Encoder Representations from Transformers
GPT	Generative Pre-trained Transformer
SSM	State Space Model
SVM	Support Vector Machine
GPU	Graphics Processing Unit
TPU	Tensor Processing Unit
RNN	Recurrent Neural Network
FFT	Fast Fourier Transform
SRAM	Static Random-Access Memory
HBM	High Bandwidth Memory
LLM	Large Language Model
HiPPO	High-order Polynomial Projection Operators

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Dr. T. Aaron Gulliver, for his continuous support, invaluable guidance, and insightful feedback throughout this research. His expertise has been instrumental in shaping this work, and his encouragement has been a driving force in overcoming the challenges I encountered along the way. I am truly fortunate to have had the opportunity to learn from him.

I am also profoundly grateful to my parents for their unwavering support, love, and encouragement. Their belief in my abilities has been a source of strength and motivation throughout this journey. They have provided me with the foundation to pursue my goals and have been a constant source of comfort and inspiration.

Finally, I would like to extend my thanks to my friends, whose encouragement and collaboration have enriched this research experience. Their insights and support have made this journey more meaningful and enjoyable. This work is dedicated to my family and mentors, whose constant support has been the cornerstone of my achievements.

Chapter 1 Introduction

Sentiment analysis, a crucial technique in Natural Language Processing (NLP), has become increasingly important in today's data-driven world. It involves determining the emotional tone or attitude expressed in text, and its applications span across various domains such as market research, social media monitoring, and customer feedback analysis. By analyzing texts like reviews, social media posts, and political commentary, sentiment analysis helps identify whether the sentiments conveyed are positive or negative, providing valuable insights into public opinion and consumer behavior. In market research, for instance, sentiment analysis enables companies to gauge customer opinions and feedback in real time, thereby facilitating product improvements and enhancing customer satisfaction. In the political arena, it is used to measure public sentiment toward candidates and policies, influencing campaign strategies and policy development.

Given the widespread applications and the need for accurate sentiment detection, a variety of models have been developed, ranging from traditional machine learning approaches like Naive Bayes and SVMs to more advanced deep learning techniques such as Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM), and transformers [6]. Each of these models has distinct advantages and limitations when applied to sentiment analysis tasks.

LSTMs are particularly effective at retaining sequential information over time, making them suitable for tasks that require the management of short to medium length sequences. Their gating mechanisms help mitigate issues like the vanishing gradient problem, yet they are less efficient for long sequences due to their sequential processing. This limitation becomes apparent with large datasets where LSTMs can struggle to capture long-range dependencies effectively. CNNs, on the other hand, excel at identifying local patterns within text, thanks to their ability to process data in

parallel across multiple filters. This computational efficiency makes CNNs a strong choice for tasks where local features are crucial. However, their fixed receptive fields limit their capacity to capture long-range dependencies, a task better suited for models like transformers.

Transformers represent a significant advancement in NLP by addressing many of the shortcomings of LSTMs and CNNs. Their self-attention mechanism allows for the modeling of both short- and long-range dependencies, independent of token distance within a sequence. This capability, combined with the parallelization of training, makes transformers highly scalable and efficient, particularly for large datasets. However, the self-attention mechanism quadratic complexity leads to substantial memory and computational requirements, especially when processing very long sequences.

Recently, a novel State Space Model (SSM) known as Mamba [3] was introduced. Mamba features a selective scan mechanism designed for context-dependent reasoning and incorporates hardware-aware optimizations that take advantage of Graphics Processing Unit (GPU) parallelization, allowing for efficient training. Unlike transformers, which typically require input padding to process sequences of varying lengths, Mamba offers greater flexibility by handling sequences of different lengths without padding. This capability, along with its ability to mitigate the vanishing gradient problem through orthogonal matrix constraints, positions Mamba as a promising alternative for long-sequence tasks.

In this study, a comprehensive evaluation of several prominent techniques in sentiment analysis is provided, focusing on LSTM, CNN, Bidirectional Encoder Representations from Transformers (BERT), Generative Pre-trained Transformer (GPT), and the innovative Mamba model. These models were integrated into an end-to-end neural network and their performance was compared on the Twitter US airline dataset [4] and the IMDB movie review dataset [5]. Through this comparative analysis, we aim to highlight the strengths and limitations of each model, particularly in

their ability to efficiently handle complex language patterns and varying sequence lengths, thereby contributing valuable insights to the ongoing development of sentiment analysis techniques in NLP.

Chapter 2 Background

2.1 Transformers

Transformer models [1, 2, 6] have emerged as a foundational architecture in the field of NLP. They rely on a self-attention mechanism which allows the model to weigh the importance of different words in a sequence when processing text to provide a robust understanding of context. This capability enables transformers to excel in complex tasks requiring long-term dependencies. As shown in Figure 2.1, a transformer is composed of an encoder and a decoder. A transformer encoder processes the input sequence and generates a contextual representation while the decoder takes this representation and generates an output sequence. Popular transformer-based models include BERT [1], which uses the encoder component of the transformer model and outputs a semantic representation of the input, and GPT [2], which uses the transformer decoder component and is known for its text generation capabilities. They are flexible, scalable, and also parallelizable on GPUs which enables fast training compared to traditional Recurrent Neural Networks (RNNs). However, transformer models also present challenges such as high computational cost and complexity.

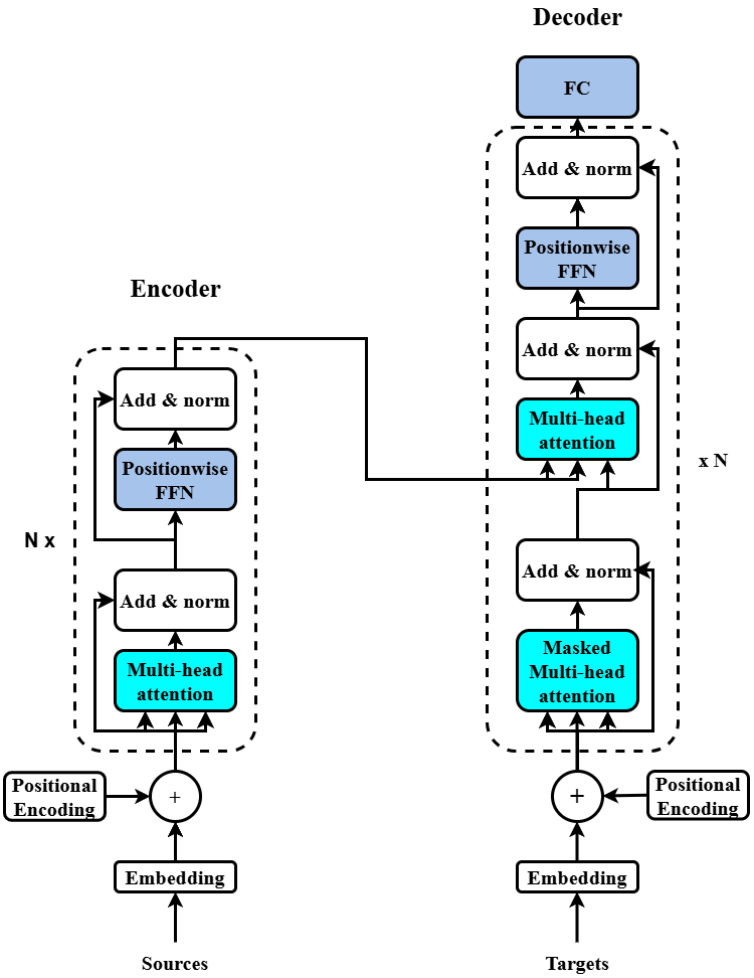

Figure 2.1: The transformer architecture.

2.2 State Space Models

Recently, SSMs [3,7] in deep learning have gained attention for their potential in sequence modeling and as an alternative to transformers. They are related to RNNs as they use recurrence on a hidden state. They represent a continuous system that maps an input $x(t)$ to an output $y(t)$ through an implicit latent state $h(t)$. SSMs can capture the temporal dependencies and dynamics that are crucial for accurate sentiment classification.

SSMs represent system states and predict subsequent states based on the inputs. As illustrated in Figure 2.2, they make predictions through two equations. The state equation describes how the state evolves given an input $x(t)$ and the previous hidden state $h(t)$. The output equation describes how the hidden state and input are translated into an output prediction. Matrices A, B, C, D are trainable parameters. Matrix A describes how the previous hidden state is transferred to the new hidden state. Matrix B describes how the input $x(t)$ is considered when updating $h(t)$. Matrix C determines how $h(t+1)$ is considered in calculating the output $y(t)$. Matrix D determines how the input $x(t)$ is transferred to the output $y(t)$. Matrix D does not affect how the hidden state is calculated and can be thought of as a skip connection.

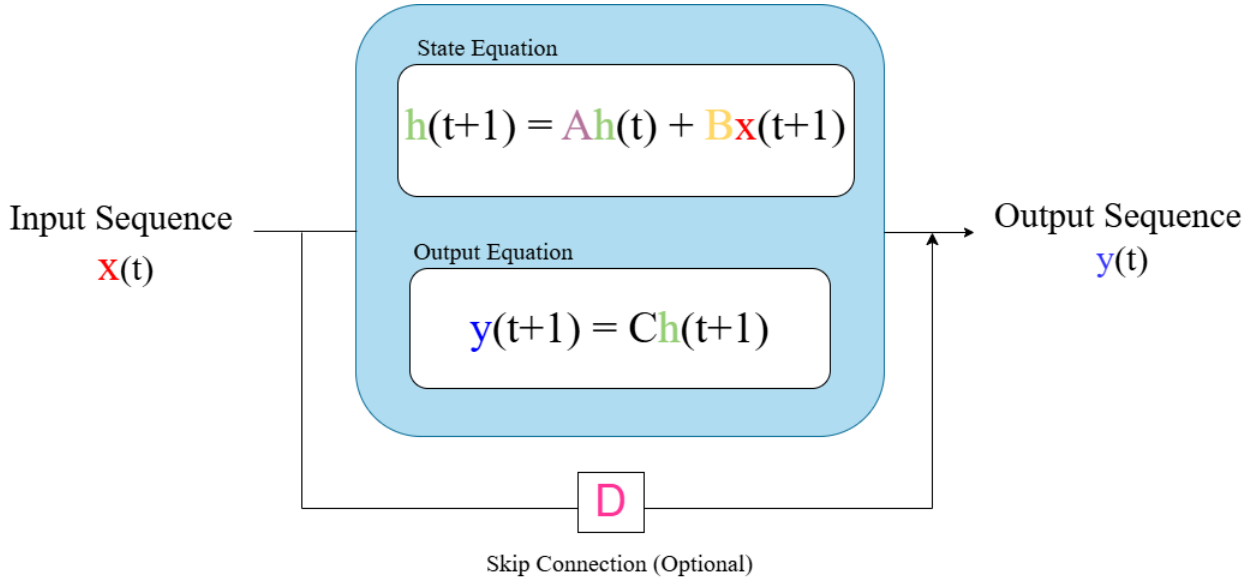

Figure 2.2: SSM transition equations for sequence processing.

2.2.1 Recurrent Representation

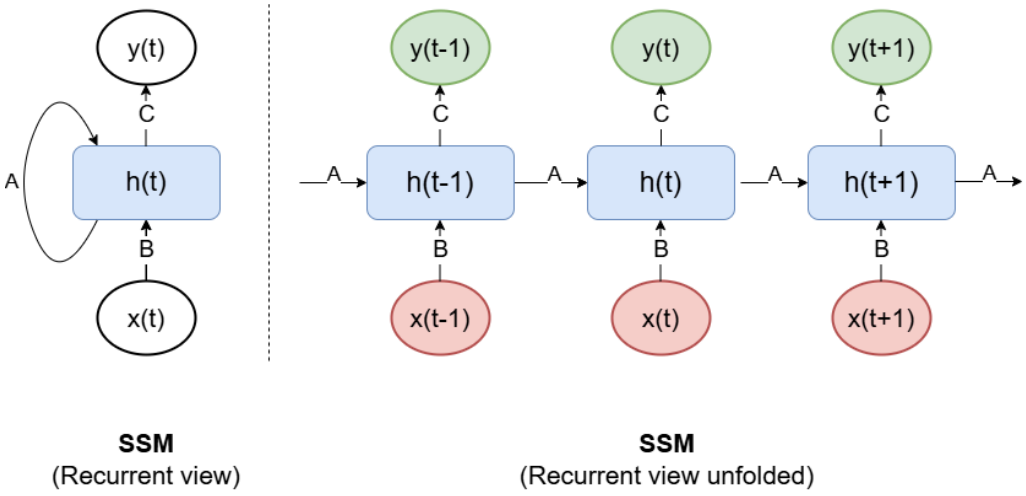

Figure 2.3: Recurrent representation of an SSM.

An SSM can be formulated as a linear RNN. Figure 2.3 shows an SSM unrolled over time similar to an RNN. The output of the SSM is fed back to itself, and the hidden state is updated over time. However, similar to a non-linear RNN or LSTM, the recurrent part of the SSM processes data sequentially, meaning each step depends on the output of the previous step. This dependency creates a chain-like structure that prevents parallel execution, making them less efficient on modern hardware architectures that leverage parallel computing capabilities to break down complex tasks into smaller, independent sub-tasks to be processed simultaneously.

2.2.2 Convolution Representation

The following equations demonstrate the process of deriving the SSM kernel from the SSM formulation

$$h(0) = Bx(0) \tag{2.1}$$
$$y(0) = Ch(0) = CBx(0) \tag{2.2}$$
$$h(1) = Ah(0) + Bx(1) = ABx(0) + Bx(1) \tag{2.3}$$
$$y(1) = CABx(0) + CBx(1) \tag{2.4}$$
$$h(2) = Ah(1) + Bx(1) = A^2Bx(0) + ABx(1) + Bx(2) \tag{2.5}$$
$$y(2) = CA^2Bx(0) + CABx(1) + CBx(2) \tag{2.6}$$
$$\begin{aligned} y(t) &= CA^tBx(0) + CA^{t-1}Bx(1) + \ldots + CABx(t) \\ &= k * x(t), \; k = (CB, CAB, \ldots, CA^tB) \end{aligned} \tag{2.7}$$

where $*$ denotes convolution. These equations indicate that, when calculating $y(t)$, we can avoid computing intermediate hidden states and output values by pre-calculating the kernel k. The final output $y(t)$ is obtained by convolving the input sequence with this kernel. This SSM convolutional kernel can be computed using a Fast Fourier Transform (FFT) [7] and can be parallelized during training, making it highly efficient for modern hardware like GPUs and Tensor Processing Units (TPUs). Moreover, by constraining the state space transition matrix A [7], low-rank approximations can be applied to further reduce the number of computations.

2.3 Related Work

Many sentiment analysis techniques rely on deep learning models due to their ability to capture intricate patterns and contextual dependencies in textual data. Models such as RNNs, LSTMs, CNNs, and transformers have been widely used. RNNs and LSTMs are particularly effective in handling sequential data and capturing long-term dependencies, which is crucial for understanding the sentiment conveyed across long texts. Although traditionally used in image processing, CNNs have shown prowess in extracting local features and patterns in text through their convolutional layers. More recently, transformer models such as BERT and GPT have set new benchmarks in sentiment analysis by leveraging self-attention mechanisms to capture global dependencies and context from large-scale pre-trained language representations. Moreover, many popular transformer models come with pre-trained weights [10, 11, 12] and have extensive libraries available, making them straightforward and easy to fine-tune.

Dhanalakshmi et al. [13] presented an analysis of CNNs in comparison with fully connected neural networks on the IMDB movie review dataset. The text was preprocessed to remove irrelevant symbols. They achieved an accuracy of 86% using CNNs and demonstrated superior performance compared to fully connected neural networks.

Uddin et al. [14] investigated the use of LSTMs for analyzing depression in social media data written in the Bangla language. Their study aimed to detect signs of depression from Bangla tweets using a small, stratified dataset. Through careful hyperparameter tuning, specifically by employing a 5-layer LSTM network with 128 units, the model demonstrated high accuracy in detecting depression.

Alahmary et al. [15] applied deep learning techniques to classify sentiments of Saudi dialect texts. They applied three techniques: Support Vector Machine (SVM), LSTMs, and bidirectional

LSTM on 32063 tweets. They found LSTM-based techniques outperform the SVM algorithm and that Bidirectional LSTM achieved 94% accuracy which is better than LSTM (92%).

Dhola et al. [16] conducted a comparative study to evaluate the performance of BERT compared to SVM, Naive Bayes, and LSTM on the Sentiment140 dataset which consists of 1.6 million Tweets. They preprocessed the data using tokenization, stemming, lemmatization, and stop-word and punctuation removal. They found that the BERT model outperformed the other models, achieving 85.4% accuracy.

Tan et al. [17] proposed a new hybrid model by combining two popular models, namely Robustly optimized BERT (RoBERTa) and LSTM, for sentiment analysis. RoBERTa maps the words into compact meaningful contextual embeddings while LSTM captures long-distance semantic context effectively. They achieved F1-scores of 93%, 91%, and 90% on the IMDB movie review dataset, Twitter US airline dataset, and Sentiment140 dataset, respectively. The results obtained illustrate the effectiveness and efficiency of hybrid models in accurately predicting text sentiments.

Chapter 3 Machine Learning

In this study, the goal is to evaluate the performance of various established models, namely LSTM, CNN, GPT, BERT, and Mamba. In this section, we discuss these models and their implementation.

3.1 Convolutional Neural Network

Originally designed for image processing, CNNs have shown remarkable effectiveness in NLP tasks due to their ability to capture local dependencies in input data, which is crucial when working with text. Convolutional kernels slide over a portion of the input text which allows CNNs to effectively detect local patterns of consecutive words, similar to capturing n-gram contexts. CNNs are also highly parallelizable, making them efficient to train on modern GPUs. While CNNs capture local patterns effectively, they might not capture long-range dependencies. The proposed CNN model for text classification leverages these strengths by embedding input text sequences into dense vectors, which are then processed by multiple convolutional layers with different filter sizes to capture various n-gram features. Each convolutional layer applies a 2D convolution followed by ReLU activation to generate feature maps, which are then pooled using max pooling to extract the most important features. The pooled features are concatenated, passed through a dropout layer to prevent overfitting, and finally processed by a fully connected layer for classification. This model effectively captures local dependencies and hierarchical structures in text, making it well-suited for tasks such as sentiment analysis and text categorization.

3.2 Long Short-Term Memory

An LSTM is a type of RNN designed for sequence modeling tasks. It creates an internal hidden state that captures temporal dynamics. This architecture makes LSTMs particularly well-suited for sequential tasks where the context of the input data is essential. LSTMs were developed to overcome the limitations of traditional RNNs, particularly the issues of vanishing and exploding gradients. It consists of memory cells to maintain the state over time helping to store and retrieve information across long sequences, an input gate to regulate the extent to which new information flows into the memory cell, a forget gate to determine the amount of information from the previous cell state to be forgotten, and an output gate to control the output based on the memory cell and current input.

The proposed LSTM-based model implementation for text classification begins by embedding the input text sequences into dense vectors. These vectors are fed into an LSTM layer that processes the sequences to capture both immediate and long-term dependencies within the text. The hidden states from the LSTM are averaged to create a fixed-size representation of the input sequence, which helps in maintaining the contextual information. This averaged output is then passed through a dropout layer to prevent overfitting, and finally through a fully connected layer for classification.

3.3 BERT

BERT is a transformer-based machine learning model designed by Google [10] to process and analyze large amounts of natural language data. It uses a bidirectional approach to read a

sequence of words using self-attention to capture the context of a word based on both the preceding and succeeding words. BERT undergoes a two-step process: pre-training and fine-tuning. In pre-training, BERT learns to predict masked words in sentences and determine if one sentence follows another, providing it with a robust understanding of language. During fine-tuning, BERT adapts to specific tasks, such as text classification, using an extra output layer. In the proposed implementation, the tokenized input is converted into embeddings and passed through multiple BERT encoder layers. The final hidden state corresponding to the classification token is used for class prediction through a dropout layer and a classification head.

3.4 GPT

GPT is an autoregressive language model developed by OpenAI [11] that excels in natural language processing tasks by predicting the next word in a sequence based on the preceding words. Its architecture is based on a transformer model and uses casual self-attention mechanisms to handle dependencies across tokens in the input text. The model processes a text sequence token by token, and generates one output at a time. Because each prediction depends on the previous words in the sequence, the model effectively remembers the context and uses it to inform subsequent predictions. In the GPT implementation, text data is tokenized, processed by the pre-trained GPT model from Hugging Face [11], and classified through a classification head tailored to the specific datasets. During fine-tuning, the model adjusts its weights based on the training data, optimizing its ability to predict class labels accurately.

3.5 MAMBA

Mamba [3], a linear time sequence model with selective state space, was introduced to overcome the computational inefficiencies of transformers when dealing with long sequences. Unlike transformers, whose memory and computational costs scale quadratically with sequence length, Mamba has a streamlined end-to-end neural network architecture that omits attention and MLP blocks, resulting in faster inference and linear scaling with sequence length. The key contributions of Mamba are its selective scan algorithm and hardware-aware approach.

3.5.1 Selective Scan Algorithm

Common SSMs typically use fixed dynamic transitions (e.g., the matrices A, B, and C), that prevent them from selecting relevant information from their context or deciding, based on the input, what information to pass through the hidden state. Consequently, traditional SSMs such as S4 [7] struggle with context-aware reasoning, which is essential for NLP tasks. In contrast, as depicted in Figure 3.1, Mamba can dynamically decide which parts of the input sequence to pay attention to and which parts to ignore, allowing it to efficiently predict a meaningful output.

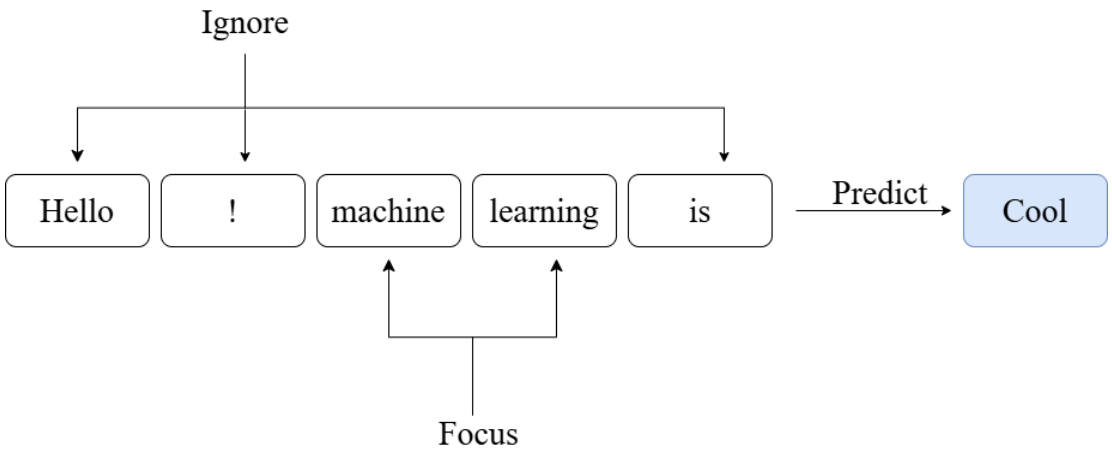

Figure 3.1: Dynamic transitions selectively retain information.

Mamba addresses this limitation by making the matrices B and C, as well as the step size Δ, input-dependent by incorporating the sequence length and batch size of the input. This input dependency allows the Δ to selectively determine what to retain in the hidden state. A larger step size enables the model to emphasize the current input word over the context, while a smaller step size favors maintaining the previous context over the current word. The selective scan algorithm ensures that the step size, B, and C are input-dependent, making them unsuitable for pre-computation using the convolutional approach. Mamba uses a parallel associative scan to perform the prefix sum operation, improving its efficiency in handling context-aware reasoning in NLP tasks.

3.5.2 Hardware-aware Algorithm

It was identified in [3] that a significant limitation of recent GPUs that hinders the performance of SSMs is the limited transfer (I/O) speed between small but fast static random-access memory (SRAM) and large but slow high-bandwidth memory (HBM). Inspired by flash attention [18], Mamba reduces the frequency of transfers between HBM and SRAM. This is achieved by loading SSM parameters directly from the slow HBM into the fast SRAM, where discretization and recurrence are performed, and then writing the final output back to HBM. Since data copying in GPUs is much slower than computation, Mamba avoids saving intermediate states used for backpropagation, instead recomputing them during the backward pass. Consequently, the selective scan layer has the same memory efficiency as an optimized transformer utilizing flash attention [18].

3.5.3 Structured State Space Model

To address the vanishing gradient problem commonly encountered in RNNs, many prior SSMs impose constraints on the matrix A, such as enforcing orthogonality or unitarity. These constraints help ensure that the hidden states remain stable as the model processes sequences. Drawing inspiration from High-order Polynomial Projection Operators (HiPPO) theory [9], Mamba initializes the A matrix as an orthogonal HiPPO matrix given by

$$A_{nk} = \begin{cases} (2n+1)^{1/2}(2k+1)^{1/2} & \text{- entries below the diagonal} \\ n+1 & \text{- entries on the diagonal} \\ 0 & \text{- entries above the diagonal} \end{cases} \quad (3.1)$$

where n is the row index and k is the column index. The orthogonality of A helps prevent the vanishing gradient problem, while the HiPPO initialization enables more stable updates of hidden states during sequence processing.

As shown in Figure 3.2, the proposed Mamba implementation comprises an embedding layer, multiple stacks of Mamba blocks, an average pooling layer, and an output classifier. The embedding layer transforms input tokens into dense vectors. The output from the Mamba blocks, which represents a sequence of latent states for each token, is averaged to produce a fixed-size representation of the entire sequence. This representation, encapsulating the overall contextual information, is then passed through a fully connected layer for classification.

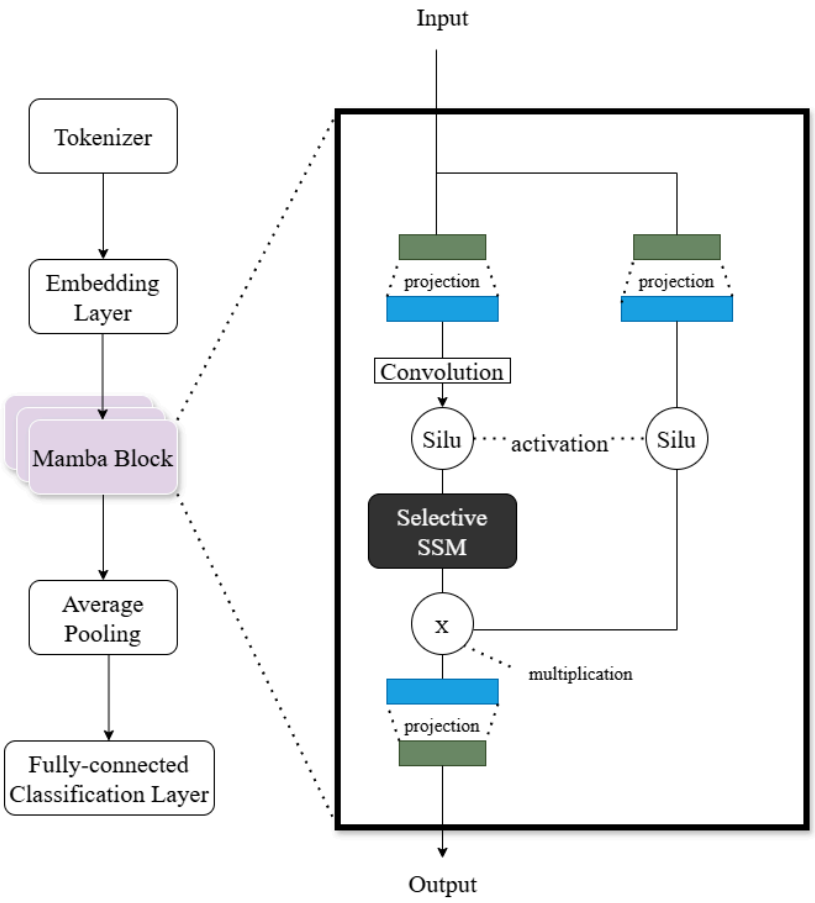

Figure 3.2: The Mamba classifier implementation.

Stacks of Mamba blocks, akin to stacks of transformer blocks, enhance the model ability to capture increasingly complex features and dependencies within the data, enabling it to better represent intricate relationships and patterns. Each Mamba block begins with a linear projection for state expansion, followed by convolution to aggregate information from neighboring tokens, thereby avoiding independent token calculations. SiLU activation functions introduce non-linearity to help the model learn more complex patterns. The selective scan mechanism is incorporated within the selective SSM cell.

Table 3.1 presents a comparison of the CNN, LSTM, Mamba, BERT, and GPT models based on their context dependency, training speed, computation cost, and sequence handling. CNN excels

in capturing local dependencies with small receptive fields, while LSTM, Mamba, and GPT capture unidirectional context and long-range dependencies, and BERT captures bidirectional context. In terms of training speed, CNN, Mamba, BERT, and GPT benefit from parallelism, making them faster, whereas LSTM is slower due to its sequential processing. Regarding computation cost, CNN is efficient, with performance dependent on filter size, while Mamba and LSTM scale linearly with sequence length. In contrast, BERT and GPT have higher computational costs, scaling quadratically with sequence length. Additionally, CNN requires a fixed-size input, whereas LSTM, Mamba, BERT, and GPT can handle variable-length sequences.

Aspect	**Dependency Capture**	**Training Speed**	**Computation Cost**	**Handle Sequence Length**
CNN	Local dependencies (n-grams)	Fast due to CNN parallelism	Efficient and dependent on the number of filters and kernel size.	Fixed-size input handling required
LSTM	Unidirectional context and long-range dependencies	Slow due to sequential processing	Linearly scaling to sequence length	handles variable-length sequences
Mamba	Unidirectional context and long-range dependencies	Fast due to parallelism with associative scan algorithm	Linearly scaling to sequence length [3]	handles variable-length sequences
BERT	Bidirectional context and dependency	Fast due to parallelization	Quadratic scaling to sequence length [19]	handles variable-length sequences up to 512 tokens [10]
GPT	Unidirectional context and long-range dependency	Fast due to parallelization	Quadratic scaling to sequence length [19]	handles variable-length sequences up to 1024 tokens [11]

Table 3.1: Model comparison based on dependency, speed, cost, and sequence handling.

Chapter 4 Experiment Setup

4.1 Datasets

The Twitter US airline dataset and the IMDB movie review dataset are used to evaluate the five models. The Twitter US airline dataset is a widely recognized benchmark for sentiment analysis in NLP. It was collected from Twitter, offering real-world examples of customer opinions about airline services. The dataset comprises 14,640 samples, with 3,099 neutral, 2,363 positive, and 9,178 negative sentiments. We divided the dataset into 70% for training, 15% for validation, and 15% for testing. The IMDB movie review dataset consists of movie reviews from the Internet Movie Database (IMDB) and was designed for binary sentiment classification tasks. It includes 25,000 samples in the training subset and 25,000 samples in the testing subset, with each subset containing an equal number of positive and negative reviews. The reviews vary in length, from a few sentences to paragraphs containing thousands of words, posing a realistic challenge for NLP models. For our experiments, 20,000 samples were randomly chosen from the training dataset for training and the remaining 5,000 samples were used for validation. Figure 4.1 illustrates the sequence length distribution of the IMDB movie review dataset and Twitter US airline dataset.

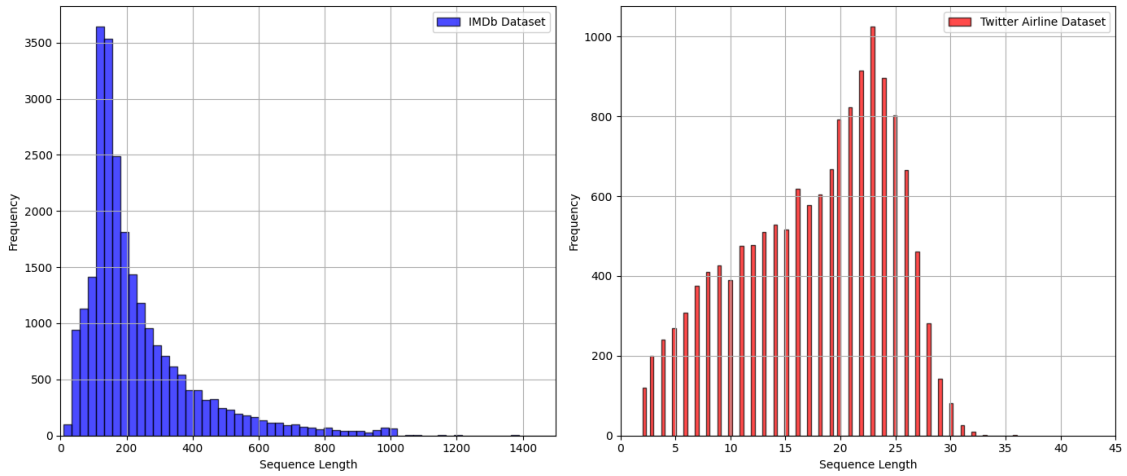

Figure 4.1: Sequence length distribution of the IMDB movie review dataset and Twitter US airline dataset.

4.2 Experiment Settings

Tokenizers were employed to convert raw text into numerical representations suitable for machine learning models. They were configured to pad or truncate sequences to a maximum length of 512 tokens.

The LSTM and CNN models were trained from scratch, whereas the BERT, GPT, and Mamba models were initialized with pre-trained weights, specifically, BERT-base, GPT-2-small, and Mamba-130M obtained from the Hugging Face repositories [10, 11, 12]. All models were trained using the AdamW optimizer over 50 epochs, with early stopping applied if the validation loss did not improve for 5 consecutive epochs, a weight decay rate of 0.01, and a cosine annealing learning rate scheduler. To prevent overfitting, a dropout layer with a rate of 0.1 was applied to all models prior to the final classification layer.

Each model was trained on both the Twitter US airline dataset and the IMDB movie review dataset. Performance was evaluated based on test loss, accuracy, precision, recall, F1-score, training time per epoch, evaluation time per epoch, and the number of epochs required for convergence. Accuracy, precision, recall, and F1-score are defined as

$$\text{Accuracy} = \frac{(TP+TN)}{TP+FP+TN+FN} \tag{4.1}$$

$$\text{Precision} = \frac{TP}{(TP+FP)} \tag{4.2}$$

$$\text{Recall} = \frac{TP}{(TP+FN)} \tag{4.3}$$

$$\text{F1-score} = \frac{2TP}{2\,TP+FP+FN} \tag{4.4}$$

where True Positive (TP) is the number of instances where the model correctly predicts that a sample belongs to a class, True Negative (TN) is the number of instances where the model correctly predicts that a sample does not belong to a class, False Negative (FN) is the number of instances where the

model incorrectly predicts that a sample does not belong to a class, and False Positive (FP) is the number of instances where the model incorrectly predicts that a sample belongs to a class.

4.2.3 Twitter US Airline Dataset

In the experiments with the CNN architecture, a range of hyperparameter settings were explored to optimize the model performance. Embedding dimensions of 256, 512, and 768 were tested, along with filter size configurations (2, 3), (2, 3, 4), (2, 3, 4, 5), and (2, 3, 4, 5, 6). The number of filters considered were 256, 512, and 768. For the LSTM architecture, the same embedding dimensions were used (256, 512, and 768), and networks with 1, 2, 4, 8, and 12 LSTM layers were examined. Initially, the best hyperparameters for both CNN and LSTM models were found using a learning rate of 3×10^{-5} and a batch size of 32. We then fine-tuned the batch size and learning rate based on preliminary results. Additionally, experiments were conducted using the Mamba model trained from scratch with the same configuration as the LSTM, and the validation losses were compared. By systematically exploring these hyperparameters, the optimal configurations for CNN and LSTM models in capturing sentiment from the datasets were determined.

4.2.4 IMDB Movie Review Dataset

The optimal CNN and LSTM model hyperparameters identified through experiments on the Twitter US Airline dataset were applied to the IMDB movie review dataset. This allowed for an evaluation of model generalizability across different domains of sentiment analysis. While the

Twitter dataset focuses on short, informal texts related to customer service experiences, the IMDB dataset consists of longer, more detailed reviews. By applying the same hyperparameter configurations, we were able to assess how well the models, optimized for airline-related sentiment, perform on movie review data, which involves more complex language and diverse sentiment expressions. Table 4.1 gives the model parameter settings including the number of parameters, number of layers, and the hidden dimension sizes.

	Parameters	Layers	Dimensions
Mamba-130M	129M	24	768
GPT-2-small	124M	12	768
BERT-base	109M	12	768
LSTM	33M	2	768
CNN	24M	3	768

Table 4.1: Model parameter settings.

4.3 Fine Tuning

In the experiment on the Twitter US airline dataset, we fine-tuned the pre-trained Large Language Models (LLMs), specifically Mamba, BERT, and GPT, using an initial learning rate of 3×10^{-6}. For the CNN and the LSTM models, we experimented with learning rates of 3×10^{-4}, 1×10^{-4}, 3×10^{-5}, and 1×10^{-5}. When experimenting on the IMDB movie review dataset, we fine-tuned the LLMs using learning rates 9×10^{-7}, 3×10^{-7}, and 1×10^{-7}. The CNN and LSTM models were trained with learning rates of 3×10^{-5}, 1×10^{-5}, and 3×10^{-6}. During training for both datasets, we monitored validation loss and validation accuracy as primary indicators of model performance. If we observed overfitting (where validation loss increased while training

loss decreased) or underfitting (where both training and validation loss remained high or validation accuracy plateaued), we adjusted the learning rate by multiplying it by a factor of 3 for overfitting or by $\frac{1}{3}$ for underfitting. To determine the maximum batch size, we tested sizes that are powers of 2, incrementally doubling them until the memory limits of the GPU were reached.

4.4 Loss function

The cross-entropy loss function was used to measure the performance of the classification models. This function quantifies the difference between the predicted probability distribution and the actual label distribution, increasing as the predicted probabilities deviate from the true labels. The cross-entropy loss is defined as

$$\sum_{i=1}^{N}\sum_{j=1}^{C}(y_{i,j}\log((p_{i,j}))) \tag{4.5}$$

where C is the number of classes, N is the number of samples, $y_{i,j}$ is the true label for instance i in class j, and $p_{i,j}$ is the corresponding predicted probability.

4.5 Experiment Environment

All experiments were conducted on an Ubuntu system with 64GB of memory and an NVIDIA RTX4070 (12GB) GPU with CUDA version 11.8.

Chapter 5 Results and Discussion

5.1 Twitter US Airline Dataset

For the CNN and LSTM models, we systematically tested multiple configurations to optimize their performance on the Twitter US airline sentiment dataset. Figure 5.1 presents the validation loss of the CNN architecture for various combinations of filter size, number of filters, and embedding dimensions. These results show that increasing the embedding dimension from 256 to 768 leads to lower validation loss for all configurations. Larger filter sizes consistently performed worse, suggesting that they may cause the model to lose detailed information and impair generalization capabilities. For the number of filters, it was observed that a moderate quantity (512 filters) provided the best validation performance. Increasing the number of filters beyond this did not necessarily improve CNN performance, especially with a large embedding dimension. The model with 256 filters of sizes 2, 3, and 4 and an embedding dimension of 768 exhibited the lowest validation loss.

Figure 5.2 presents a comparison of the validation loss for both the LSTM and Mamba architectures with different numbers of layers and embedding dimensions. For the LSTM model, an increase in the number of layers increases the validation loss, especially when the embedding dimension is large, suggesting potential overfitting. In contrast, the Mamba model exhibits less sensitivity to the number of layers and generally outperforms the LSTM model, highlighting Mamba's ability to effectively manage deeper architectures. The optimal LSTM configuration was identified as two layers with an embedding dimension of 768.

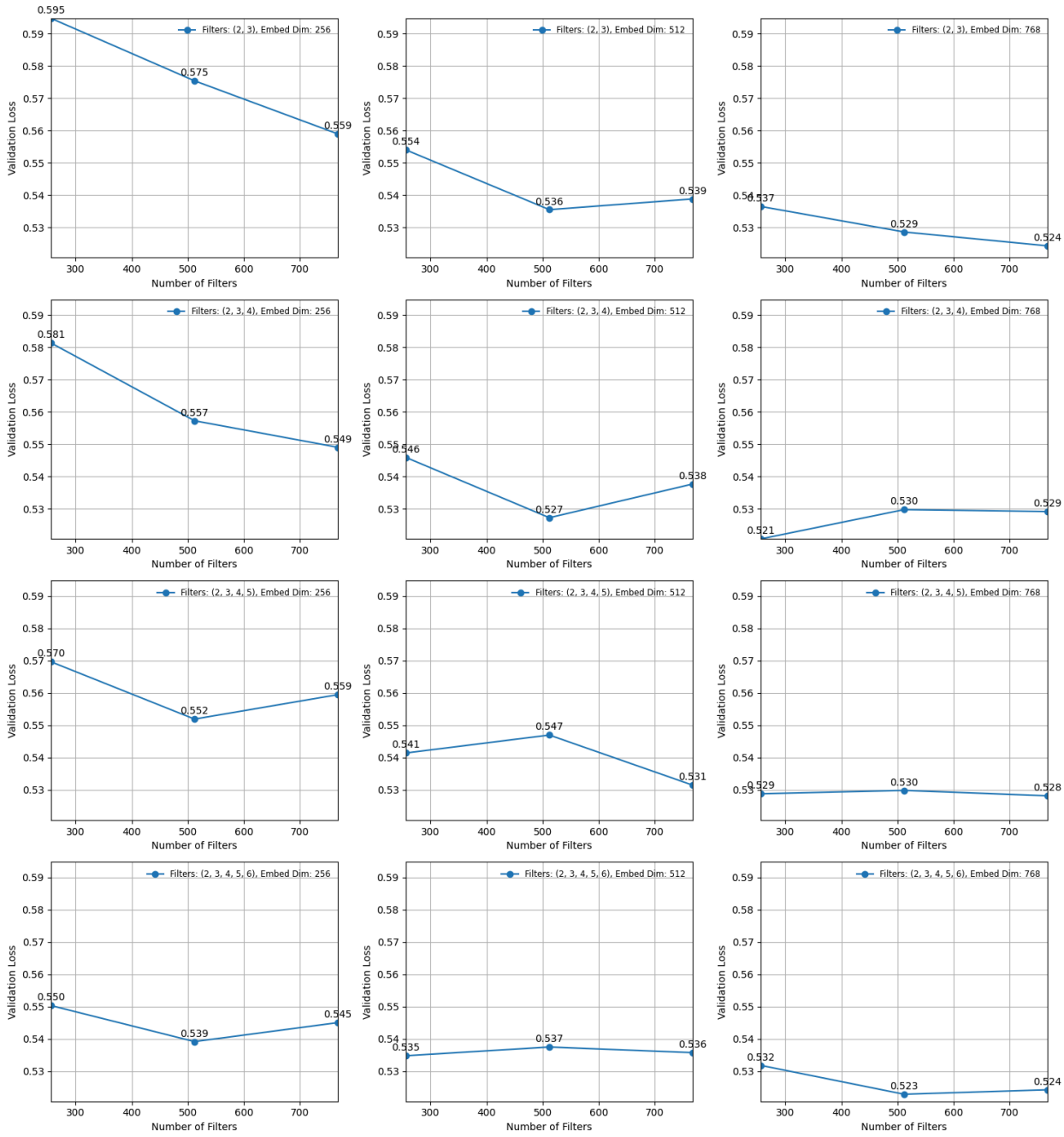

Figure 5.1: CNN validation loss using different CNN filter sizes, embedding dimensions, and numbers of CNN filters.

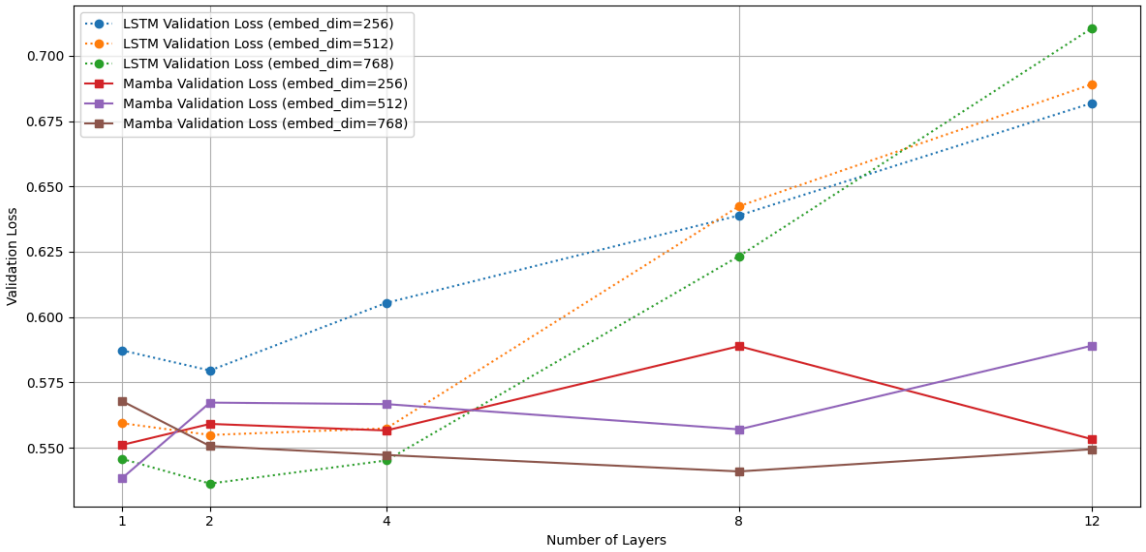

Figure 5.2: LSTM and Mamba validation loss with different numbers of layer and embedding dimensions.

Figure 5.3 shows that the CNN model achieved the fastest training time with a batch size of 512, while the LSTM model trained most efficiently using a batch size of 256. Regarding the maximum batch sizes for LLMs, it was observed that GPT could handle a maximum batch size of 32 without encountering GPU memory issues, whereas BERT and Mamba could manage batch sizes up to 64. This indicates that the GPT model has higher memory consumption compared to BERT and Mamba. To ensure a fair comparison among all models, we standardized the batch size to 32 for training on the Twitter US airline dataset.

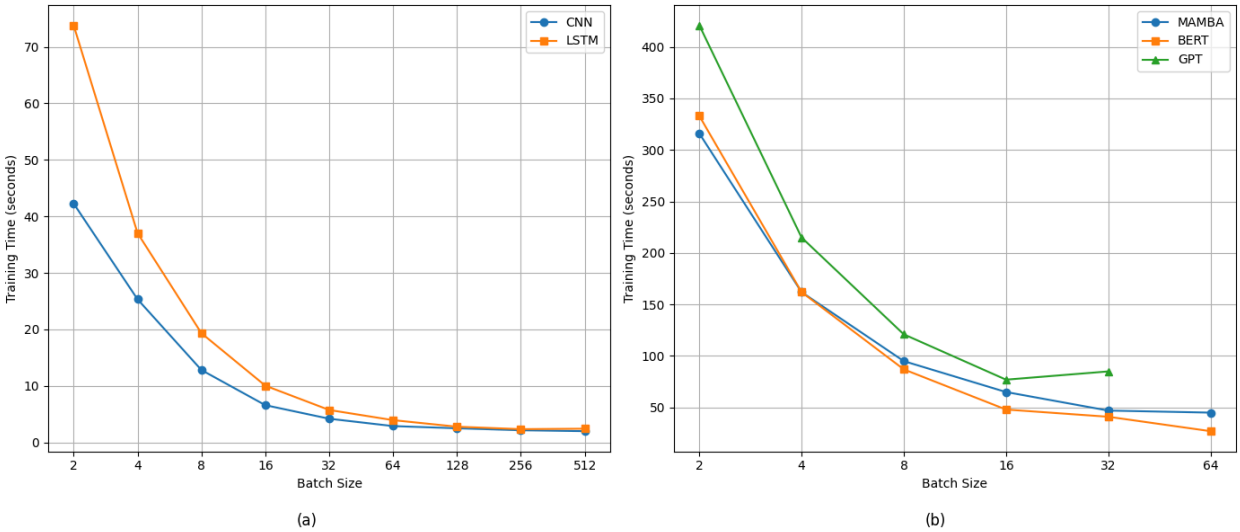

Figure 5.3: Training time per epoch versus batch size for the Twitter US airline dataset with the (a) CNN and LSTM, and (b) Mamba, BERT, and GPT models.

Figures 5.5 and 5.6 illustrate the training and validation losses of the CNN and LSTM models, respectively, with the optimal configurations from Figures 5.1 and 5.2, and different learning rates. The CNN model consists of three layers with kernel sizes 2, 3, and 4, an embedding dimension of 768, and 256 filters. The LSTM model comprises two layers with an embedding dimension of 768. For both models, a learning rate of 3×10^{-5} provided the best balance between training speed and convergence.

Figures 5.7, 5.8, and 5.9 present the training and validation losses of the Mamba, BERT, and GPT models, respectively, when trained with a batch size of 32 and various learning rates. These results show that the Mamba and GPT models achieved the best balance between convergence and generalization at a learning rate of 3×10^{-6}, while the BERT model performed optimally at a learning rate of 9×10^{-7}.

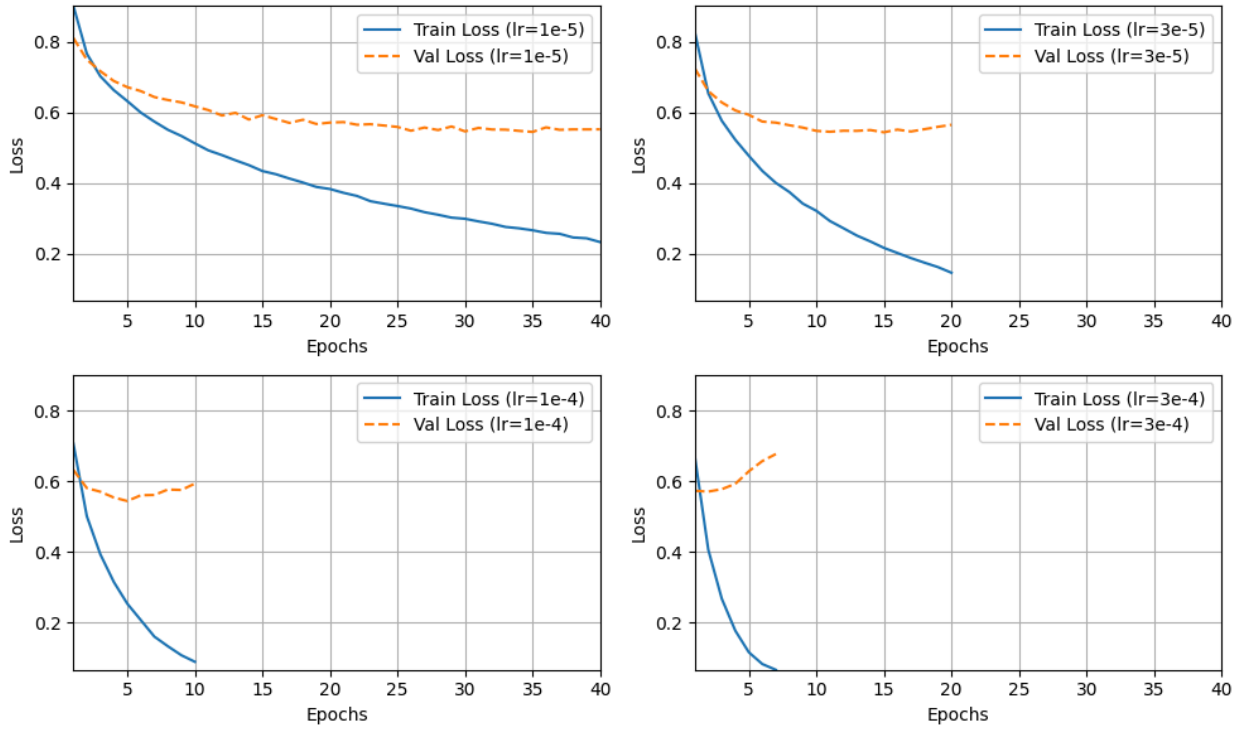

Figure 5.4: CNN training and validation loss for the Twitter US airline dataset.

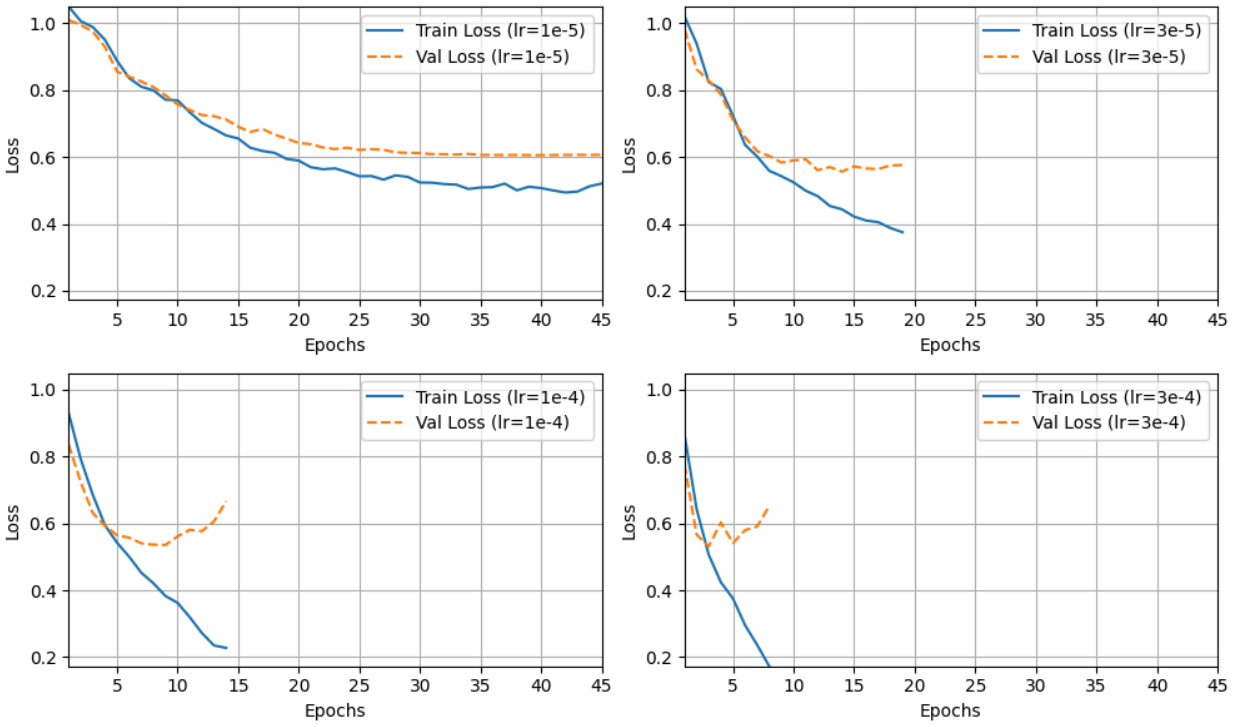

Figure 5.5: LSTM training and validation loss for the Twitter US airline dataset.

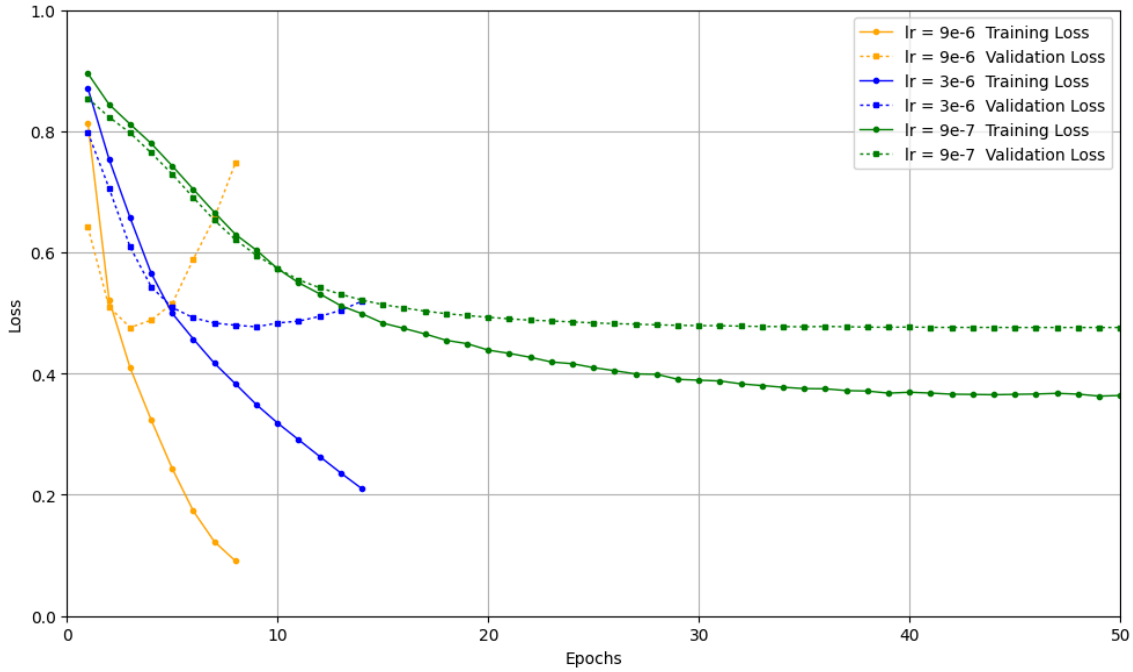

Figure 5.6: Mamba training and validation loss for the Twitter US airline dataset.

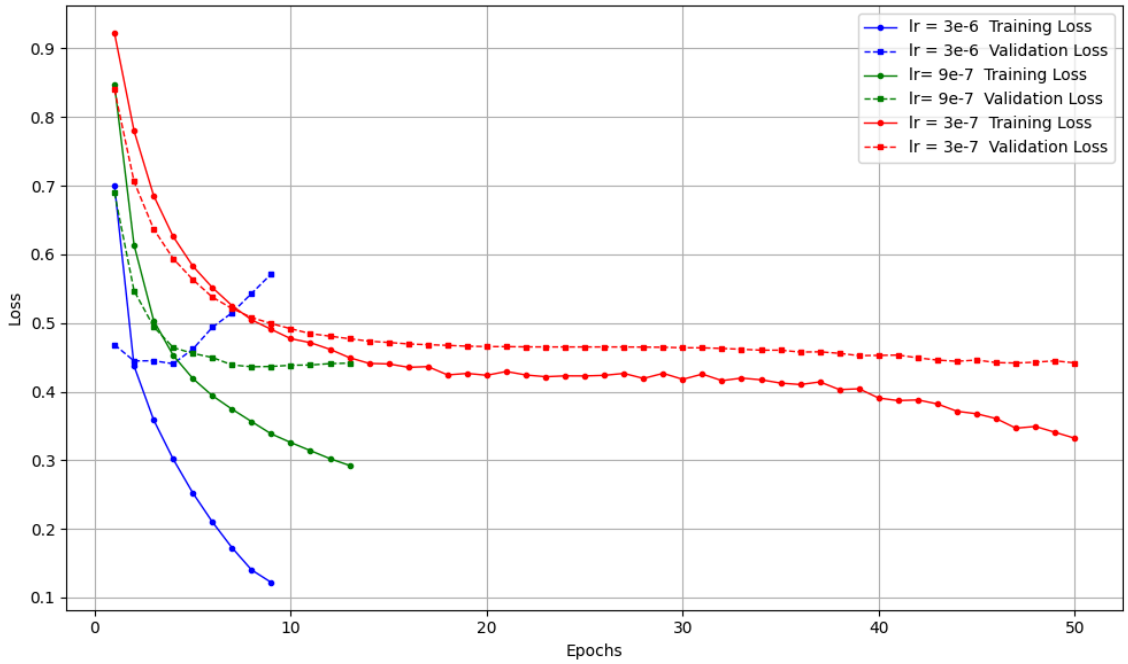

Figure 5.7: BERT training and validation loss for the Twitter US airline dataset.

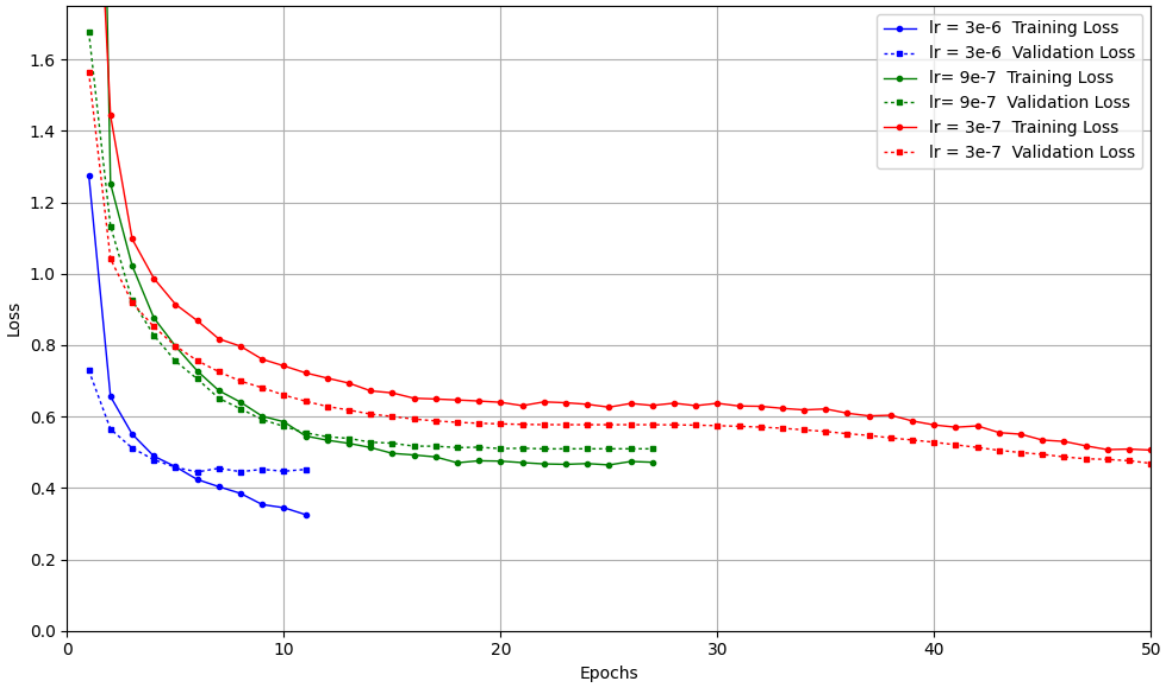

Figure 5.8: GPT training and validation loss for the Twitter US airline dataset.

The results given in Table 5.1 for the Twitter US airline dataset reveal distinct performance differences among the models. BERT leads with the highest test accuracy (83.9%), F1-score (83.8%), recall (83.9%), and precision (83.8%), indicating its superior ability to classify sentiment in this dataset. Mamba and GPT follow closely with similar performance, both achieving around 82% test accuracy and F1-scores around 81%. LSTM and CNN have lower test accuracies of 76.2% and 78.6%, respectively, reflecting their reduced effectiveness on this task compared to the transformer-based models. In terms of test loss, BERT also has the lowest value (0.426), reinforcing its efficiency in minimizing errors. Mamba and GPT have slightly higher test losses (0.459) and so are competitive. In contrast, LSTM and CNN exhibit noticeably higher test losses, with LSTM reaching 0.566 and CNN 0.532, which correlates with their comparatively lower accuracy.

Regarding training time, CNN is the fastest model with a training time of just 0.60 min, attributed to its minimal training time per epoch (0.05 min) and fast evaluation time. However, this comes at the cost of lower performance. LSTM also trains relatively quickly, with a total training

time of 1.04 min, but it has the lowest accuracy among the models. BERT, on the other hand, balances efficiency and performance, with a training time of 3.60 min and a training time per epoch of 0.45 min. Despite Mamba and GPT requiring slightly longer training times per epoch (0.75 min for Mamba and 1.28 min for GPT), their total training times are still reasonable at 6.75 and 7.68 min, respectively. BERT achieves optimal performance in just 8 epochs, indicating its efficiency in reaching peak accuracy faster than the other models. Mamba requires 9 epochs, while GPT reaches optimality in 6 epochs. LSTM and CNN require 13 and 15 epochs, respectively, despite their lower overall accuracy.

In summary, when classifying sentiments on the Twitter US airline dataset, BERT outperforms the other models, offering the best accuracy and lowest loss with relatively efficient training. Mamba and GPT also perform well, while LSTM and CNN provide faster training times but with lower accuracy, making them suitable for quicker, though less precise, sentiment analysis tasks.

Twitter US airline	Mamba	GPT	BERT	LSTM	CNN
Test loss	0.459	0.459	0.426	0.566	0.532
Test accuracy	82.3%	82.0%	83.9%	76.2%	78.6%
F1-score	82.2%	81.8%	83.8%	75.6%	77.5%
Recall	82.3%	82.0%	83.9%	76.2%	78.6%
Precision	82.1%	81.8%	83.8%	75.3%	78.6%
Learning rate	3×10^{-6}	3×10^{-6}	9×10^{-7}	3×10^{-5}	3×10^{-5}
Training time per epoch (min)	0.75	1.28	0.45	0.08	0.05
Evaluation time per epoch (min)	0.08	0.05	0.05	0.01	0.01
Epochs to optimality	9	6	8	13	15
Total training time (min)	6.75	7.68	3.60	1.04	0.60

Table 5.1: Experiment results for the Twitter US airline dataset.

5.2 IMDB Movie Review Dataset

Similar to Section 5.1, to ensure a fair comparison between models, a batch size of 8 is used for training all models with the IMDB movie review dataset.

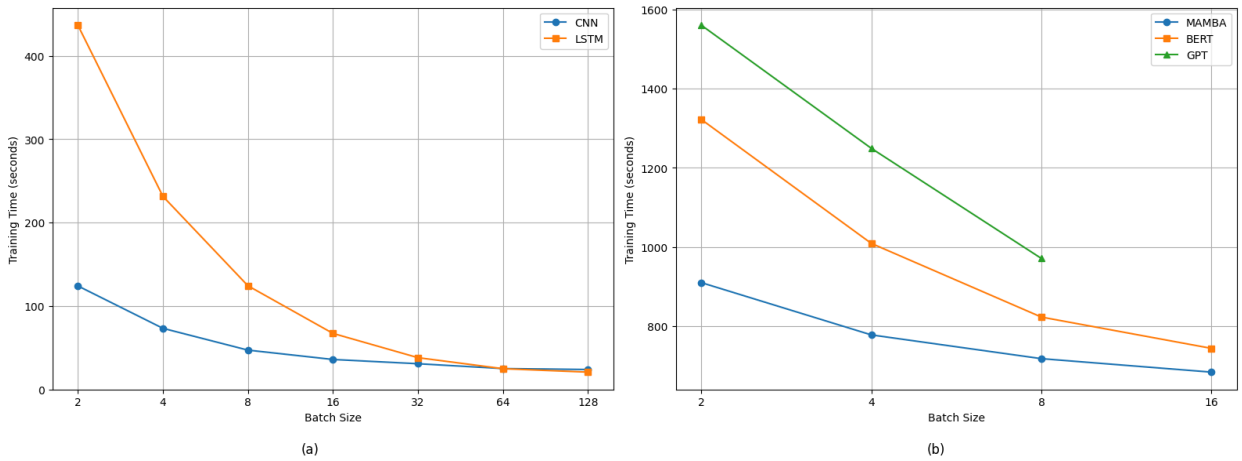

Figure 5.9: Training time per epoch versus batch size for the IMDB movie review dataset with the (a) CNN and LSTM, and (b) Mamba, BERT, and GPT models.

Figures 5.10 to 5.14 show the training and validation loss of the five models using different learning rates. For the CNN and LSTM models, a learning rate of 1×10^{-5} offers the best balance between training speed and convergence, while a learning rate of 3×10^{-7} works better for the Mamba, BERT, and GPT models.

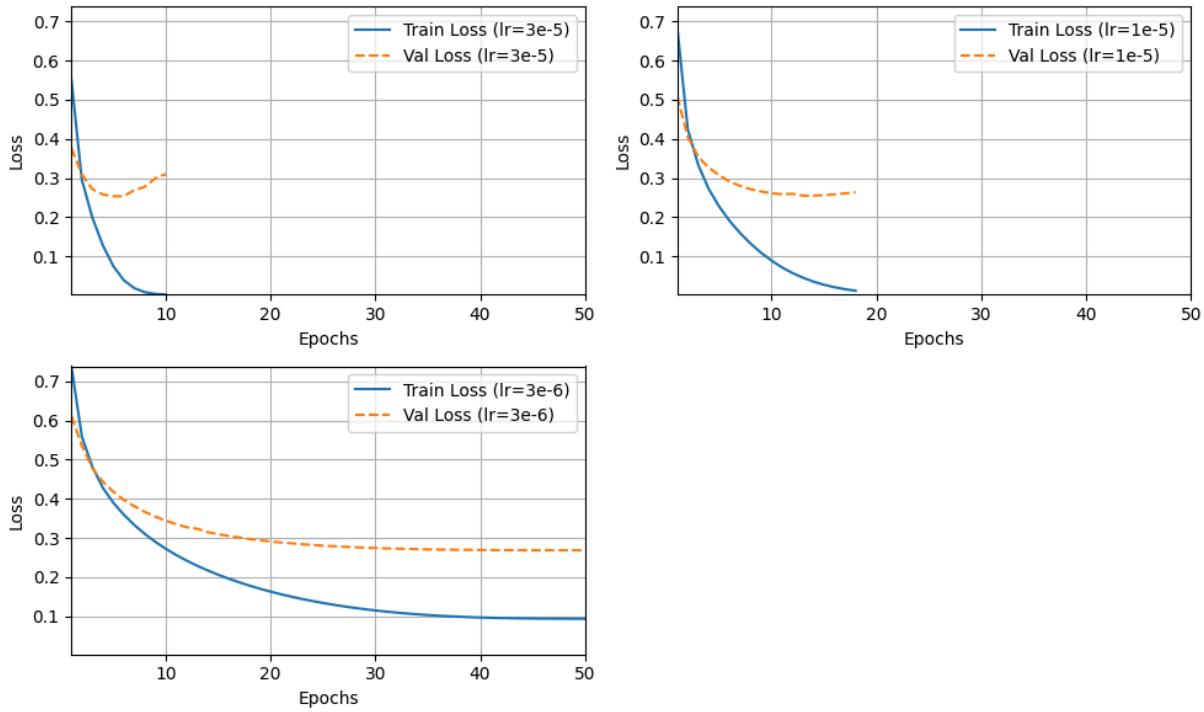

Figure 5.10: CNN training and validation loss for the IMDB movie review dataset.

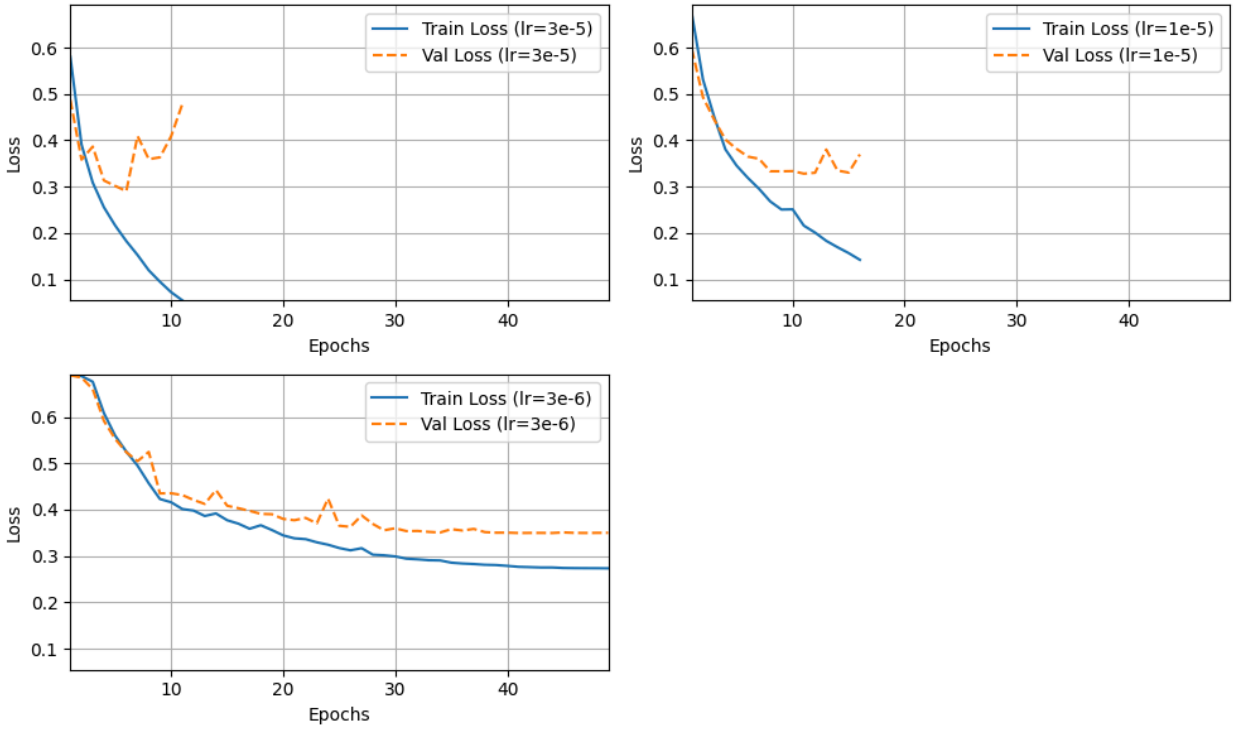

Figure 5.11: LSTM training and validation loss for the IMDB movie review dataset.

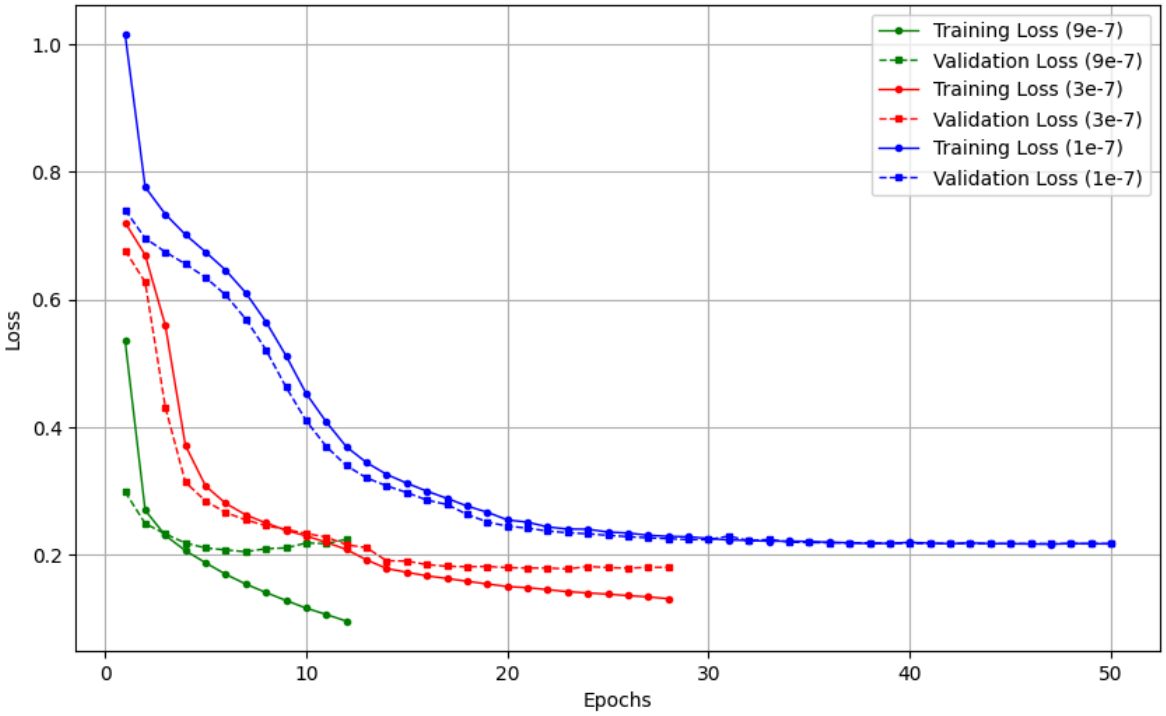

Figure 5.12: Mamba training and validation loss for the IMDB movie review dataset.

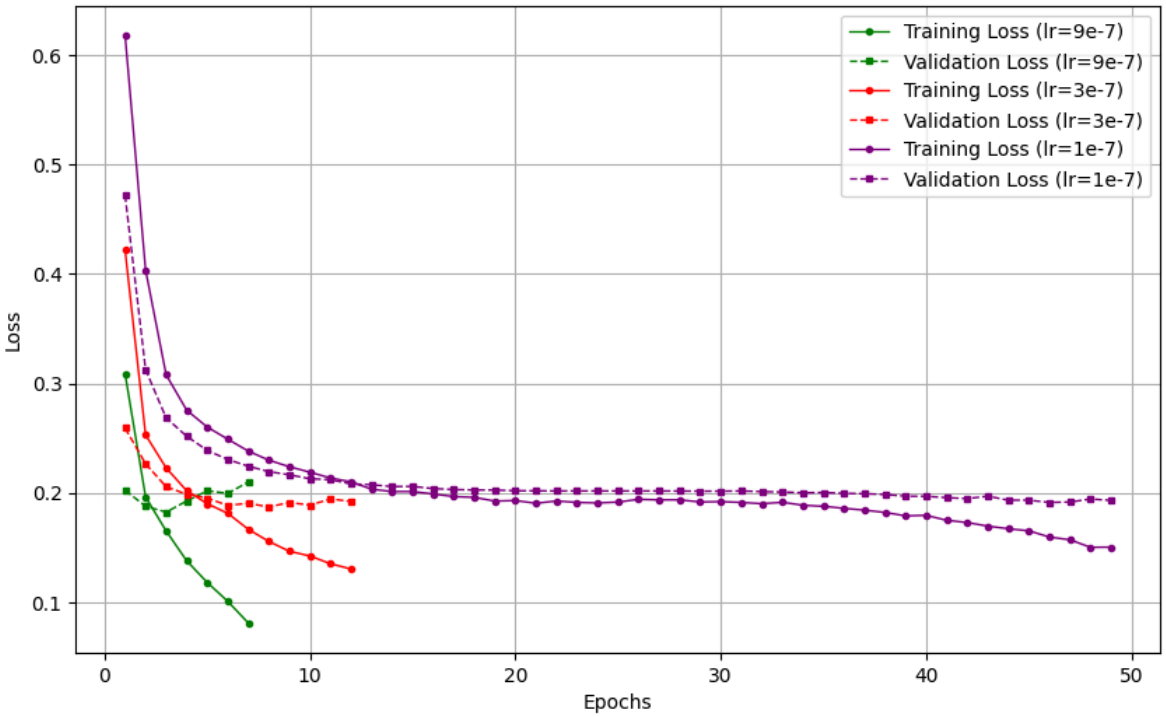

Figure 5.13: BERT training and validation loss for the IMDB movie review dataset.

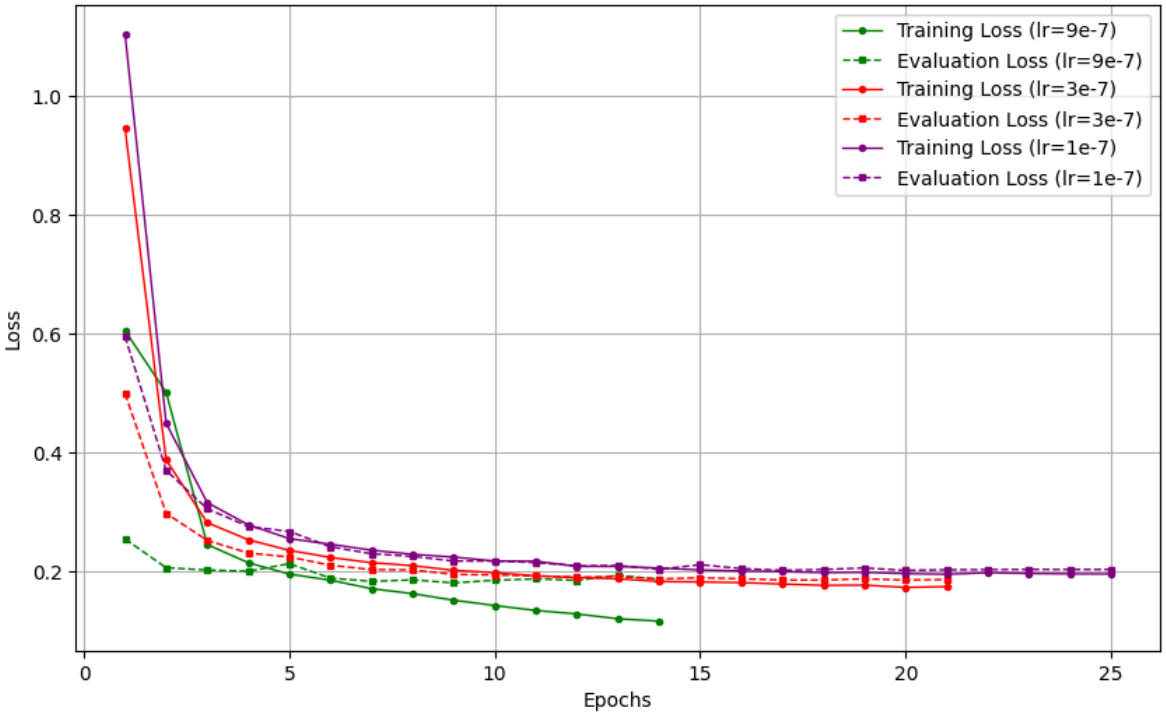

Figure 5.14: GPT training and validation loss for the IMDB movie review dataset.

Table 5.2 presents the sentiment analysis results using LSTM, CNN, Mamba, GPT, and BERT as backbone models for classification on the IMDB movie review dataset. Among these models, BERT demonstrates the best overall performance, achieving the highest test accuracy (93.9%). Mamba closely follows with a test accuracy of 93.6%, showcasing it as a competitive alternative to traditional deep learning models. GPT also performs well, with a slightly lower test accuracy of 92.8%. The simpler models, LSTM and CNN, are behind with test accuracies of 86.3% and 89.9%, respectively, reflecting their more limited ability to capture complex patterns in the dataset.

When examining the total training time, a significant difference between the models is observed. GPT required the longest total training time (275.91 min), followed by Mamba (233.45 min) and BERT (107.60 min). In contrast, LSTM and CNN were notably faster, with training times

of 23.30 min and 16.80 min, respectively. This demonstrates that while the larger transformer-based models offer higher accuracy, they have a considerably greater computational cost in terms of training time.

The epoch-level data reveals interesting differences. Despite GPT having the longest training time per epoch (16.23 min) and evaluation time (4.10 min), it reached optimal performance after 17 epochs, contributing to its overall total training time. BERT, which has a similar epoch duration (13.45 min), required only 8 epochs to achieve optimal performance, significantly reducing the total training time compared to GPT and Mamba. In contrast, while LSTM and CNN were much faster per epoch, they needed more epochs to reach optimal performance. Notably, CNN required 24 epochs, yet its fast epoch times (0.75 min for training and 0.24 min for evaluation) enabled it to have the shortest total training time. LSTM required only 10 epochs, but its relatively moderate epoch duration (2.33 min) led to a longer total training time than CNN.

In summary, BERT and Mamba offer the best performance in terms of accuracy and test loss but this comes with increased computational costs. BERT efficiency in reaching optimal performance with fewer epochs makes it an appealing choice for balancing accuracy and training time. Conversely, while LSTM and CNN are faster to train, their lower accuracy and poorer performance limit their effectiveness for the task considered, especially in comparison to the more advanced models.

IMDB movie review	Mamba	GPT	BERT	LSTM	CNN
Test loss	0.179	0.185	0.176	0.343	0.264
Test accuracy	93.6%	92.8%	93.9%	86.3%	89.9%
F1-score	93.6%	92.8%	93.9%	86.3%	89.9%
Precision	93.6%	92.8%	93.9%	86.3%	89.9%
Recall	93.6%	92.8%	93.9%	86.3%	89.9%
Learning rate	3×10^{-7}	3×10^{-7}	3×10^{-7}	1×10^{-5}	1×10^{-5}
Training time per epoch (mins)	10.15	16.23	13.45	2.33	0.75
Evaluation time per epoch (mins)	3.05	4.10	3.52	0.89	0.24
Epochs to optimality	23	17	8	10	24
Total training time (mins)	233.45	275.91	107.60	23.30	16.80

Table 5.2 Experiment results for the IMDB movie review dataset.

5.3 Discussion

LSTMs are a powerful model used to handle long-term dependencies in NLP tasks. However, due to their sequential nature, LSTMs limit parallelization during both forward propagation and backpropagation which contributes to higher computational costs. Even though the LSTM gating mechanism helps to mitigate the vanishing gradient problem, the results show that when stacking multiple LSTM layers, the model performance drops significantly as shown in Figure 5.2. Specifically, the LSTM model demonstrated the highest test loss and the lowest accuracy across both the Twitter US airline dataset and the IMDB movie review dataset, as evidenced by the results in

Tables 5.1 and 5.2. Additionally, the gates and memory cells in LSTMs require significant memory to store intermediate states and gradients, leading to high computational cost during training.

The CNN implementation uses a similar number of parameters as the LSTM model, but achieved better performance with lower test loss and higher accuracy on both datasets. This aligns with the CNN ability to capture local patterns more effectively. As shown in Tables 5.1 and 5.2, the CNN outperformed the LSTM model of a similar size, particularly in terms of training time per epoch and total training time, which highlights its efficiency in handling shorter sequences with local dependencies.

When compared to LSTM and CNN, the BERT and GPT models, which leverage transformer-based architectures, outperformed both, achieving lower test losses and higher accuracy. These models benefit from the ability to process sequences in parallel and effectively model complex relationships in the data, as reflected in their superior F1-scores, precision, and recall across both datasets.

Mamba was designed to address the limitations of LSTMs. By incorporating a selection mechanism within SSMs, Mamba enables the model to retain relevant information more effectively. Unlike the LSTM heuristic gating mechanism, the Mamba selection mechanism is more general, allowing for more robust parameterization and initialization. The orthogonal constraint on the Mamba transition matrix helps prevent the vanishing gradient problem. As reflected in the results, Mamba performance closely matches that of the BERT and GPT models, particularly on the IMDB movie review dataset, where it achieved a comparable F1-score, precision, and recall. Additionally, Mamba training and evaluation times per epoch were significantly lower than those of the GPT model, demonstrating its efficiency. However, Mamba required more epochs to converge, indicating a slower learning process compared to transformer-based models.

An intriguing observation emerges from the IMDB movie review dataset where the Mamba training time per epoch is less than that of transformer-based models, unlike the results seen with the Twitter US airline dataset. This suggests that Mamba gains an advantage when handling longer sequences. Furthermore, Mamba does not require input padding, offering greater flexibility when dealing with sequences of varying lengths, whereas the transformer architectures BERT and GPT have maximum input sequence length constraints (512 tokens for BERT and 1024 tokens for GPT). This characteristic reduces memory usage and computational costs, particularly in tasks involving highly variable sequence lengths, such as those found in movie reviews. The ability to dynamically adjust to sequence length further enhances its performance in complex NLP tasks without the limitations imposed by predefined token constraints. This makes Mamba a more flexible choice for tasks that require processing long, unstructured text data.

For the purposes of this study, the maximum sequence length was set to 512 tokens across all models to ensure a fair comparison. However, in practical applications, models like Mamba and LSTM could benefit from their ability to handle longer sequences, potentially surpassing transformer-based architectures in tasks requiring the processing of lengthy or complex input data.

Chapter 6 Conclusion and Future Work

6.1 Conclusion

Sentiment analysis remains a pivotal task in the field of Natural Language Processing (NLP) due to its wide-ranging applications across various domains, including market research, social media monitoring, and political analysis. This study explored the effectiveness of different architectures in sentiment analysis, specifically the LSTM, CNN, BERT, GPT, and Mamba models. LSTMs, while capable of capturing long-term dependencies through their unique gating mechanisms, are limited by their sequential nature, leading to challenges in scalability and efficiency. Conversely, CNNs excel in capturing local patterns and are highly parallelizable, yet they struggle with modeling long-range dependencies. The results presented show that LSTMs and CNNs are outperformed by the transformer models BERT and GPT. BERT and GPT, which excel in capturing complex relationships and dependencies within data, have set new standards in sentiment analysis due to their ability to capture contextual information through attention mechanisms. The newer model, Mamba, is an SSM that addresses the computational inefficiencies associated with transformers by offering a linear scaling solution. Its innovative selective scan mechanism allows Mamba to process sequences efficiently while maintaining context-dependent reasoning, making it a strong contender for sentiment analysis tasks. The results obtained demonstrate that Mamba is less susceptible to the vanishing gradient problem compared to LSTM. It achieves accuracy and F1-scores comparable to BERT and GPT, with the advantage of efficient training time and evaluation time per epoch.

This work underscores the importance of selecting appropriate models based on specific task requirements and dataset characteristics. While transformer models currently lead in performance,

emerging models like Mamba offer promising alternatives for NLP, particularly when computational efficiency and scalability are critical considerations. Overall, this research contributes to a deeper understanding of sentiment analysis model architectures, providing valuable insights into their strengths and limitations. As sentiment analysis continues to evolve, the results of this study will guide researchers and practitioners in choosing the most suitable models for their applications.

6.2 Future Work

To improve the understanding of Mamba capabilities, it would be beneficial to evaluate the architecture on other text classification datasets as well as token classification tasks. Such an assessment would provide a more comprehensive view of its generalizability and robustness across diverse NLP tasks. Given its linear scalability with sequence length, future research could focus on evaluating memory usage and effectiveness on significantly longer sequences, particularly those exceeding 1,000 tokens. This would allow for deeper insights into Mamba's ability to handle tasks involving long documents or extensive textual data.

With the recent release of Mamba-V2 [8], which integrates a linear attention mechanism, there is an opportunity for further improvements in sentiment analysis. Mamba-V2 reportedly achieves processing speeds 2 to 8 times faster than the original Mamba while maintaining comparable performance. This enhanced efficiency could make Mamba-V2 a compelling alternative to traditional transformer models, especially in scenarios where computational resources are constrained or when handling long sequences is a critical requirement. The potential of Mamba-V2 highlights the need for evaluation to fully explore its potential and determine its place in the evolving landscape of NLP models.

Bibliography

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, MN, USA, pp. 4171-4188, 2019.

[2] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language Models are Unsupervised Multitask Learners,” OpenAI Blog, 2019, Available: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

[3] A. Gu and T. Dao, “Mamba: Linear-Time Sequence Modeling with Selective State Spaces,” arXiv:2312.00752, 2024.

[4] O. Sanseviero, “twitter-airline-sentiment,” Datasets at Hugging Face, 2015. Available: https://huggingface.co/datasets/osanseviero/twitter-airline-sentiment.

[5] “STANFORDNLP/imdb,” Datasets at Hugging Face, 2011. Available: https://huggingface.co/datasets/stanfordnlp/imdb.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention Is All You Need,” arXiv:1706.03762, 2023.

[7] A. Gu, K. Goel, and C. Ré, “Efficiently Modeling Long Sequences with Structured State Spaces,” arXiv:2111.00396, 2022.

[8] T. Dao and A. Gu, “Transformers are SSMs: Generalized Models and Efficient Algorithms Through Structured State Space Duality,” arXiv:2405.21060, 2024.

[9] A. Gu, I. Johnson, A. Timalsina, A. Rudra, and C. Ré, “How to Train Your HiPPO: State Space Models with Generalized Orthogonal Basis Projections,” arXiv:2206.12037, 2022.

[10] “google-bert/bert-base-uncased,” huggingface.co, Jan. 18, 2024. Available: https://huggingface.co/google-bert/bert-base-uncased

[11] “openai-community/gpt2,” huggingface.co. Available: https://huggingface.co/openai-community/gpt2

[12] “Mamba,” huggingface.co, 2014. Available: https://huggingface.co/docs/transformers/en/model_doc/mamba.

[13] S. Dhanalakshmi et al. “Sentiment Analysis and Classification Using Convolutional Neural Network Architecture,” International Conference on Technological Advancements in Computational Sciences, Tashkent, Uzbekistan, pp. 836-841, 2022.

[14] A. H. Uddin, D. Bapery, and A. S. M. Arif, “Depression Analysis from Social Media Data in Bangla Language using Long Short Term Memory (LSTM) Recurrent Neural Network Technique,” International Conference on Computer, Communication, Chemical, Materials and Electronic Engineering, Rajshahi, Bangladesh, 2019.

[15] R. M. Alahmary, H. Z. Al-Dossari, and A. Z. Emam, “Sentiment Analysis of Saudi Dialect Using Deep Learning Techniques,” International Conference on Electronics, Information, and Communication, Auckland, New Zealand, 2019.

[16] K. Dhola and M. Saradva, “A Comparative Evaluation of Traditional Machine Learning and Deep Learning Classification Techniques for Sentiment Analysis,” International Conference on Cloud Computing, Data Science & Engineering, Noida, India, pp. 932-936, 2021.

[17] K. L. Tan, C. P. Lee, K. M. Lim, and K. S. M. Anbananthen, “Sentiment Analysis With Ensemble Hybrid Deep Learning Model,” IEEE Access, vol. 10, pp. 103694-103704, 2022.

[18] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. Ré, “FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness,” arXiv:2205.14135, 2022.

[19] “Weights & Biases,” W&B, 2024. Available: https://wandb.ai/wandb_fc/tips/reports/The-Problem-with-Quadratic-Attention-in-Transformer-Architectures--Vmlldzo3MDE0Mzcz