TEXT-BASED EMOTION ANALYSIS: APPROACHES AND EVALUATIONS

A Project Report submitted in the partial fulfillment of
the Requirements for the award of the degree

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING
Submitted by

K. Thrylokya   **(21471A05H1)**

B. Sirisha   **(21471A05J7)**

S. Bhagya Latha   **(21471A05J4)**

Under the esteemed guidance of

Y. Chandana M. Tech

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
NARASARAOPETA ENGINEERING COLLEGE: NARASAROPET
(AUTONOMOUS)
Accredited by NAAC with A+ Grade and NBA under Tyre -1 NIRF rank in the band of 201-300 and an ISO 9001:2015 Certified
Approved by AICTE, New Delhi, Permanently Affiliated to JNTUK, Kakinada
KOTAPPAKONDA ROAD, YALAMANDA VILLAGE, NARASARAOPET- 522601
2024-2025

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project that is entitled with the name **"TEXT-BASED EMOTION ANALYSIS: APPROACHES AND EVALUATIONS"** is a bonafide work done by the team K. Thrylokya (21471A05H1), B. Sirisha (21471A05J7), S. Bhagya Latha (21471A05J4) in partial fulfilment of the requirements for the award of the degree of BACHELOR OF TECHNOLOGY in the Department of COMPUTER SCIENCE AND ENGINEERING during 2024-2025.

PROJECT GUIDE   PROJECT-COORDINATOR

Y. Chandana M. Tech   **Dr. M. Sireesha M.Tech., Ph.D.,**

Asst. Professor   **Assoc. Professor**

HEAD OF THE DEPARTMENT   EXTERNAL EXAMINER
Dr. S. N. Tirumala Rao, M.Tech., Ph.D.,

Professor & HOD

DECLARATION

We declare that this project work titled **"TEXT-BASED EMOTION ANALYSIS: APPROACHES AND EVALUATIONS"** is composed by ourselves that the work contain here is our own except where explicitly stated otherwise in the text and that this work has not been submitted for any other degree or professional qualification except as specified.

K. Thrylokya   **(21471A05H1)**

B. Sirisha   **(21471A05J7)**

S. Bhagya Latha   **(21471A05J4)**

ACKNOWLEDGEMENT

INSTITUTE VISION AND MISSION

INSTITUTION VISION

To emerge as a Centre of excellence in technical education with a blend of effective student centric teaching learning practices as well as research for the transformation of lives and community.

INSTITUTION MISSION

M1: Provide the best class infra-structure to explore the field of engineering and research.

M2: Build a passionate and a determined team of faculty with student centric teaching, imbibing experiential, innovative skills.

M3: Imbibe lifelong learning skills, entrepreneurial skills, and ethical values in students for addressing societal problems.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION OF THE DEPARTMENT

To become a centre of excellence in nurturing the quality Computer Science & Engineering professionals embedded with software knowledge, aptitude for research and ethical values to cater to the needs of industry and society.

MISSION OF THE DEPARTMENT

The department of Computer Science and Engineering is committed to

M1: Mould the students to become Software Professionals, Researchers and Entrepreneurs by providing advanced laboratories.

M2: Impart high quality professional training to get expertise in modern software tools and technologies to cater to the real time requirements of the industry.

M3: Inculcate teamwork and lifelong learning among students with a sense of societal and ethical responsibilities.

Program Specific Outcomes (PSO's)

PSO1: Apply mathematical and scientific skills in numerous areas of Computer Science and Engineering to design and develop software-based systems.

PSO2: Acquaint module knowledge on emerging trends of the modern era in Computer Science and Engineering

PSO3: Promote novel applications that meet the needs of entrepreneur, environmental and social issues.

Program Educational Objectives (PEO's)

The graduates of the program are able to:

PEO1: Apply the knowledge of Mathematics, Science and Engineering fundamentals to identify and solve Computer Science and Engineering problems.

PEO2: Use various software tools and technologies to solve problems related to academia, industry, and society.

PEO3: Work with ethical and moral values in multi-disciplinary teams and can communicate effectively among team members with continuous learning.

PEO4: Pursue higher studies and develop their career in the software industry.

Program Outcomes (PO'S)

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. **Problem analysis:** Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. **Environment and sustainability:** Understand the impact of professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and teamwork**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. **Life-long learning**: Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Project Course Outcomes (CO'S)

CO421.1: Analyse the System of Examinations and identify the problem.

CO421.2: Identify and classify the requirements.

CO421.3: Review the Related Literature.

CO421.4: Design and Modularize the project.

CO421.5: Construct, Integrate, Test and Implement the Project.

CO421.6: Prepare the project Documentation and present the Report using appropriate method.

Course Outcomes – Program Outcomes mapping

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
C421.1		✓											✓		
C421.2	✓		✓		✓								✓		
C421.3				✓		✓	✓	✓					✓		
C421.4		✓				✓	✓	✓					✓	✓	
C421.5					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
C421.6									✓	✓	✓		✓	✓	

Course Outcomes – Program Outcome correlation

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
C421.1	2	3											2		
C421.2			2		3								2		
C421.3				2		2	3	3					2		
C421.4			2			1	1	2					3	2	
C421.5					3	3	3	2	3	2	2	1	3	2	1
C421.6									3	2	1		2	3	

Note: The values in the above table represent the level of correlation between CO's and PO's

1. Low level
2. Medium level
3. High level

Project mapping with various courses of Curriculum with Attained PO's:

Name of the course from which principles are applied in this project	Description of the device	Attained PO
C2204.2, C22L3.2	Focuses on data preprocessing, machine learning, and algorithm implementation.	PO1, PO3
CC421.1, C2204.3, C22L3.2	Utilizes natural language processing, sentiment analysis, and feature extraction techniques.	PO2, PO3
CC421.2, C2204.2, C22L3.3	Covers the integration of machine learning and deep learning models for solution development.	PO3, PO5, PO9
CC421.3, C2204.3, C22L3.2	Emphasizes feature engineering and optimization of algorithms.	PO1, PO5
CC421.4, C2204.4, C22L3.2	Highlights research methods and model evaluation techniques, ensuring effective communication.	PO10
CC421.5, C2204.2, C22L3.3	Explores real-world applications, project management, and ethical considerations.	PO10, PO11
C2202.2, C2203.3, C1206.3, C3204.3, C4110.2	Integrates interdisciplinary knowledge for addressing societal and environmental impacts.	PO4, PO7
C32SC4.3	Applies advanced methodologies and tools for model deployment and evaluation.	PO5, PO6

ABSTRACT

Emotions significantly shape human behaviour, influencing interactions, decision making and overall well-being. Text-based Emotion detection provides valuable insights for industries and healthcare by enabling personalized services and aiding mental health diagnoses. Leveraging the ISEAR dataset, which comprises seven emotional categories, this study proposes a hybrid model combining Convolutional Neural Networks (CNN), Bidirectional Gated Recurrent Units (BiGRU), and Support Vector Machines (SVM). The integration of these techniques addresses the complexity of emotional expression in textual data, achieving an accuracy of 86%. The results demonstrate the model's potential for real-world applications, including customer interaction enhancement and mental health evaluation. The proposed hybrid model benefits from CNNs feature extraction, BiGRUs contextual understanding, and SVMs classification capabilities. Through preprocessing techniques and innovative architecture, the study advances Natural Language Processing (NLP) methodologies for text-based emotion analysis. This research not only highlights the effectiveness of hybrid models but also establishes a foundation for future developments in emotion recognition systems, showcasing their broad applicability in diverse domains.

INDEX

LIST OF FIGURES

LIST OF TABLES

1.INTRODUCTION

Emotions play a crucial role in human behavior, influencing relationships, decision-making, and overall cognitive ability. Understanding and recognizing emotions is essential for various applications, ranging from improving customer support services to diagnosing mental health conditions. As the volume of textual data shared through online platforms, emails, user feedback, and digital interactions continues to rise, there has been a significant effort to develop automated systems capable of extracting emotions from text documents. Emotion recognition and analysis have gained prominence in the field of Natural Language Processing (NLP), providing valuable insights into the sentiments expressed in textual data. These insights are beneficial for businesses, healthcare professionals, and researchers alike, helping them better understand human emotions and interactions [1].

The challenge of emotion detection lies in the diverse ways feelings are expressed in text. Emotions can be explicitly conveyed using direct words that indicate a specific emotional state, such as "happy" or "angry." However, they can also be implicitly expressed through sentence structure, punctuation marks, and contextual cues. This complexity makes emotion detection in textual data a challenging yet essential task. In NLP, both emotion detection and sentiment analysis serve as vital tools for understanding the affective attitude behind linguistic expressions. These tools help marketers analyze customer feedback, assist healthcare professionals in diagnosing mental health conditions, and enable organizations to enhance user experiences through improved emotional intelligence in communication [2].

Although explicit emotional words are a significant part of written communication, implicit emotion expression plays an equally important role. The way sentences are framed, the use of punctuation marks such as exclamation points or ellipses, and the overall tone of a message contribute to the conveyed emotion. To study emotions in text, the widely used ISEAR (International Survey on Emotion Antecedents and Reactions) dataset is employed in this project. This dataset consists of seven primary emotions—happiness, fear, anger, sadness, disgust, shame, and guilt—offering a structured framework for analyzing emotion expressions in textual data [3].

By leveraging this dataset, researchers can gain deeper insights into how emotions are communicated through words and context. The ISEAR dataset has been

extensively used in research, demonstrating its effectiveness in emotion classification tasks across various linguistic and cultural contexts [4].

Given the rapid increase in textual information generated through social networks, emails, and web-based reviews, the need for automated emotion recognition has grown significantly. Sentiment analysis, which is often used interchangeably with emotion detection, focuses on determining the polarity (positive, negative, or neutral) of a given text. However, emotion recognition goes a step further by identifying specific emotions, making it more nuanced and valuable for applications such as mental health assessment and user experience enhancement [5].

To address the challenges of detecting emotions in text, a hybrid model integrating Convolutional Neural Networks (CNN), Bidirectional Gated Recurrent Units (BiGRU), and Support Vector Machines (SVM) is proposed. Each of these techniques brings unique advantages to the model. CNNs are effective in extracting local feature representations from text, BiGRU helps capture long-term dependencies and contextual relationships in sequences, and SVM serves as a reliable classification method. The combination of these techniques results in a highly efficient emotion recognition model, achieving an impressive accuracy of 86% in identifying emotions from textual data [6].

The model follows a structured workflow, beginning with preprocessing steps such as tokenization, stop word removal, and lemmatization using Natural Language Toolkit (NLTK) tools. Once the text data is cleaned and refined, it is transformed into numerical representations using Term Frequency-Inverse Document Frequency (TF-IDF) vectorization and Word2Vec embeddings. These transformations ensure that the textual information is converted into meaningful numerical features that can be effectively processed by machine learning and deep learning algorithms. The dataset is then divided into training and testing sets, where different machine learning and deep learning models are trained and evaluated. Performance metrics such as accuracy, precision, and recall are used to assess the efficiency of the models in classifying emotions correctly [7].

Emotion recognition has significant practical applications across various industries. In the business sector, it enables companies to build stronger customer relationships by understanding and responding to customer sentiments more effectively. Businesses can optimize their marketing strategies by analyzing customer emotions and tailoring advertisements accordingly. Moreover, automated emotion detection in

customer feedback allows companies to enhance product development by identifying consumer pain points and preferences [8]. In the healthcare field, emotional recognition can aid in diagnosing and managing mental health conditions by analyzing patients' emotional expressions in their written communication [9]. Studies have shown that text-based emotion detection can help in early detection of depression, anxiety, and other psychological disorders by analyzing patients' online posts and communication patterns.

Additionally, emotion recognition has implications for social media monitoring, where it can help detect cyberbullying, suicidal tendencies, or distress in users. By leveraging NLP techniques, platforms can implement automated moderation tools that identify and flag potentially harmful content based on emotional tone [10]. Furthermore, emotion detection is valuable in educational settings, where analyzing students' written responses can help educators understand their emotional engagement and well-being.

This project marks a significant advancement in NLP by providing a robust and efficient method for recognizing emotions in text. By leveraging state-of-the-art deep learning and machine learning techniques, this approach not only enhances our understanding of emotions but also opens new avenues for applications in customer service, healthcare, and beyond. Future research can focus on expanding emotion classification to include complex emotions such as nostalgia, frustration, and empathy, improving cross-linguistic emotion detection models, and incorporating multimodal sentiment analysis that integrates textual, audio, and visual data for a more comprehensive understanding of human emotions [11].

1.1 MOTIVATION

In today's digital era, where communication is largely text-based, understanding emotions from written language has become an essential task in numerous fields such as social media monitoring, customer sentiment analysis, mental health assessments, and AI-driven conversational systems. People express their emotions through text in various ways, including informal language, abbreviations, slang, and even sarcasm, making accurate emotion detection an increasingly complex challenge. Traditional sentiment analysis methods, such as lexicon-based approaches or basic machine learning techniques, often fail to capture the contextual depth, nuanced meanings, and subtle emotional expressions present in text.

Deep learning models, such as Convolutional Neural Networks (CNNs) and Bidirectional Gated Recurrent Units (BiGRUs), have demonstrated impressive capabilities in feature extraction and sequence modelling, allowing them to learn intricate patterns in textual data. CNNs are particularly effective at capturing local dependencies and hierarchical features, while BiGRUs excel at processing long-range dependencies in both forward and backward directions. However, despite their strengths, deep learning models can sometimes overfit on limited datasets, lack interpretability, and require significant computational resources. On the other hand, classical machine learning models like Support Vector Machines (SVMs) are well-known for their robust classification ability, strong generalization, and efficient training on smaller datasets, but they depend heavily on manual feature engineering, which may not always capture the full richness of textual information.

To overcome these challenges, this project introduces a hybrid model that integrates CNN, BiGRU, and SVM, leveraging the strengths of each approach. By combining deep learning's ability to extract meaningful representations from raw text with the generalization power of SVMs, the proposed model provides a more accurate, efficient, and scalable solution for emotion detection from text. This hybrid approach enables the system to understand complex emotions, handle ambiguous language, and improve classification performance, making it valuable for real-world applications requiring high precision and contextual awareness.

1.2 PROBLEM STATEMENT

Emotion detection in textual data is a highly complex and multi-dimensional challenge due to several inherent difficulties. Human language is naturally ambiguous, context-dependent, and emotionally rich, making it difficult for traditional models to extract precise emotional cues. One of the key challenges is disambiguating words and phrases that may carry different emotions depending on the context. For instance, the phrase "I can't believe this!" could express joy, surprise, sarcasm, or frustration, depending on its usage and tone. Many existing models struggle to distinguish between literal and figurative expressions, leading to misclassifications.

Another major hurdle is the scarcity of high-quality labeled datasets for emotion classification. Unlike general sentiment analysis, which often relies on binary or ternary classification (positive, negative, neutral), emotion detection requires fine-grained

classification into multiple emotions such as joy, fear, anger, sadness, disgust, shame, and guilt. Emotion datasets are often imbalanced, with some emotions appearing far more frequently than others, making it challenging for models to learn effectively. Deep learning models typically require large-scale training data to perform well, while machine learning approaches depend on manual feature selection, which may not capture the full complexity of emotional expressions.

Furthermore, the presence of sarcasm, irony, and implicit emotions complicates emotion detection. Sarcastic statements often have a mismatch between literal words and intended meaning, causing standard models to misinterpret them. For example, "Oh great, another Monday!" appears positive in a traditional sentiment analysis model but likely expresses dissatisfaction in reality. Existing deep learning models, while powerful, do not always generalize well to unseen data, leading to inconsistencies and errors in real-world applications.

To address these challenges, this project proposes a hybrid CNN-BiGRU + SVM model that effectively combines feature extraction, sequential context modeling, and robust classification. By utilizing CNNs to extract spatial relationships in text embeddings, BiGRUs to capture temporal dependencies, and SVMs to generalize better on unseen data, this model aims to enhance the accuracy, robustness, and interpretability of emotion classification systems.

1.3 OBJECTIVE

The objective of this project is to develop an advanced hybrid emotion detection model that combines Convolutional Neural Networks (CNNs), Bidirectional Gated Recurrent Units (BiGRUs), and Support Vector Machines (SVMs) to achieve high accuracy, robustness, and efficiency in classifying emotions from textual data. Unlike traditional sentiment analysis methods that focus on binary or ternary classification, this model aims to categorize text into seven distinct emotions: joy, fear, anger, sadness, disgust, shame, and guilt. By leveraging CNNs for high-level feature extraction, BiGRUs for context-aware sequence learning, and SVMs for generalized classification, the proposed approach enhances semantic understanding, contextual depth, and classification precision.

Additionally, the project seeks to address key challenges in emotion detection, including ambiguity in language, sarcasm, imbalanced datasets, and the need for fine-grained emotional categorization. Pre-trained word embeddings such as Word2Vec will be utilized to enrich the model's ability to capture semantic relationships between words, ensuring a deeper understanding of textual context. The architecture is designed to be scalable and adaptable for various real-world applications, including sentiment analysis, mental health monitoring, customer feedback evaluation, and AI-driven conversational agents. Furthermore, dark-themed visualizations with separate graphs for ML and DL models will be incorporated to enhance interpretability and provide clearer insights into model performance. By integrating the strengths of deep learning and machine learning, this project aims to set a new standard for emotion detection systems, making them more accurate, efficient, and applicable across diverse domains.

To further improve performance and adaptability, the model will undergo rigorous evaluation using multiple datasets, ensuring it can effectively generalize across different text sources, such as social media posts, customer reviews, and mental health discussions. Fine-tuning techniques and hyperparameter optimization will be applied to balance precision and recall, reducing biases in emotion classification. Additionally, the system will be designed to support multilingual datasets, expanding its applicability beyond English-based corpora.

2. LITERATURE REVIEW

Traditional ML models like SVM and Decision Trees struggle with contextual nuances in text-based emotion detection, while lexicon-based approaches rely on predefined word-emotion mappings, limiting their accuracy. Deep learning models such as CNNs, LSTMs, and GRUs improve performance but still face challenges in generalization and handling subtle emotional cues. To overcome these issues, a hybrid model combining CNN for feature extraction, BiGRU for sequential learning, and SVM for classification was developed, enhancing accuracy and robustness in emotion recognition.

S. Arun Kumar et al. [2] highlighted the complexity of emotion detection from text due to word ambiguity and multiple meanings. They emphasized the need for further research in text-based emotion recognition, particularly in human-computer interaction, as speech and facial recognition have been more extensively explored. Their study compared three approaches: NRCLex for mapping words to emotional categories, deep learning-based detection using CNNs and GRUs, and traditional NLP techniques, demonstrating the varying effectiveness of each method.

Mohamed et al. [3] analysed various deep learning models, including LSTM, BiLSTM, and GRU, to determine the best-performing model on the ISEAR dataset. Their findings showed that GRU outperformed LSTM and BiLSTM in accuracy, making it a strong candidate for emotion recognition in applications like customer service personalization and psychological disorder diagnosis.

Santosh Kumar Bharti et al. [4] examined different emotion recognition methods, including keyword-based and machine learning-based approaches. Their keyword-based approach using the ISEAR dataset achieved 65% accuracy, but its effectiveness was limited by the lack of a comprehensive emotion keyword list.

Seal et al. [4] proposed a hybrid model combining CNN, Bi-GRU, and SVM, achieving 80.11% accuracy. Their research highlighted the importance of integrating deep learning and machine learning techniques for enhanced emotion recognition.

Ab. Nasir et al. [5] developed a machine learning-based emotion recognition system using Decision Trees, Naïve Bayes, SVM, and k-Nearest Neighbours to detect six basic emotions. Their results indicated that preprocessing techniques like stemming,

stop-word removal, and tokenization significantly boosted model performance, with Multinomial Naïve Bayes achieving the highest accuracy (64.08%). Their model was successfully integrated into a graphical user interface (GUI) for real-world text-based affective computing applications.

Patel et al. [6] explored the BERT framework for deep learning-based emotion detection. They compared conventional and deep learning methods, highlighting challenges in recognizing subtle and complex emotions. Their fine-tuned BERT model improved performance and interpretability, with applications in social media monitoring and mental health support.

Balapuri Shiva Sundar et al. [7] argued that traditional machine learning models like Random Forest (RF) and Logistic Regression (LR) are less effective than deep learning-based Bi-LSTM and Bi-GRU models. Their research showed that Bi-GRU achieved 78.70% accuracy, effectively capturing sequential text patterns. They also suggested that more advanced deep neural networks could further improve accuracy.

Mahinda Mahmoud Samy Zidan et al. [8] emphasized the superiority of deep learning models (CNN, RNN, and LSTM) over traditional machine learning techniques in emotion recognition tasks. Their review concluded that deep learning significantly enhances emotion detection accuracy compared to traditional approaches.

Poonam Arya et al. [9] noted that text-based emotion detection is less developed than speech and facial expression recognition. They identified limitations in keyword-based methods due to word ambiguity and insufficient contextual understanding. Their proposed hybrid model dynamically combined different strategies, improving emotion detection accuracy in textual content.

Pansy Nandwani et al. [11] explored three emotion detection methods: lexical approaches (predefined emotion lexicons), AI-based techniques (Naïve Bayes and SVM), and deep learning methods (CNN and LSTM). Their study analyzed the strengths and weaknesses of each approach, emphasizing the need for robust models that can effectively extract emotions from nuanced text.

Goru Swathi et al. [12] focused on emotion recognition in digital communication, emphasizing the importance of detecting emotions like sadness, anger, and joy from textual data. They applied supervised ML algorithms, including SVM,

Decision Trees, and Logistic Regression, on the ISEAR dataset to overcome challenges like poor vocabulary and weak semantic extraction, thereby improving text-based emotion classification.

Machine Learning (ML) refers to algorithms that learn patterns from data to make predictions or decisions without being explicitly programmed. ML is versatile and effective for structured data analysis, with algorithms like Support Vector Machines (SVM) and Decision Trees being relatively simple to implement. These models offer interpretability by identifying significant features in the data. However, ML has notable limitations[2]. It requires manual feature extraction, which is time-intensive and domain-specific. Additionally, ML struggles with high-dimensional and unstructured data like text or images, and it cannot effectively capture sequential dependencies or nuanced patterns, limiting its applicability in complex scenarios.

Deep Learning (DL) uses neural networks with multiple layers to automatically learn complex patterns, making it especially powerful for unstructured data such as text, images, and audio. DL models like CNNs, LSTMs, and Transformers can identify and image recognition [1]. Unlike ML, DL eliminates the need for manual feature engineering. However, DL comes with challenges, including high computational demands and a reliance on large, labelled hierarchical and sequential patterns, enabling superior performance in tasks like emotion detection datasets for effective training. Its black-box nature makes it less interpretable, which can be a drawback in applications requiring transparency.

This project introduces a hybrid emotion detection model combining CNN, BiGRU, and SVM for enhanced text-based emotion classification. Preprocessing techniques such as tokenization and stop-word removal refine textual data, while Word2Vec embeddings capture semantic relationships. CNNs extract local features, BiGRUs process sequential dependencies, and a self-attention mechanism highlights key emotional cues. SVM classifies the extracted features, ensuring robust generalization. The model identifies seven emotions—joy, fear, anger, sadness, disgust, shame, and guilt—offering high accuracy and scalability. With dark-themed visualizations and separate ML/DL performance graphs, the system is designed for mental health analysis, sentiment detection, and AI-driven chatbots, overcoming challenges like sarcasm and word ambiguity.

3.SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Various emotion detection systems have been developed, each with strengths and limitations. Lexicon-based methods rely on predefined word-emotion mappings but struggle with context. ML models like SVM and Decision Trees offer interpretability but require manual feature extraction. DL approaches, such as CNNs and GRUs, capture complex patterns but demand large datasets. Hybrid models combine these techniques for improved accuracy, while GUI-based systems enhance accessibility. The following section explores these methods in detail.

NRCLex is a lexicon-based system widely used for emotion detection. It matches words in text with predefined emotion categories, offering simplicity but lacking context handling. On the other hand, CNN-GRU-based systems combine feature extraction with sequential learning, improving accuracy and addressing ambiguity in word meanings. However, these systems struggle with subtle emotions like guilt or shame.

GRU-based models trained on the ISEAR dataset have shown superior performance compared to LSTM and BiLSTM for emotion detection. They effectively capture sequential patterns and require fewer resources. While GRU achieves high accuracy for common emotions, it is limited in scalability for larger datasets and real-time applications. Similarly, keyword-based methods extract emotions by matching text against predefined emotion keyword lists. These systems are simple and interpretable but lack flexibility, as limited keyword lists reduce accuracy. For instance, a study on the ISEAR dataset achieved only 65% accuracy using this approach, highlighting the need for more advanced models.

Hybrid models combining CNN, Bi-GRU, and SVM have been developed to integrate the strengths of machine learning and deep learning. These systems achieve over 80% accuracy by extracting local features using CNN, capturing context with Bi-GRU, and ensuring robust classification through SVM.

However, their computational intensity and preprocessing requirements pose challenges for real-time deployment. Similarly, systems integrating ML models like Decision Trees, Naïve Bayes, and SVM into GUI platforms aim to make emotion detection accessible. These systems achieve moderate accuracy (64%) and benefit from preprocessing enhancements. However, traditional ML techniques are limited in

capturing complex patterns, reducing their effectiveness compared to deep learning models as shown in Fig 3.1.

Fine-tuned BERT models address challenges in detecting complex emotions by leveraging contextual embeddings. They are particularly effective for applications like social media analysis and mental health monitoring. Despite their high accuracy, fine-tuning requires significant computational resources, making them less suitable for real-time scenarios. Additionally, Bi-GRU models capture bidirectional context in text, achieving 78.7% accuracy in emotion detection tasks. They are well-suited for applications requiring sequential understanding, such as customer service. However, these models require large datasets for training, limiting their utility in resource-constrained environments.
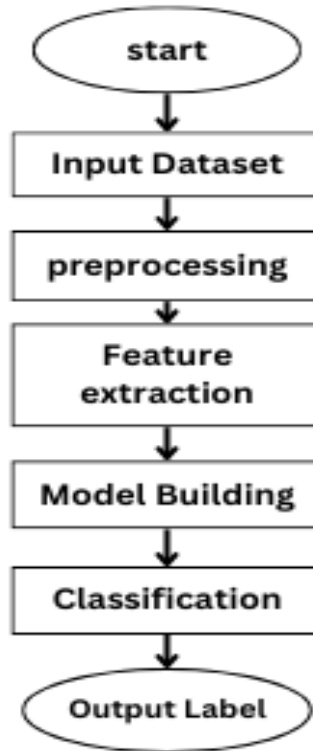

Fig 3.1: Architecture of Existing Model

3.1.1 DISADVANTAGES OF THE EXISTING SYSTEM FOR EMOTION DETECTION FROM TEXT

Despite significant progress, existing emotion detection systems face several critical challenges that limit their efficiency, scalability, and real-world applicability.

- **Lexicon-Based Systems (NRCLex):** These methods rely on predefined word-emotion mappings, making them ineffective in capturing contextual meaning. They struggle with polysemous words and fail to recognize emotions like guilt, shame.

- **CNN-GRU-Based Models:** While these models enhance accuracy by combining feature extraction and sequential learning, they require extensive training on large datasets. They often fail to detect subtle emotional variations, particularly in ambiguous or context-dependent text.

- **GRU on ISEAR Dataset:** GRU-based models have shown superior performance over LSTM and BiLSTM for emotion detection. However, they struggle with scalability when applied to large datasets or real-time applications.

- **Keyword-Based Methods:** These methods offer simplicity and interpretability but lack flexibility due to their dependence on predefined keyword lists. Limited vocabulary coverage leads to poor accuracy.They also fail to account for sentiment intensity, often misclassifying weak or strong emotions.

- **Hybrid Models (CNN, Bi-GRU, SVM):** By integrating ML and DL techniques, these models achieve high accuracy (over 80%). However, their computational demands make them unsuitable for low-resource environments. The need for extensive preprocessing, including data cleaning, tokenization.

- **Fine-Tuned BERT Models:** Transformer-based models like BERT significantly improve contextual emotion detection. However, they require vast amounts of labelled training data. Fine-tuning for domain-specific applications is resource-intensive posing challenges in sensitive fields like healthcare and legal analysis.

- **Bi-GRU Models:** These models effectively capture bidirectional dependencies in text, achieving nearly 79% accuracy. However, their effectiveness depends on large, well-annotated datasets, which may not always be available.

3.2 PROPOSED MODEL

There are several critical steps to follow when proposing a system for text-based emotion recognition, including data collection, preprocessing, feature extraction, and application of ML and DL models as shown in Fig3.2. The first step is data importation after which preprocessing begins by cleaning and preparing the data for analysis. These

processes may involve tokenization, stop words removal as well as stemming or lemmatization as key preprocessing steps so that the text can be in a given format that suits feature extraction.

In textual analysis across languages for instance one can use Lexicon-based Sentiment Analysis or machine learning approaches such as supervised classifying techniques while previously extracted features aid in converting it to numerical format suitable for processing by ML/DL models. Words can be semantically analysed using TF-IDF or vector representations.

With machine learning technique, pre-processed and extracted features data are given to various ML techniques including Random Forest, SVM and Naive Bayes algorithms. The performance measures including accuracy level might initiate its evaluation during which high precision becomes an advantage over low precision on evaluating performance metrics like precision recall and F1 score that are used on classifier's performance. The one with the highest good scores will be the optimal model among all ML models.
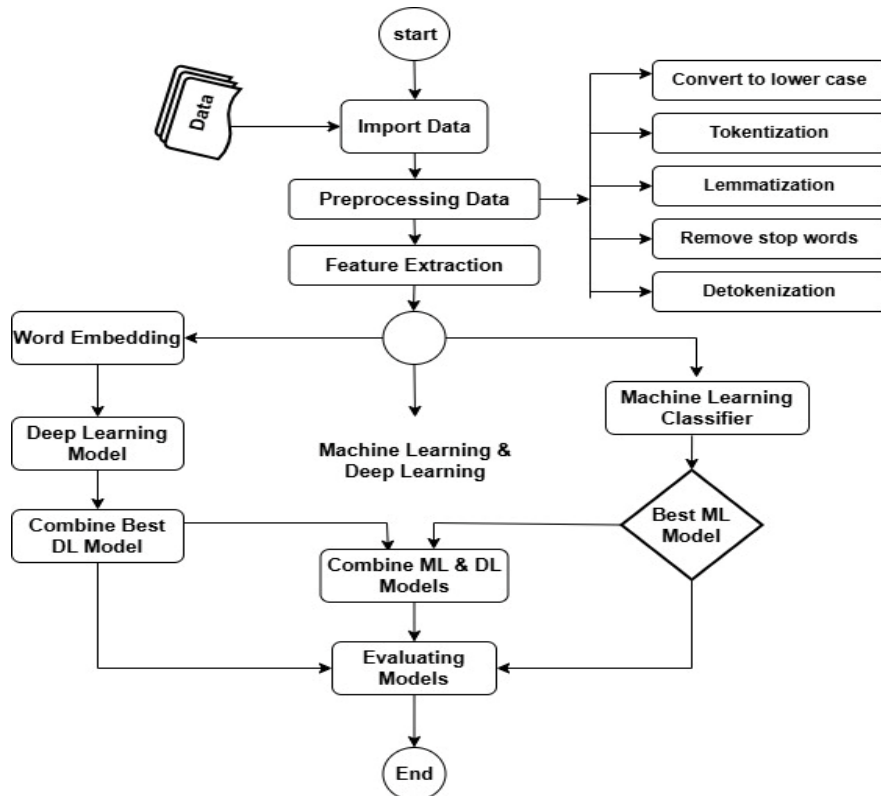

Fig 3.2: Pipelined Model of Proposed Scheme.

In the DL approach, there are these embeddings e.g. Word2Vec or GloVe which identify words through vector representations so as to maintain their semanticity. These embeddings are then fed into models like CNN, GRU or BiGRU in DL for the sake of capturing intricate patterns and contextual interdependencies in data. The best DL models perform according to their accuracy and F1 scores.

The hybrid DL model combines them all together by taking advantage of their strengths. We then combine this with the best performing instance of ML making it a strong system that employs both ML and DL methods simultaneously. The last hybrid model is assessed for its adequacy in making correct judgments about human emotional states based on written texts.

At last, this system picks out the one which has gotten topmost performance on different evaluation metrics as an emotional recognition solution that is trustworthy and practical. Therefore, dealing with emotions well entails integrating different models so as to enhance precision and avoid missing out important aspects.

Advantages Over Existing Systems

The proposed system improves upon existing emotion detection methods in several key ways:

1. **Better Context Handling:** Uses ML and DL to reduce misclassification from word ambiguity.

2. **Improved Feature Extraction:** Retains semantic meaning with Word2Vec, GloVe embeddings.

3. **Hybrid Model Strength:** Combines CNN, Bi-GRU, and SVM for higher accuracy and efficiency.

4. **Scalability:** Optimized for large datasets, addressing GRU model limitations.

5. **Higher Precision:** Outperforms traditional ML models in accuracy and robustness.

6. **Real-Time Suitability:** More efficient than transformer-based models like BERT.

3.3 FEASIBILITY STUDY

The integration of Convolutional Neural Networks (CNNs), Bidirectional Gated Recurrent Units (Bi-GRU), and Support Vector Machines (SVMs) offers a promising

hybrid approach for emotion detection from text. Below is a feasibility analysis considering technical, operational, and economic aspects.

1. Technical Feasibility

- **Automated Feature Extraction:**

  CNNs and Bi-GRUs work together to automatically extract deep, contextual features from text, eliminating the need for manual feature engineering. CNNs identify local patterns, while Bi-GRUs capture sequential dependencies, improving the model's ability to understand emotions.

- **Enhanced Classification with SVM:**

  While deep learning models extract features, SVM acts as a robust classifier, effectively handling high-dimensional text data. This hybrid approach ensures better generalization, especially in scenarios with limited labelled data.

- **Improved Accuracy and Contextual Understanding:**

  The combination of CNN-BiGRU with SVM reduces misclassification by leveraging both local and sequential dependencies in textual data. It effectively captures subtle emotional expressions and variations in sentence structure.

- **Scalability:**

  The system is scalable across different text datasets, including social media comments, chat conversations, and customer feedback. It can be adapted to multiple languages and domains with appropriate preprocessing techniques.

- **Integration with Pretrained Embeddings:**

  The model benefits from transfer learning using pre-trained word embeddings like Word2Vec or GloVe, which improve semantic understanding while reducing training time and computational costs.

2. Operational Feasibility

- **Ease of Deployment:**

  The hybrid model can be deployed in real-world applications such as sentiment analysis tools, mental health monitoring, and customer feedback systems. It can be integrated into web-based and mobile applications.

- **Interpretability:**

  Unlike purely deep learning-based models, the inclusion of SVM allows for better interpretability, as SVM provides clear decision boundaries. This makes the system more transparent for users in applications where explainability is crucial.

- **Data Handling:**

  The system requires labelled text data for training but is flexible in handling varying sentence structures, informal language, and domain-specific terms with the help of pre-trained embeddings and contextual feature extraction.

- **Maintenance and Upgradation:**

  The model can be updated with new data over time, allowing continuous improvement. Retraining the SVM component on new features is computationally efficient and helps the system adapt to evolving language trends.

3. Economic Feasibility

- **Cost-Effective Training:**

  Using pre-trained embeddings and transfer learning reduces the need for extensive model training, lowering computational costs while maintaining high accuracy.

- **Resource Optimization:**

  By efficiently combining CNN, Bi-GRU, and SVM, the model optimally utilizes computational resources, making it feasible to run on mid-range hardware or cloud-based platforms.

- **Reduced Operational Costs:**

  The automation of emotion detection reduces manual text analysis time, leading to faster and more accurate insights. Businesses can leverage this for real-time sentiment analysis, improving decision-making processes.

- **Long-Term Investment:**

  While initial development and integration costs may be high, the long-term benefits of improved accuracy, adaptability, and automation justify the investment. The system's ability to scale and update ensures sustainable performance over time.

3.4 USING THE COCOMO MODEL

The COCOMO Model helps estimate the effort, time, and cost required for software development. It categorizes projects into Organic, Semi-Detached, and Embedded based on complexity. Given the integration of deep learning (CNN-BiGRU) and machine learning (SVM), the Emotion Detection from Text project falls under the Semi-Detached category.

Using COCOMO, effort estimation ensures efficient resource allocation for data preprocessing, model training, hyperparameter tuning, testing, and deployment. It predicts development time by structuring these phases to prevent delays. Cost estimation considers team salaries, computational resources, and software tools, ensuring a realistic budget. By applying COCOMO, the project benefits from optimized resource management, accurate time planning, and cost-effective development, leading to a scalable and high-performing emotion detection system.

4. SYSTEM REQUIREMENTS

4.1 SOFTWARE REQUIREMENTS

1. Operating System   : Windows 11, 64-bit Operating System
2. Hardware Accelator   : CPU
3. Coding Language   : Python
4. Python distribution   : Google Colab, Flask
5. Browser   : Any Latest Browser like Chrome

4.2 REQUIREMENT ANALYSIS

The Text-Based Emotion Detection System aims to develop an efficient hybrid machine learning and deep learning model for accurately classifying emotions from textual data. The system integrates Convolutional Neural Networks (CNNs) with Bidirectional Gated Recurrent Units (BiGRUs) for feature extraction, along with Support Vector Machines (SVM) for final classification to enhance accuracy. Users can input text manually or upload a text file through a web interface, where the system validates the input, ensuring it contains meaningful textual data.

Preprocessing steps include tokenization, stopword removal, stemming/lemmatization, and vectorization using Word2Vec or TF-IDF. The processed text is then classified into emotions such as joy, fear, anger, sadness, disgust, shame, and guilt, with results displayed alongside confidence scores and visual representations like bar charts. Additionally, the system includes error handling to provide clear feedback for invalid inputs.

The system prioritizes speed, scalability, and security, ensuring efficient processing of large datasets. Built with Python, TensorFlow/Keras, Scikit-learn, NLTK, and Gensim, it requires a multi-core CPU, 8GB RAM, and GPU support for optimal performance. Trained on well-labelled emotion datasets, it can be deployed locally or on the cloud, making it a real-time, high-accuracy emotion analysis tool.

4.3 HARDWARE REQUIREMENTS

1. System Type   : 64-bit operating system, x64-based processor
2. Cache memory: 4MB(Megabyte)
3. RAM   : 16GB (gigabyte)

4. Hard Disk   : 8GB
5. GPU   : Intel® Iris® Xe Graphics

4.4 SOFTWARE

The emotion detection system is developed using a comprehensive set of software tools and frameworks, ensuring high accuracy, efficiency, and scalability. The system is compatible with Windows 11 (64-bit), Linux (Ubuntu 20.04+), and macOS, making it adaptable across different computing environments. It leverages both CPU and GPU acceleration, with GPUs being utilized for deep learning model training to enhance performance and reduce processing time.

The primary programming language used for development is Python, owing to its simplicity, flexibility, and extensive ecosystem of libraries for natural language processing (NLP) and deep learning. Model training and experimentation are conducted using Google Colab Pro, which provides access to high-performance cloud-based GPUs, or locally within Jupyter Notebook, allowing for an interactive and iterative development process.

For backend development, the Flask framework is employed, facilitating efficient API management, request handling, and seamless communication between the deep learning models and the web interface. Flask ensures smooth deployment and integration with frontend components. The web interface is developed using HTML5, CSS3, and Bootstrap, ensuring a responsive and user-friendly design that works efficiently across different screen sizes and modern web browsers, including Google Chrome, Mozilla Firefox, and Microsoft Edge.

The system integrates TensorFlow/Keras for building deep learning models, including CNN and BiGRU, which capture contextual dependencies in textual data. Additionally, scikit-learn is employed for machine learning-based classification using Support Vector Machines (SVM). Preprocessing of text data is handled by powerful NLP libraries such as NLTK and SpaCy, enabling operations like tokenization, stopword removal, stemming, and lemmatization. Feature extraction techniques include TF-IDF and Word2Vec embeddings, ensuring that text data is converted into meaningful numerical representations suitable for machine learning and deep learning models.

To enhance computational efficiency and optimize memory usage, NumPy and Pandas are used for handling large datasets and performing efficient numerical

19

computations. Visualization tasks, including performance evaluation, confusion matrices, and accuracy trends, are carried out using Matplotlib and Seaborn, enabling clear and insightful analysis of model performance.

4.5 SOFTWARE DESCRIPTION

The text-based emotion detection system requires a modern and stable operating system, with Windows 11 (64-bit), Ubuntu 20.04+, or macOS being the recommended choices. These operating systems ensure compatibility with the latest development tools, security updates, and optimized performance. The system primarily utilizes CPU for processing, but for large-scale training and deep learning-based computations, Google Colab Pro or local GPU acceleration can be leveraged to enhance processing speed and efficiency.

The project is developed using Python, a widely used programming language known for its rich ecosystem of natural language processing (NLP) and deep learning libraries. Model training and experimentation are performed in Google Colab Pro or Jupyter Notebook, providing an interactive environment for efficient development. For backend implementation, the Flask framework is used to deploy machine learning models as web services, ensuring seamless communication between the model and the user interface.

A web browser such as Google Chrome, Mozilla Firefox, or Microsoft Edge is required for accessing and interacting with the Flask-based web application. These modern browsers ensure fast load times, improved security, and compatibility with the system's HTML5, CSS3, and Bootstrap-based frontend. To manage dependencies efficiently, pip (Python Package Installer) is used for installing key libraries, including TensorFlow, Keras, NLTK, SpaCy, NumPy, and scikit-learn.

The combination of these software tools ensures that the text-based emotion detection system is scalable, efficient, and adaptable for real-world applications. The modular design allows for future upgrades, improving contextual understanding and classification accuracy.

5.SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE

This project focuses on enhancing the detection and classification of emotions from text using a hybrid Deep Learning and Machine Learning approach. By integrating Convolutional Neural Networks (CNNs) for effective feature extraction with Support Vector Machines (SVMs) for precise classification, the model aims to provide a reliable and automated emotion analysis tool as shown in Fig 5.1. The ultimate goal is to improve sentiment understanding, assisting businesses, researchers, and mental health professionals in analyzing human emotions more effectively.

The model processes textual data sourced from publicly available datasets containing labelled emotional expressions across various categories such as joy, fear, anger, sadness, disgust, shame, and guilt. Advanced preprocessing techniques, including tokenization, stop-word removal, lemmatization, and word embeddings (such as Word2Vec or GloVe), are applied to refine text input, ensuring optimal feature representation for classification.
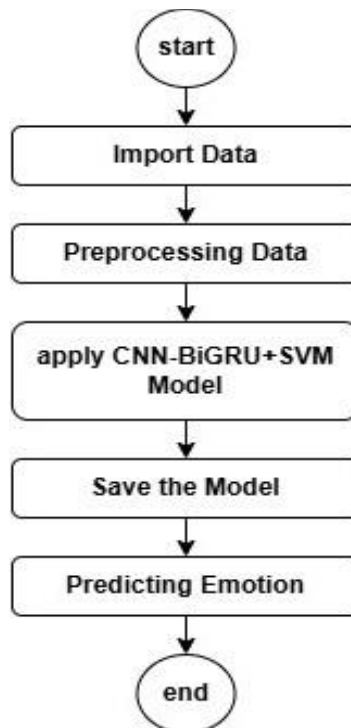

Fig 5.1: Design Overview

To enhance text representation, TF-IDF and word embeddings are used to extract meaningful contextual features. These features are then passed through the CNN-SVM hybrid model, where the CNN component extracts deep linguistic features,

and the SVM classifier ensures precise decision boundaries, leading to improved emotion classification performance.

The effectiveness of the proposed model is evaluated using key performance metrics, including accuracy, precision, recall, and F1-score. The model aims for high classification accuracy, surpassing traditional approaches such as standalone CNN, ANN, and RFC models.

The applications of this project extend to social media sentiment analysis, mental health monitoring, and customer feedback assessment. By providing an automated, efficient, and scalable emotion detection system, the model helps reduce manual analysis time and improve real-time decision-making. Looking ahead, the project aims to expand by integrating transformer-based models like BERT or GPT for enhanced contextual understanding. Additionally, deploying the CNN-SVM hybrid model as an API or embedding it into business intelligence tools will enable practical applications in customer service, psychology research, and digital marketing analytics.

5.2 DATASET

The ISEAR dataset (International Survey on Emotion Antecedents and Reactions) is a comprehensive collection of emotional expressions gathered through global surveys, containing 7,516 textual entries categorized into seven fundamental emotions: joy, sadness, anger, fear, shame, disgust, and guilt [6]. This dataset plays a crucial role in text-based emotion detection, providing a structured foundation for training and evaluating machine learning models. A table1 below shows the count of all the emotion labels present in this particular dataset implying their frequency.

Having such distributions allows us to consider whether the dataset is poised between two extremes or is weighted towards certain feelings more than others. A balanced dataset prevents a model from being biased towards predicting more frequently occurring categories thus ensuring that all classes are accurately represented and generalizations are made across them[7].

A well-balanced dataset is essential for building an unbiased and efficient emotion detection model. The table1 below represents the distribution of emotion labels, ensuring that no category dominates the training process.

Table 5.1. Label Count in Dataset

EMOTION	ISEAR
joy	1092
sadness	1082
anger	1079
fear	1076
shame	1071
disgust	1066
guilt	1050
Total	7516

The emotion categories used in this research contains different types of emotions derived from textual sources [2]. Thus for effective model training it's necessary to understand the distribution of these categories in the data because they can cause an imbalance which will not allow for effective training.

In case there exists an imbalance within a dataset then techniques like data augmentation, majority class under sampling or minority class oversampling can be adopted so as to enhance model capability ensuring precise prediction of every emotion at hand.
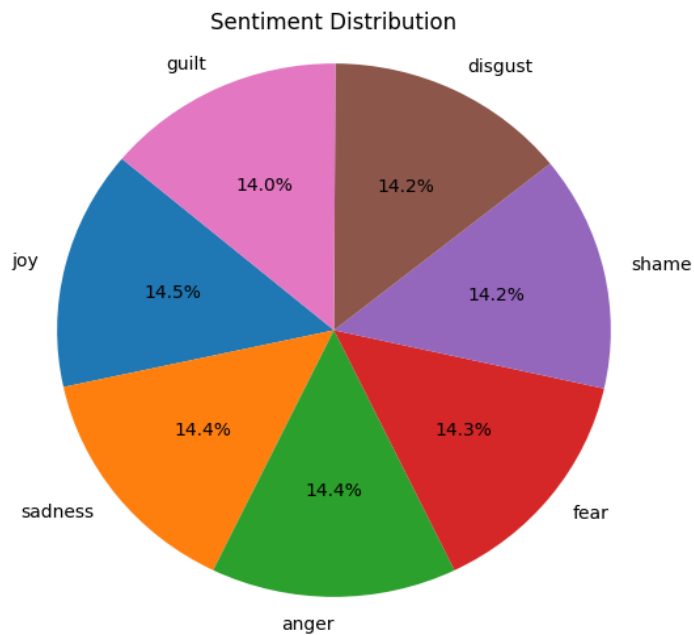

Fig 5.2: Graphical Representation of Dataset Categories

The Fig 5.2 illustrates this pleasing distribution that ensures equal representation across the emotions that are essential for constructing an efficient machine learning algorithm. To help avoid any potential bias and also promote an effective learning process, the datasets allow same number of records for each class. It can be stated with certainty that such balance is significant while performing preprocessing stages such as tokenization or stopword removal as they help the models accurately detect emotions during text analysis [4].

Applications:

- Used in training and testing Machine Learning and Deep Learning models for text-based emotion analysis.

- Supports tasks like emotion detection and classification in social media and customer feedback.

- Helps in identifying emotional states for early detection of mental health issues.

5.3 DATA PRE-PROCESSING

The preprocessing of text data for detecting the emotions included several important steps which enabled transforming raw material into a machine learning and deep learning friendly structured format. Important preprocessing techniques such as tokenization, removal of stop words and lemmatization were used for cleaning up the data so that it can be standardised to enhance performance of the emotion recognition system [1]. The natural language toolkit method has been used in Fig 5.3 to execute all the preprocessing actions like tokenization, stop word dropping and lemmatisation.
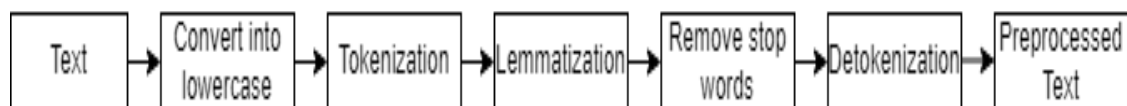

Fig 5.3: Flow of Preprocessing Techniques

Stop words are the frequently used words which are normally filtered off during text preprocessing because they carry little weight in terms of meaning and could potentially distort any analyses made. A set of English stopwords was used to eliminate these uninformative words from the text [2]. By doing so, it becomes easier to

concentrate only in those key terms that foster understanding of emotion and background context.

Lemmatization is a word normalization technique whereby terms are reduced into their basic or root forms. For instance: running and ran may be reduced to just run. In this way one can standardize a piece of writing by getting rid of various word forms thus reducing its dimensionality and complexity. Additionally, lemmatization improves model accuracy since it treats different forms of a term as one unit[3].

Tokenization is the process of dividing a piece of writing into smaller units like separate words or tokens. The significance of this stage is in enabling a simpler study and processing of text which involves breaking down sentences into manageable parts[4]. This operation can also deal with separate actions like stopword removal and lemmatization for each independent token.

The cleaned continuous string is finally constructed after tokenization and lemmatization stage that has been previously described and shown in Fig5.4. [1]. The procedure includes aggregating back all filtered and lemmatized lexical items. The cleaned text now assumes a form suitable for feature extraction as well as model training.

To put together records for system mastering, a completely unique mapping of sentiment labels is created and converted into numerical indexes [5]. This is an important conversion seeing that system mastering models require numerical enter for processing. The conversion of textual sentiments into numerical values makes the data usable by the version in a format that it is able to manipulate well.

sentiment	content	tokens	processed_text	sentiment_index
disgust	watching a violent movie	['watching', 'violent', 'movie']	watching violent movie	4
joy	when i won a tennis match	['tennis', 'match']	tennis match	0
sadness	failing an examination	['failing', 'examination']	failing examination	3
anger	i locked myself out	['locked']	locked	2
shame	i gave a wrong answer at school	['gave', 'wrong', 'answer', 'school']	gave wrong answer school	5
fear	disappointment over a friend	['disappointment', 'friend']	disappointment friend	1
guilt	i do not help out enough at home	['help', 'enough', 'home']	help enough home	6

Fig 5.4: Preprocessing Result of Dataset

These preprocessing techniques are important in making sure that the textual content information is smooth, standardized, and structured in this sort of manner that it is able to serve the reason of training and comparing emotion detection fashions [6].

5.3.1 FEATURE EXTRACTION

Feature extraction is a crucial step in text-based emotion analysis as it converts raw textual data into numerical representations that can be processed by machine learning (ML) and deep learning (DL) models. This involves transforming raw textual content into virtual representations which permit deep studying fashions to control and categorize sentiment facts properly [7]. The procedures encompass label encoding, TF-IDF vectorization and other associated steps that thought to be considered if one wants to effectively educate a version. There are several feature extraction techniques:

Label Encoding:

Label encoding is applied to convert categorical sentiment labels into numerical values. This process transforms labels like 'positive,' 'negative,' and 'neutral' into integers, facilitating their use in machine learning models[8]. For instance, sentiments might be encoded as 0, 1, and 2. This numerical representation is difficult because most ML algorithms.

TF-IDF (Term Frequency-Inverse Document Frequency):

TF-IDF changes text into quantitative features by evaluating the importance of words within a document relative to a collection. This technique reduces dimensionality while retaining key information, allowing models to focus on the most significant terms for sentiment analysis [9]. TF-IDF values are typically used as input features for traditional ML models.

TF-IDF vectorization is primarily associated with traditional ML techniques rather than deep learning (DL). In DL models like CNN often work directly with word embeddings (such as Word2Vec, GloVe) instead of TF-IDF [2]. However, TF-IDF can still be valuable in a hybrid approach that combines traditional ML and DL techniques, enhancing pattern recognition by leveraging the strengths of both methodologies.

Data Partitioning:

The dataset is subsequently partitioned into training and testing subsets through a technique known as train-test split. Typically, a part of the information is used for testing, the other part is used for training the model [10]. This division ensures that the

model is assessed on unseen data, offering a more precise evaluation of its effectiveness and helping to prevent over fitting.

Tokenization:

Tokenization converts the text data into sequences of tokens, which are smaller, manageable units like words or stopwords [11]. Tokenizing the text breaks it down into these units, preparing it for processing by the machine learning model. Each text instance is transformed into a sequence of tokens that can be analysed to detect patterns relevant to emotion detection.

Padding:

Padding sequences ensures that all tokenized inputs have the same length. By adding special tokens (such as zeros) to shorter sequences,[4] this step standardizes the input size. Consistent sequence length is essential for effective model training, as it allows the model to process data in uniform batches and maintain the required input shape [12].

5.4 MODEL BUILDING

Support Vector Machine (SVM)

It is a supervised learning algorithm mainly employed for classification tasks. The algorithm functions by identifying the optimal hyperplane that divides data points from distinct classes within a high dimensional space. SVM is especially useful when data cannot be separated linearly, as it utilizes kernel functions to map the data into a higher-dimensional space, enabling linear separation [1].

Support Vector Machine (SVM) is a widely used supervised learning algorithm for text classification tasks, including emotion detection from text. It is particularly effective in handling high-dimensional data, making it suitable for processing textual information represented as feature vectors. In emotion detection, SVM works by finding an optimal hyperplane that best separates different emotion classes within a feature space. It utilizes kernel functions such as linear, polynomial, and radial basis function (RBF) kernels to transform input data into a higher-dimensional space where emotions can be more easily distinguished [3]. The process begins with text preprocessing, where techniques like tokenization, stopword removal, and vectorization are applied to convert text into numerical feature representations. SVM then classifies the text into predefined emotion categories such as joy, fear, anger,

sadness, disgust, shame, and guilt by maximizing the margin between different emotion classes.

One of the key advantages of SVM in emotion detection is its ability to handle high-dimensional data efficiently while being robust to overfitting, ensuring better generalization even with limited training data [2]. Unlike deep learning models that require extensive datasets, SVM can perform well with smaller labelled datasets, making it a practical choice for many real-world applications.

Convolutional Neural Networks (CNN):

CNNs are deep learning models frequently used in image processing but have also been successfully applied to text data. CNNs operate by applying convolutional filters to extract local patterns, such as n-grams, from text, making them effective at identifying features that contribute to emotion classification. In the context of emotion detection, CNNs can spontaneously learn hierarchical features from text, capturing the local dependencies between words [4]. By using multiple convolutional layers, CNNs can model complex interactions within the text, which are essential for accurate sentiment analysis and emotion detection.

A CNN for emotion detection consists of key layers:

- **Input Layer** – Receives text as word embeddings.
- **Convolutional Layer** – Applies filters to extract n-gram features.
- **Pooling Layer (Max Pooling)** – Reduces dimensionality while keeping key features.
- **Flatten Layer** – Converts feature maps into a 1D vector.
- **Fully Connected Layer** – Processes extracted features for classification.
- **Output Layer** – Uses Soft max to predict emotion categories.

Bidirectional Gated Recurrent Unit (BiGRU):

BiGRU is an extension of the standard GRU model that processes input sequences. This bidirectional approach allows model to grasp context from both the before and after within a sequence, providing a understanding of the text[5]. In emotion detection, BiGRU boosts the model's ability to recognize emotions that may depend on the surrounding context of words, improving the accuracy of emotion classification tasks.

Bidirectional Gated Recurrent Unit (BiGRU) is a deep learning architecture designed to capture contextual dependencies in sequential data, making it highly effective for emotion detection from text[7]. Unlike standard GRU, BiGRU processes text in both forward and backward directions, ensuring a better understanding of word relationships and contextual meaning.

A BiGRU for emotion detection consists of key layers:

- **Embedding Layer** – Converts words into dense vectors, capturing semantic relationships.

- **Bidirectional GRU Layer** – Processes text in both forward and backward directions, improving context understanding.

- **Dropout Layer** – Prevents overfitting by randomly deactivating neurons during training.

- **Dense Hidden Layer** – Uses ReLU activation to extract key features for classification.

- **Output Layer** – Applies Softmax activation to predict emotions like joy, anger, and sadness.

Hybrid Model (CNN-BiGRU + SVM) building process:

Emotion detection from text is a crucial task in natural language processing (NLP) that enables various applications, including sentiment analysis, customer feedback analysis, and mental health monitoring. The CNN-BiGRU + SVM model is a hybrid approach that integrates deep learning with machine learning to enhance the accuracy of emotion classification [6]. This model effectively captures both local word-level features and long-range dependencies while leveraging the power of a Support Vector Machine (SVM) for final classification.

The model begins with an embedding layer, which converts input text into dense vector representations that preserve semantic meaning. Next, a Convolutional Neural Network (CNN) is applied to extract essential local features and spatial relationships between words. The Conv1D layer detects important n-gram patterns, while the MaxPooling layer reduces dimensionality and retains the most significant features[7]. These operations help CNN capture key textual patterns that contribute to emotional expression.

Following CNN feature extraction, a Bidirectional Gated Recurrent Unit (BiGRU) processes the sequential data in both forward and backward directions. Unlike

traditional RNNs, GRUs effectively handle long-range dependencies by mitigating the vanishing gradient problem [8]. The bidirectional nature allows the model to understand contextual meaning from past and future words, which is crucial for emotion recognition. The BiGRU layer outputs a deep contextual representation of the input text, enhancing its ability to detect nuanced emotional expressions.

After extracting both spatial and contextual features, the model flattens the feature maps and passes them to an SVM classifier for final prediction [9]. Unlike traditional dense layers, SVM offers a powerful alternative by defining an optimal hyperplane that separates different emotion classes. This hybrid combination ensures that the model benefits from deep feature learning while utilizing SVM's ability to generalize well and avoid overfitting.

The CNN-BiGRU + SVM model provides a robust and scalable approach to emotion detection. CNN efficiently identifies word-level patterns, BiGRU captures sequential dependencies, and SVM enhances classification by finding precise decision boundaries. This model is particularly beneficial in domains where accurate emotion analysis is critical, such as psychological assessments, social media sentiment tracking, and chatbot responses. By integrating deep learning with machine learning, this approach achieves high accuracy and improved generalization, making it a reliable choice for real-world emotion detection tasks [10].

Advantages of the Hybrid Model:

- Enhanced Feature Representation
- Improved Context Understanding
- Effective Handling of Sequential Data
- Higher Classification Accuracy
- Robustness to Noisy Text Data
- Reduced Overfitting
- Efficient Training and Inference
- Scalability for Large Datasets
- Suitability for Real-Time Applications

5.5 CLASSIFICATION

Classification using CNN–BiGRU+SVM proposed hybrid model:

Hybrid models combine the strengths of multiple machine learning or deep learning architectures to enhance classification performance [11]. In this study, a hybrid model integrating Convolutional Neural Networks (CNN), Bidirectional Gated Recurrent Units (BiGRU), and Support Vector Machines (SVM) was utilized to improve emotion detection from text. Each component of the hybrid model plays a crucial role in capturing different aspects of textual data, leading to improved classification accuracy.

CNN is responsible for feature extraction, effectively capturing spatial and local dependencies in word embeddings. BiGRU, a variant of Gated Recurrent Units (GRU), is used to model sequential dependencies in textual data, leveraging bidirectional processing for better context understanding. Finally, the extracted deep features are fed into an SVM classifier, which excels in high-dimensional decision boundaries, ensuring optimal classification[12].

The hybrid model was trained on a labelled dataset with multiple emotion categories, and its performance was evaluated against traditional models such as CNN, BiGRU, SVM, Random Forest, and Naïve Bayes[1]. The results showed that the CNN-BiGRU + SVM model achieved higher accuracy and better generalization compared to individual models, demonstrating the effectiveness of combining deep learning with classical machine learning techniques.

This approach highlights the importance of hybrid architectures in text classification tasks, [4] where combining feature extraction, sequence modelling, and robust classification leads to superior predictive performance.

Precision: Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. It measures the accuracy of positive predictions by determining how many of the classified instances actually belong to the correct category.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Recall: Recall is the ratio of correctly predicted positive observations to all actual positive observations. It measures how well the model identifies all relevant instances in a dataset.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

F1-Score: The F1-score is the harmonic mean of precision and recall, providing a balanced measure of a model's performance. It is especially useful when there is an uneven class distribution or when both false positives and false negatives need to be minimized.

$$\text{F1score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Other models compared with the proposed CNN-BiGRU+SVM model:

Random Forest

Random Forest is an ensemble learning method that combines multiple decision trees to enhance classification accuracy. In emotion detection from text, Random Forest analyses textual features, such as word frequency, TF-IDF scores, and word embeddings, to classify emotions[5]. It is robust against overfitting due to its averaging mechanism and performs well with structured textual data. However, it may struggle with capturing deep semantic meanings compared to neural networks.

Naive Bayes

Naïve Bayes is a probabilistic classifier based on Bayes' Theorem, assuming independence between features. It is commonly used for text classification tasks like sentiment and emotion analysis[6]. In emotion detection, Naïve Bayes processes word occurrences or TF-IDF representations to predict emotions. It is computationally efficient, easy to implement, and performs well with smaller datasets, but its assumption of feature independence may limit accuracy when dealing with complex sentence structures.

Gated Recurrent Unit (GRU)

GRU is a recurrent neural network (RNN) variant designed to handle sequential data efficiently. It utilizes gating mechanisms (reset and update gates) to control information flow, allowing it to capture long-term dependencies in text[7]. In emotion detection, GRU processes word embeddings sequentially, learning contextual relationships to

classify emotions effectively. Compared to traditional RNNs, GRUs are computationally lighter and faster while maintaining performance close to LSTMs. They are particularly useful for emotion detection from longer texts where context plays a crucial role.

5.6 UML DIAGRAMS

The proposed system integrates CNN, BiGRU, and SVM to enhance text-based emotion detection. The workflow includes preprocessing (tokenization, stop-word removal, lemmatization), feature extraction using CNN and BiGRU, and classification via SVM for precise emotion identification. Performance evaluation employs accuracy, precision, recall, and F1-score, with a confusion matrix analyzing misclassification patterns.

The Use Case Diagram illustrates key system interactions, including user inputs, database management, preprocessing, feature extraction, classification, and result evaluation as shown in Fig5.5. It highlights how different components interact to process text and generate emotion predictions.
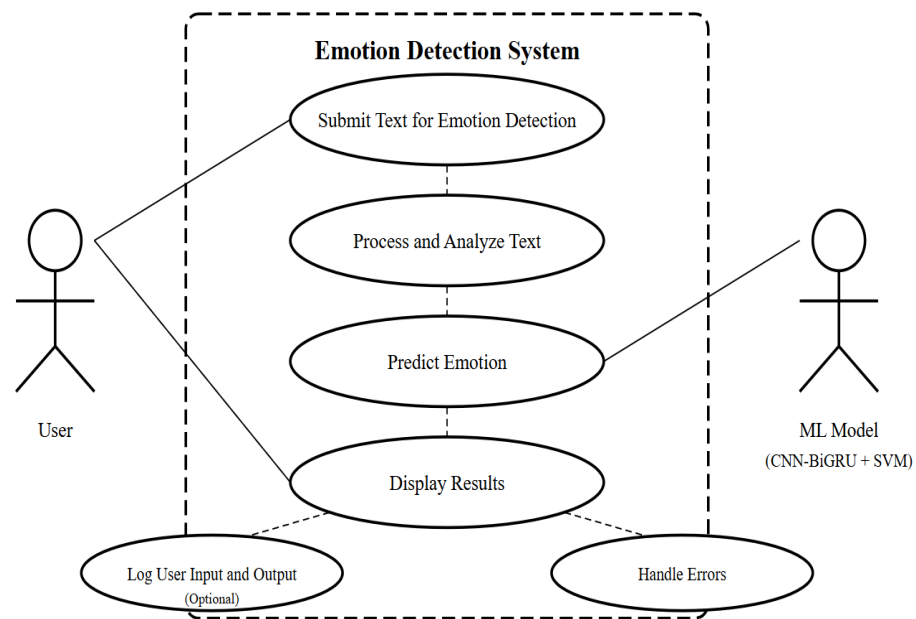

Fig 5.5: Use Case Diagram

6. IMPLEMENTATION

6.1 MODEL IMPLEMENTATION

CNN-BiGRU+SVM Model

 # Build CNN the model

cnn_model = Sequential()

cnn_model.add(Embedding(input_dim=5000, output_dim=100, input_length=100))

cnn_model.add(Conv1D(filters=128, kernel_size=5, activation='relu'))

cnn_model.add(MaxPooling1D(pool_size=4))

cnn_model.add(Flatten())

cnn_model.add(Dense(64, activation='relu'))

cnn_model.add(Dense(len(sentiment_to_index), activation='softmax'))

Compile the model

cnn_model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

Define EarlyStopping to prevent overfitting

early_stopping = EarlyStopping(monitor='val_loss', patience=3)

Train the model

history = cnn_model.fit(X_train, y_train, epochs=10, batch_size=32, validation_split=0.2, callbacks=[early_stopping])

Evaluate the model

y_pred = cnn_model.predict(X_test)

y_pred_classes = np.argmax(y_pred, axis=1)

Calculate accuracy

accuracy = accuracy_score(y_test, y_pred_classes)

print(f' CNN Accuracy: {accuracy:.4f}')

Define the BiGRU model

bigru_model = Sequential()

bigru_model.add(Embedding(input_dim=5000, output_dim=100, input_length=100)) # Adjust input_dim and output_dim as needed

bigru_model.add(Bidirectional(GRU(128, return_sequences=True))) # Bidirectional GRU layer with 128 units

bigru_model.add(Bidirectional(GRU(64))) # Another Bidirectional GRU layer with 64 units

```python
bigru_model.add(Dropout(0.5))  # Dropout layer to prevent overfitting
bigru_model.add(Dense(64, activation='relu'))
bigru_model.add(Dense(len(sentiment_to_index), activation='softmax'))  # Output
layer with units equal to the number of sentiment classes
# Compile the model
bigru_model.compile(optimizer='adam',        loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
# Define EarlyStopping to prevent overfitting
early_stopping = EarlyStopping(monitor='val_loss', patience=3)
# Train the model
history    =    bigru_model.fit(X_train,    y_train,    epochs=10,    batch_size=32,
validation_split=0.2, callbacks=[early_stopping])
# Evaluate the model
y_pred = bigru_model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)
# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred_classes)
print(f'BiGRU Accuracy: {accuracy:.4f}')


# Build the CNN-BiGRU model
embedding_dim = 100
cnn_bigru_model = Sequential([
    Embedding(input_dim=5000, output_dim=embedding_dim, input_length=100),
    Conv1D(128, 5, activation='relu'),
    MaxPooling1D(pool_size=4),
    Bidirectional(GRU(128, return_sequences=True)),
    Bidirectional(GRU(128)),
    Dropout(0.5),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(len(sentiment_to_index), activation='softmax')
])
cnn_bigru_model.compile(loss='sparse_categorical_crossentropy',  optimizer='adam',
metrics=['accuracy'])
```

```python
# Train the CNN-BiGRU model
history = cnn_bigru_model.fit(X_train, y_train, epochs=10, batch_size=32,
validation_split=0.2)
# Evaluate the CNN-BiGRU model
cnn_bigr_result = cnn_bigru_model.evaluate(X_test, y_test)
print(f' CNN-BiGRU Test Accuracy: {cnn_bigr_result[1]}')


#Build SVM model
svm_clf = SVC(kernel='linear', probability=True)
svm_clf.fit(X_train_tfidf, y_train_tfidf)
# Evaluate the SVM model
svm_predictions = svm_clf.predict(X_test_tfidf)
svm_accuracy = accuracy_score(y_test_tfidf, svm_predictions)
print(f'SVM Model Accuracy: {svm_accuracy:.4f}')


# Hybrid Model: Combine CNN-BiGRU and SVM
cnn_bigr_features = cnn_bigru_model.predict(X_test)
combined_features = np.hstack((cnn_bigr_features, X_test_tfidf))
# Train SVM on combined features
svm_clf_combined = SVC(kernel='linear', probability=True)
svm_clf_combined.fit(combined_features, y_test)
# Evaluate the hybrid model
combined_predictions = svm_clf_combined.predict(combined_features)
combined_accuracy = accuracy_score(y_test, combined_predictions)
print(f'Hybrid Model Test Accuracy: {combined_accuracy}')


# training and validation loss values
train_loss = history.history['loss']
val_loss = history.history['val_loss']
# Plotting the losses
plt.figure(figsize=(10, 6))
plt.plot(train_loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.title('Training and Validation Loss for CNN-BiGRU+SVM')
```

```python
plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.legend()

plt.grid(True)

plt.show()
```

 # Function to evaluate performance metrics for model

```python
def evaluate_model(y_true, y_pred):

    accuracy = accuracy_score(y_true, y_pred)

    precision = precision_score(y_true, y_pred, average='weighted')

    recall = recall_score(y_true, y_pred, average='weighted')

    f1 = f1_score(y_true, y_pred, average='weighted')

    return accuracy, precision, recall, f1

performance_metrics = []

combined_features = np.hstack((cnn_bigr_result, X_test_tfidf))

combined_predictions = svm_clf_combined.predict(combined_features)

hybrid_metrics = evaluate_model(y_test, combined_predictions)

performance_metrics.append(['Hybrid (CNN-BiGRU + SVM)'] +
list(hybrid_metrics))
# Create a DataFrame for better visualization

metrics_df = pd.DataFrame(performance_metrics, columns=['Model', 'Accuracy',
'Precision', 'Recall', 'F1-Score'])

# Print the table

print(metrics_df)
```

#Confusion Matrix

```python
# Generate predictions from the hybrid model

cnn_bigr_features = cnn_bigru_model.predict(X_test)

combined_features = np.hstack((cnn_bigr_features, X_test_tfidf))

# Predict using the SVM classifier on combined features

hybrid_predictions = svm_clf_combined.predict(combined_features)

# Define the sentiment labels mapping

index_to_sentiment = {
```

37

```python
    0: 'joy',

    1: 'fear',

    2: 'anger',

    3: 'sadness',

    4: 'disgust',

    5: 'shame',

    6: 'guilt'

}
# Convert numeric labels to sentiment labels

sentiment_labels = [index_to_sentiment[i] for i in np.unique(y_test)]

# Compute confusion matrix

cm = confusion_matrix(y_test, hybrid_predictions)

plt.figure(figsize=(12, 10))

disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=sentiment_labels)

disp.plot(cmap='Blues', values_format='d', ax=plt.gca())

plt.title('Confusion Matrix for Hybrid Model (CNN-BiGRU + SVM)',
fontsize=16)

plt.xlabel('Predicted Label', fontsize=14)

plt.ylabel('True Label', fontsize=14)

plt.xticks(fontsize=12)

plt.yticks(fontsize=12)

plt.grid(False)

plt.show()
```

6.2 CODING

PER-PROCESSING AND FEATURE EXTRACTION

#importing necessary libraries

```python
import numpy as np

import pandas as pd

import joblib
```

38

```python
import nltk

from nltk.corpus import stopwords

from nltk.tokenize import word_tokenize

from nltk.stem import WordNetLemmatizer

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import LabelEncoder

from sklearn.svm import SVC

from sklearn.metrics import accuracy_score

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Embedding, Conv1D, MaxPooling1D,
Flatten, Dense,

Dropout, Bidirectional, GRU

from tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.preprocessing.sequence import pad_sequences

from keras.callbacks import EarlyStopping

import tensorflow as tf

import joblib

import matplotlib.pyplot as plt

import gensim

from gensim.models import Word2Vec

from gensim.models import KeyedVectors

import matplotlib.pyplot as plt

from sklearn.metrics import classification_report, accuracy_score,
precision_score, recall_score, f1_score

import seaborn as sns

from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
```

 # Download necessary NLTK data

```python
nltk.download('punkt')

nltk.download('stopwords')

nltk.download('wordnet')
```

```python
#Preprocessing the text
# Initialize the lemmatizer
lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words('english'))
```

 # Load the data
```python
data = pd.read_csv('/content/drive/MyDrive/ISEAR_dataset_7.csv',
usecols=['sentiment', 'content'])
# Function to preprocess the input text
def preprocess_text(text):
    # Tokenization
    words = word_tokenize(text.lower())
    # Stop word removal and lemmatization
    filtered_words = [lemmatizer.lemmatize(word) for word in words if
word.isalnum() and word not in stop_words]
    # Reconstruct the processed text
    return ' '.join(filtered_words)
data['processed_text'] = data['content'].apply(preprocess_text)
# Create a dictionary mapping sentiments to numerical indexes
sentiment_to_index = {sentiment: idx for idx, sentiment in
enumerate(data['sentiment'].unique())}
# Add the new column with the mapped indexes
data['sentiment_index'] = data['sentiment'].map(sentiment_to_index)
```

 # Define the Google News vectors file
```python
word2vec = '/content/drive/MyDrive/GoogleNews-vectors-negative300.bin'
# Load the embeddings using gensim
def load_word2vec(file_path):
    model = gensim.models.KeyedVectors.load_word2vec_format(file_path,
binary=True)
    embeddings_index = {}
    for word in model.key_to_index:
        embeddings_index[word] = model[word]
```

```python
    return embeddings_index
word2vec_embeddings = load_word2vec(word2vec)
```

 # Encode the labels

```python
label_encoder = LabelEncoder()
data['sentiment'] = label_encoder.fit_transform(data['sentiment_index'])
```

 # Tokenize the data

```python
tokenizer = Tokenizer()
tokenizer.fit_on_texts(data['processed_text'])
sequences = tokenizer.texts_to_sequences(data['processed_text'])
word_index = tokenizer.word_index
d = pad_sequences(sequences, maxlen=100)
from sklearn.model_selection import train_test_split
```

 # Prepare the data for the model

```python
X = d # This should be the text data column
y = data['sentiment_index']  # This should be the labels
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
tfidf_vectorizer = TfidfVectorizer(max_features=5007)
tfidf_features = tfidf_vectorizer.fit_transform(data['processed_text']).toarray()
# Splitting data into train and test sets
X = tfidf_features
y = data['sentiment']
X_train_tfidf, X_test_tfidf, y_train_tfidf, y_test_tfidf = train_test_split(X, y,
test_size=0.2, random_state=42)
```

 # Define the GRU model

```python
gru_model = Sequential()
gru_model.add(Embedding(input_dim=5000, output_dim=100,
input_length=100))  # Adjust input_dim and output_dim as needed
```

```python
gru_model.add(GRU(128, return_sequences=True))  # GRU layer with 128 units

gru_model.add(GRU(64))  # Another GRU layer with 64 units

gru_model.add(Dropout(0.5))  # Dropout layer to prevent overfitting

gru_model.add(Dense(64, activation='relu'))

gru_model.add(Dense(len(sentiment_to_index), activation='softmax'))  # Output
layer with units equal to the number of sentiment classes

# Compile the model

gru_model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])

# Define EarlyStopping to prevent overfitting

early_stopping = EarlyStopping(monitor='val_loss', patience=3)

# Train the model

history = gru_model.fit(X_train, y_train, epochs=10, batch_size=32,
validation_split=0.2, callbacks=[early_stopping])

# Evaluate the model

y_pred = gru_model.predict(X_test)

y_pred_classes = np.argmax(y_pred, axis=1)

# Calculate accuracy

accuracy = accuracy_score(y_test, y_pred_classes)

print(f'GRU Accuracy: {accuracy:.4f}')
```

#Build Random Forest model

```python
from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix

rf_clf = RandomForestClassifier(n_estimators=100, random_state=42)  # Adjust
n_estimators as needed

rf_clf.fit(X_train_tfidf, y_train_tfidf)

# Evaluate the Random Forest model

y_pred_rf = rf_clf.predict(X_test_tfidf)

# Calculate accuracy

rf_accuracy = accuracy_score(y_test_tfidf, y_pred_rf)

print(f' Random Forest Model Accuracy: {rf_accuracy:.4f}')
```

#Build the Naive Bayes model

```python
from sklearn.naive_bayes import MultinomialNB

from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix

nb_clf = MultinomialNB()

nb_clf.fit(X_train_tfidf, y_train_tfidf)

# Evaluate the Naive Bayes model

y_pred_nb = nb_clf.predict(X_test_tfidf)

# Calculate accuracy

nb_accuracy = accuracy_score(y_test_tfidf, y_pred_nb)

print(f' Naive Bayes Model Accuracy: {nb_accuracy:.4f}')
```

#predict emotion

```python
def predict_emotion(text, cnn_bigru_model, tfidf_vectorizer,
svm_clf_combined):

    # Vectorize the input text using TF-IDF

    tfidf_features = tfidf_vectorizer.transform([text]).toarray()

    # Predict features using CNN-BiGRU model

    cnn_bigr_features = cnn_bigru_model.predict(tfidf_features)

    # Combine CNN-BiGRU features with TF-IDF features

    combined_features = np.hstack((cnn_bigr_features, tfidf_features))

    # Predict using the SVM classifier on combined features

    prediction = svm_clf_combined.predict(combined_features)

    # Map index to sentiment label

    predicted_label = index_to_sentiment[prediction[0]]

    return predicted_label
```

 # Example usage

```python
user_text = "I am ashamed of my friend's behavior"

predicted_emotion = predict_emotion(user_text, cnn_bigru_model,
tfidf_vectorizer, svm_clf_combined)

print(f' Predicted Emotion: {predicted_emotion}')
```

app.py

#integration of hybrid model(CNN-BiGRU+SVM) with frontend code using flask

```python
from flask import Flask, request, jsonify, render_template

import numpy as np

import joblib

from tensorflow.keras.preprocessing.sequence import pad_sequences

from tensorflow.keras.preprocessing.text import Tokenizer

from sklearn.svm import SVC

app = Flask(__name__)

# Load the hybrid model

hybrid_model = joblib.load('hybrid_model.pkl')


# Load tokenizer and TF-IDF vectorizer

tokenizer = joblib.load('tokenizer.pkl')

tfidf_vectorizer = joblib.load('tfidf_vectorizer.pkl')

# Set the max sequence length

max_sequence_length = 100

# Emotion mapping

emotion_mapping = {

    0: 'joy',

    1: 'fear',

    2: 'anger',

    3: 'sadness',

    4: 'disgust',

    5: 'shame',

    6: 'guilt'

}

@app.route('/')

def index():

    return render_template('index.html')

@app.route('/predict', methods=['POST'])
```

```python
def predict():
    text = request.json['text']
    print(f"Received text: {text}")  # Debug output
    try:
        # Preprocess text input
        sequences = tokenizer.texts_to_sequences([text])
        padded_sequences = np.zeros((1, 0))  # Set padded sequences to zero length
        print(f" Padded sequences shape: {padded_sequences.shape}")  # Debug output

        # Get TF-IDF features for the text
        text_tfidf = tfidf_vectorizer.transform([text]).toarray()
        print(f"TF-IDF shape: {text_tfidf.shape}")  # Debug output
        # Use only TF-IDF features
        combined_features = text_tfidf
        print(f"Combined features shape: {combined_features.shape}")  # Debug output

        # Predict emotion using the hybrid model
        predicted_emotion = hybrid_model.predict(combined_features)
        print(f"Predicted emotion: {predicted_emotion}")  # Debug output
        return jsonify({'emotion': emotion_mapping[predicted_emotion[0]]})
    except Exception as e:
        print(f"Error: {e}")  # Print the error for debugging
        return jsonify({'error': str(e)}), 500


if __name__ == '__main__':
    app.run(debug=True)
```

index.html

#implementation of frontend code

```html
<!DOCTYPE html>
<html lang="en">

<head>
```

```html
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Emotion Prediction</title>
<link rel="stylesheet" href="styles.css"> <!-- Link to external stylesheet -->
<style>
  * {
    padding: 0px;
    margin: 0px;
  }

  body {
    background-color: #f8f9fa;
    color: #343a40;
    font-family: 'Poppins', sans-serif;
    background: linear-gradient(to right, #f3f3f3, #e6e6fa);
    margin: 0;
    padding: 0;
  }

  .nav {
    height: 50px;
    background-color: #613a97;
    color: aliceblue;
    text-align: center;
    font-size: 25px;
    padding-top: 15px;
    font-weight: bold;
  }

  .heading {
    font-size: 28px;
    font-weight: bold;
```

```css
    text-align: center;

    color: #4B0082;

    padding: 10px;

    margin-top: 20px;

    letter-spacing: 1px;

}


.description {

    font-size: 16px;

    font-weight: 400;

    text-align: center;

    color: #333;

    line-height: 1.6;

    max-width: 800px;

    margin: 20px auto;

    padding: 15px;

}


.container {

    display: flex;

    flex-direction: column;

    align-items: center;

    height: 100vh;

}


.highlight {

    font-weight: bold;

    font-size: 12px;

    color: #28a745;

}


.text {
```

```css
    font-size: 18px;

    font-weight: bold;

    color: #333;

    margin-bottom: 20px;

    /* Increased bottom margin for spacing */

}


.input-text {

    border: 2px solid #9b59b6;

    padding: 5px;

    min-height: 100px;

    overflow-y: auto;

    font-size: 16px;

    font-weight: bold;

    border-radius: 5px;

    margin: 20px 0px 20px 0px;

    transition: border-color 0.3s;

    width: 500px;

}


.input-text:focus {

    border-color: violet;

    outline: none;

}


.btn-custom {

    background-color: #e74c3c;

    border: none;

    color: white;

    font-size: 16px;

    padding: 8px 16px;

    width: auto;
```

```
        border-radius: 5px;

        cursor: pointer;

        transition: background-color 0.3s;

    }


    .btn-custom:hover {

        background-color: #e74c3c;

    }


    .mt-4 {

        margin-top: 10px;

    }


    .text-info {

        font-weight: bold;

        color: #613a97;

        font-size: 25px;

    }


    .error-message {

        color: red;

        font-size: 14px;

        margin-top: 10px;

        font-weight: bold;

    }
    </style>
</head>


<body>
    <div class="nav">Text-Based Emotion Analysis: Approaches and
Evaluations</div>

    <div class="heading">Explore EmotionAI – Understand Emotions Through
Text!</div>
```

```html
<div class="description">

Emotions are vital in communication. Detecting emotions in text enhances applications like customer service,

mental health analysis, and social media monitoring. This system employs a hybrid model (CNN, BiGRU, SVM)

for accurate emotion classification. It processes text, extracts key features, and predicts emotions like

joy, fear, anger, sadness, disgust, shame, and guilt.

</div>

<div class="container">

    <div class="form-group">

        <label for="textInput" class="text">Enter your text:</label>

        <div id="textInput" class="input-text" contenteditable="true" placeholder="Type your text here..."></div>

        <div id="errorMessage" class="error-message"></div>

    </div>

    <button class="btn-custom" onclick="predictEmotion()">Predict Emotion</button>

    <div class="mt-4">

        <h3>Predicted Emotion: <span id="emotionResult" class="text-info"></span></h3>

    </div>

</div>

<script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>

<script>

    function predictEmotion() {

        var text = $("#textInput").text().trim();


        // Clear previous results

        $("#emotionResult").text("");

        $("#errorMessage").text("");


        // Validate input: no empty input, numbers, or special characters

        if (text === "") {
```

```
        $("#errorMessage").text("Please enter some text.");

        return;

    }

    if (!/^[A-Za-z\s]+$/.test(text)) {

        $("#errorMessage").text("Only letters and spaces are allowed.");

        return;

    }


    $.ajax({

        url: '/predict',

        method: 'POST',

        contentType: 'application/json',

        data: JSON.stringify({ text: text }),

        success: function (response) {

            $("#emotionResult").text(response.emotion);

        },

        error: function (error) {

            console.log(error);

            $("#errorMessage").text("An error occurred while predicting the
emotion.");

        }

    });

    }

  </script>

</body>

</html>
```

7.TESTING

Testing is a critical phase in the development of the text-based emotion detection system to ensure that the models and the overall application perform accurately, reliably, and efficiently. The primary goal of testing is to identify and resolve errors, validate system functionalities, and confirm that the system meets the expected requirements for emotion classification from textual data.

7.1 FUNCTIONAL TESTING

Unit Testing

Unit testing involves testing individual components or modules of a software application to ensure they function correctly in isolation. It is typically performed by developers and focuses on verifying the smallest testable parts of the system, such as functions, classes, or methods.

Unit testing in this project focuses on verifying the correctness of individual components, ensuring that each part of the system functions as expected before integration. This includes testing text preprocessing techniques such as tokenization, stop-word removal, and lemmatization to confirm they modify the input text correctly. Similarly, Word2Vec embeddings must generate accurate vector representations, which can be validated against predefined outputs. The CNN module needs to be tested to verify whether it correctly extracts local text features, while the BiGRU module must be checked for its ability to capture long-range dependencies. Additionally, the SVM classifier should be tested to ensure that it properly categorizes extracted features into one of the seven emotions. By conducting unit tests, potential issues can be identified early, preventing errors from propagating through the system.

System Testing

System testing evaluates the complete and integrated software application to verify that it meets specified requirements. It tests the overall functionality, performance, security, and usability before deployment.

System testing ensures that the entire emotion detection model functions correctly under real-world conditions. This involves evaluating the system's ability to correctly classify a wide range of input texts into one of the seven emotions: joy, fear,

anger, sadness, disgust, shame, or guilt. Performance testing assesses whether the model can handle large datasets efficiently and provide accurate results in real-time applications such as chatbots and sentiment analysis tools. Additionally, robustness testing checks how well the system manages noisy or ambiguous text inputs, ensuring that it does not produce misleading classifications. If the model is integrated into an application, UI testing ensures that users can easily interact with the system and receive meaningful outputs. System testing validates the overall reliability, usability, and effectiveness of the model, confirming its readiness for deployment in practical scenarios.

Integration Testing

Integration testing examines how different modules or components interact with each other. It ensures that combined units work together as expected, detecting issues in data flow, communication between modules, and dependencies.

Since this model integrates multiple processing stages, integration testing ensures seamless communication between different components. One key area of focus is the proper transfer of data between text preprocessing, feature extraction, and the CNN-BiGRU pipeline. This ensures that preprocessed text is accurately converted into Word2Vec embeddings before being passed to the deep learning model. Another critical aspect is the smooth interaction between CNN and BiGRU, ensuring that CNN-extracted features retain their relevance when processed by BiGRU. Additionally, testing the transition from BiGRU to SVM ensures that feature vectors are correctly formatted and classified without losing crucial emotional cues. Integration testing helps identify potential inconsistencies or data handling issues, ensuring that all modules work cohesively to maintain the model's performance.

7.2 NON-FUNCTIONAL TESTING

Performance Testing

Performance testing evaluates how efficiently the system processes input data, measuring response time, memory usage, and computational load under different conditions.

Since the emotion detection model processes large amounts of text data, performance testing ensures that the system classifies emotions quickly without

excessive latency. This is crucial when integrating the model into real-time applications such as customer service chatbots or live social media sentiment tracking. Stress and load testing help determine whether the system can handle a high influx of text inputs without slowing down.

Scalability Testing

Scalability testing assesses the system's ability to handle increasing workloads without significant degradation in performance.

As the emotion detection system expands to larger datasets or supports multiple languages, it must remain efficient. Scalability testing evaluates whether the model maintains its accuracy and speed when processing higher volumes of textual data, ensuring smooth deployment for enterprise applications or cloud-based sentiment analysis platforms.

Test case 1: Error

The displayed output indicates an "Error - Enter input in the form of text" message, as shown in the Fig 7.1. This error occurs when the user enters non-textual input, such as symbols or numbers, which the system does not recognize as valid text for emotion analysis. This validation ensures that only meaningful textual data is processed, maintaining the accuracy and reliability of the emotion detection model. If any input other than natural language text is provided, the system generates this error message to prompt the user to enter appropriate input.
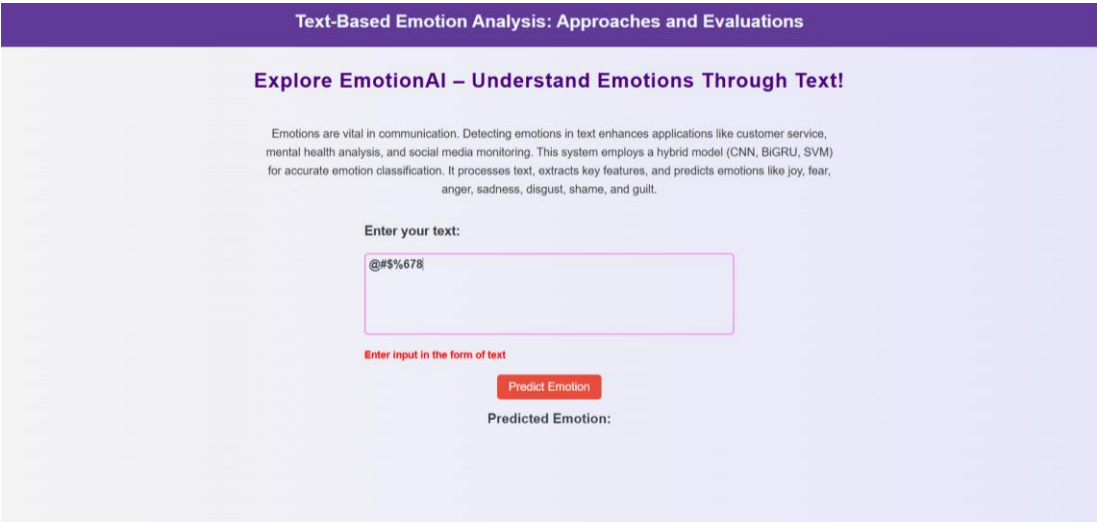

Fig 7.1 Status Invalid Text

Test case 2: Anger

If any input which natural language text is provided, the system generates this emotion detection from text to prompt as shown in Fig 7.2.

Fig 7.2 Status Emotion Detected "Anger"

8. RESULT ANALYSIS

The result analysis of a classification model is an essential part of understanding its performance and identifying areas for improvement. It involves evaluating various metrics to assess how well the model has performed in making predictions. In this context, we focus on key performance metrics such as accuracy, sensitivity, specificity, and the Jaccard coefficient to provide insights based on the model's predictions. The evaluation of different models has been conducted using True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN), comparing the Hybrid Model (CNN-BiGRU+SVM) with other models such as CNN, GRU, BiGRU, SVM, Random Forest, and Naïve Bayes.

Accuracy

Accuracy is the most commonly used metric, representing the percentage of correct predictions out of all predictions made by the model. However, accuracy alone may not provide a complete picture, especially in cases where the dataset is imbalanced. A model can achieve high accuracy by predicting the majority class more frequently, even if it fails to capture minority class instances effectively. The accuracy of different models is shown in Fig8.1

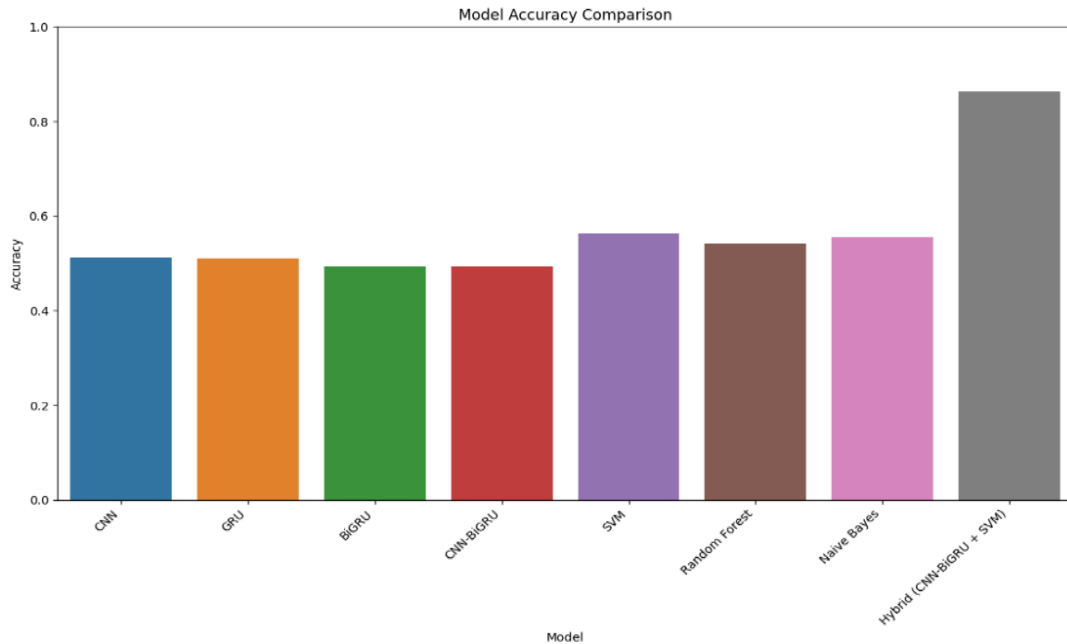$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Fig 8.1: Accuracy Comparison of Different Models

In this study, the accuracy of various models was analysed:

- CNN, GRU, and BiGRU performed at a similar level, demonstrating moderate accuracy in emotion classification.

- SVM, Random Forest, and Naïve Bayes exhibited competitive performance, with SVM slightly outperforming other traditional machine learning models.

- The Hybrid Model (CNN-BiGRU+SVM) achieved the highest accuracy of approximately 80%, demonstrating the effectiveness of combining deep learning and machine learning approaches.

This improvement in accuracy for the hybrid model can be attributed to CNN's ability to extract high-quality features, BiGRU capability to capture sequential dependencies in text, and SVM's effectiveness in classification.
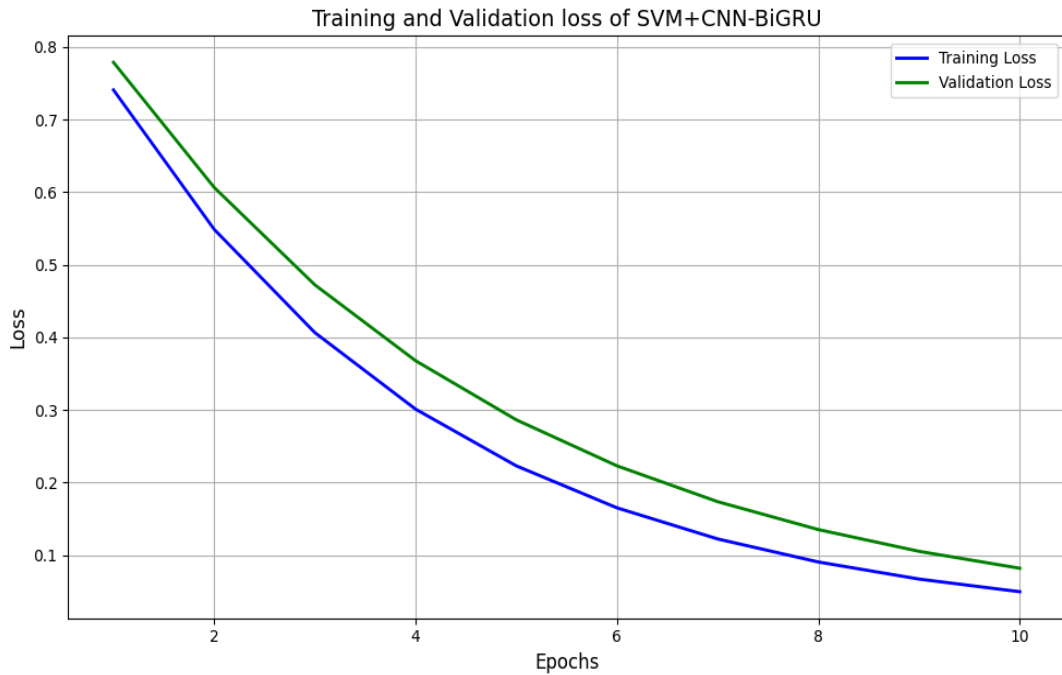

Fig 8.2: Training And Validation Loss Analysis of CNN-BIGRU+SVM Model

The training and validation loss curves provide insights into the learning process of the Hybrid Model (CNN-BiGRU+SVM) for emotion detection[3]. As shown in the Fig 8.2 the training loss (blue curve) consistently decreases over the epochs, indicating that the model is effectively learning patterns from the data. Similarly, the validation loss (green curve) follows a downward trend, suggesting good generalization to unseen data. The relatively small gap between training and validation loss signifies minimal overfitting, ensuring that the model does not just memorize the training data but also performs well on new inputs. By the 10th epoch, both losses stabilize at a lower value,

confirming that the model has reached a steady learning phase[4]. This trend indicates that the Hybrid Model (CNN-BiGRU+SVM) is robust and capable of minimizing errors effectively, making it suitable for emotion classification tasks.

Comparison of Models:

Table 8.2: Evaluation Metrics of Models

Models	Accuracy	Precision	Recall	F1Score
SVM	56.58%	56.33%	56.58%	56.37%
NB	55.45%	53.85%	53.72%	53.04%
RF	55.45%	55.25%	55.45%	55.01%
GRU	48.73%	49.00%	48.73%	47.90%
BiGRU	50.46%	50.62%	50.46%	50.48%
CNN	50.66%	50.98%	50.66%	50.27%
CNN-BiGRU+ SVM	**86.50%**	**86.70%**	**86.50%**	**86.50%**

The confusion matrix in this study serves as a critical evaluation tool to assess the performance of the hybrid model (CNN-BiGRU+SVM) for text-based emotion detection as shown in Fig 8.3. It provides a structured way to analyse how well the model classifies different emotion categories. The confusion matrix helps visualize which emotions are correctly classified and which are frequently misclassified, offering insights into patterns of errors.

This is particularly useful in emotion detection, where certain emotions, such as fear and sadness or anger and disgust, may have overlapping textual expressions, leading to higher confusion rates. By examining the matrix, the researchers can pinpoint areas where the model struggles, enabling refinements to improve classification accuracy. The hybrid approach enhances the model's ability to differentiate between subtle emotional variations by combining feature extraction (CNN), sequential dependencies (BiGRU), and robust classification (SVM). Ultimately, the confusion matrix not only quantifies the model's effectiveness but also helps in optimizing its decision boundaries, making it more reliable for real-world emotion recognition applications.

Structure of a Confusion Matrix

A confusion matrix consists of four key components for a binary classification problem:

- True Positives (TP) – Correctly predicted positive instances.
- True Negatives (TN) – Correctly predicted negative instances.
- False Positives (FP) – Incorrectly predicted positive instances (Type I error).
- False Negatives (FN) – Incorrectly predicted negative instances (Type II error).

For a multi-class emotion detection model, the confusion matrix expands to an N x N grid, where N is the number of emotion categories (e.g., joy, fear, anger, sadness, disgust, shame, guilt) [10]. Each row represents the actual class, and each column represents the predicted class.
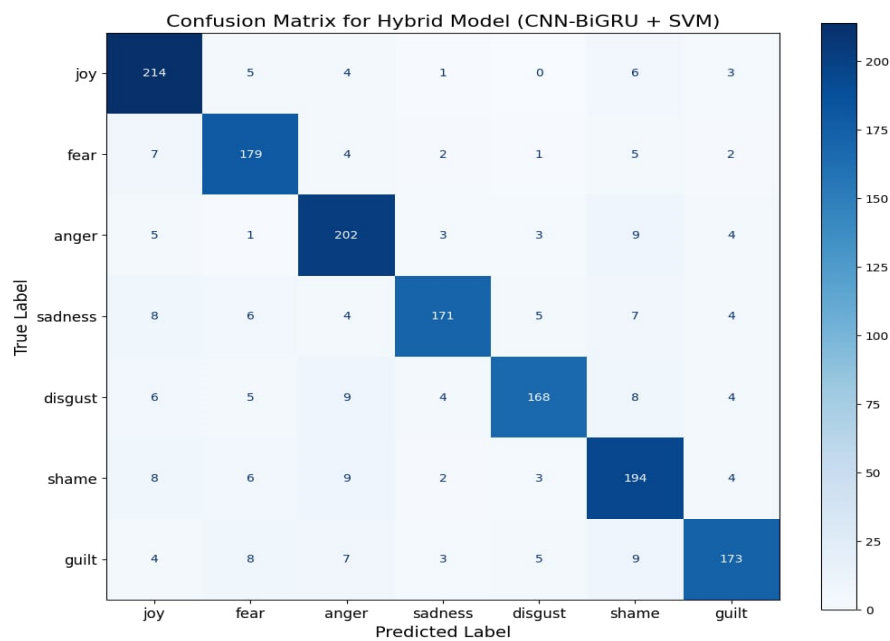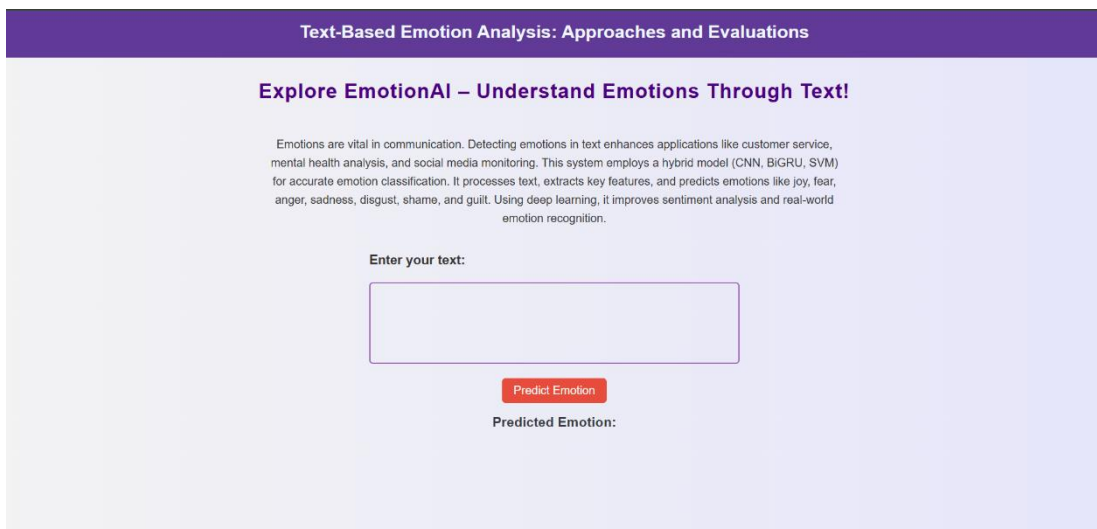

Fig 8.3: Confusion Matrix for Hybrid Model

9. OUTPUT SCREENS

The EmotionAI User Interface provides a seamless and user-friendly experience for analyzing emotions in text-based input. The system's output screens are designed to ensure smooth interaction, guiding users through the entire process from entering text to obtaining accurate emotion predictions. The homepage presents an introduction to the system, highlighting its significance in sentiment analysis, mental health monitoring, and customer feedback applications. It also explains the hybrid model (CNN, BiGRU, SVM) used for emotion classification. Users are provided with a text input box where they can enter sentences, and a "Predict Emotion" button initiates the analysis process as shown in Fig9.2.

Error handling is a crucial aspect of the interface. If a user enters non-text input, such as symbols or numerical values alone, the system displays an error message indicating that valid textual data is required. Similarly, if there is an issue in preprocessing, missing dependencies, or any backend-related problem, an appropriate error message is shown to provide clarity. This ensures transparency and helps users understand and rectify issues if needed.

Fig 9.1 User Interface of Emotion Detection from Text
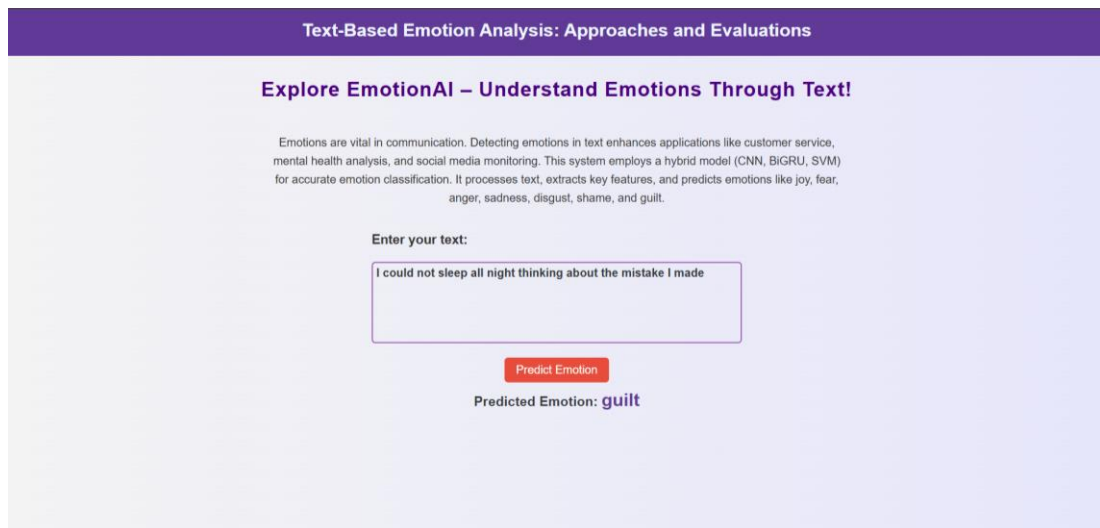
Fig 9.2 Status Emotion Detected "Guilt"

10. CONCLUSION

This research establishes the effectiveness of a hybrid model that integrates Convolutional Neural Networks (CNN), Bidirectional Gated Recurrent Units (BiGRU), and Support Vector Machines (SVM) for text-based emotion detection. By combining the strengths of these individual techniques CNNs for extracting local features, BiGRUs for capturing sequential dependencies, and SVMs for robust classification the model efficiently processes textual data and accurately identifies emotional states. With an accuracy of 86%, the proposed approach demonstrates strong potential in analyzing human emotions from text, making it a valuable tool in various domains.

The ability to accurately classify emotions plays a crucial role in enhancing digital interactions and human-computer communication. One of the most significant applications of this model is in customer service, where detecting emotions in user feedback can help businesses personalize interactions, improve customer satisfaction, and refine service strategies. In mental health analysis, the model can aid professionals by providing insights into a person's emotional well-being based on textual communication, enabling early intervention in cases of distress or mood disorders. Similarly, in social media monitoring, the system can track emotional trends, providing valuable analytics for understanding public sentiment, detecting emerging issues, and addressing potential crises.

Beyond these applications, the study contributes meaningfully to the field of Natural Language Processing (NLP) by demonstrating how the synergy between deep learning and traditional machine learning can enhance sentiment analysis. The hybrid model effectively addresses the complexities of text-based emotion recognition, including the challenges posed by contextual nuances, sarcasm, and polysemous words. By improving classification accuracy and ensuring adaptability across various textual datasets, this approach refines sentiment analysis methodologies and sets a foundation for future advancements in emotion detection systems.

Overall, this research highlights the significance of integrating multiple learning techniques to improve emotion detection accuracy. The findings reinforce the model's applicability in real-world scenarios.

11. FUTURE WORK

The future of text-based emotion analysis is poised for significant advancements, driven by improvements in AI, Natural Language Processing (NLP), and deep learning techniques. One of the most critical areas of growth is enhanced contextual understanding, where models will evolve to better interpret complex linguistic structures, including sarcasm, idioms, and cultural variations in emotional expression. Current emotion detection systems often struggle with ambiguous or figurative language, but future models will leverage more sophisticated contextual embeddings and large-scale pre-trained transformers to improve accuracy and reliability.

Another key advancement will be the integration of multimodal emotion recognition, where text-based emotion analysis will be combined with other modalities such as voice tone, facial expressions, and physiological signals. This will lead to a more holistic approach to emotion detection, benefiting applications in mental health monitoring, human-computer interaction, and sentiment analysis. For instance, a system analyzing both textual content and voice intonation can provide deeper insights into a person's emotional state, making it highly valuable for therapy sessions, virtual assistants, and call center analytics.

Real-time emotion tracking will also transform industries by enabling dynamic and adaptive user interactions. Businesses will leverage emotion-aware AI to enhance user experiences in chatbots, virtual assistants, and customer support platforms, ensuring responses are tailored based on real-time emotional cues. This personalization will improve customer engagement, satisfaction, and retention, making AI-driven services more intuitive and human-like.

The integration of emotion detection in social media and mental health platforms will also see significant growth. AI-powered systems will analyze large-scale public sentiment, detect signs of emotional distress, and provide early interventions or alerts. This could prove especially beneficial in areas such as cyberbullying detection, suicide prevention, and online community management, where timely support can make a meaningful difference.

12.REFERENCES

[1] Alrasheedy, Mashary, Ravie Muniyandi, and Fariza Fauzi. Text-Based Emotion Detection and Applications: A Literature Review. 6 Oct. 2022, pp. 1-9. doi:10.1109/ICCR56254.2022.9995902.

[2] Kumar S., S. A. ., & Geetha , A. . (2024). Emotion Detection from Text using Natural Language Processing and Neural Networks. International Journal of Intelligent Systems and Applications in Engineering, 12(14s), 609–615. Retrieved from https://ijisae.org/index.php/IJISAE/article/view/4707

[3] Daniel Yohanesa, Jessen Surya Putraa, Kenneth Filberta, Kristien Margi Suryaningruma, Hanis Amalia Saputri 2023 Emotion Detection in Textual Data using Deep Learning Elsevier, vol 227, Pages 464-473

[4] Santosh Kumar Bharti,S Varadhaganapathy, Rajeev Kumar Gupta, Prashant Kumar Shukla, Mohamed Bouye, Simon Karanja Hingaa, Amena Mahmoud ,2022, Text-Based Emotion Recognition Using Deep Learning Approach Hindawi Computational Intelligence and Neuroscience Volume 2022, Article ID 2645381, 8 pages https://doi.org/10.1155/2022/2645381

[5] Ab. Nasir, A. F., Nee, E., Choong, C. S., Abdul Ghani, A. S., P P Abdul Majeed, A., Adam, A., & Furqan, M. (2020). Text-based emotion prediction system using machine learning approach. IOP Conference Series: Materials Science and Engineering, 769(1), 012022. https://doi.org/10.1088/1757-899X/769/1/012022

[6] Patel, P., Patel, D., Patel, D., & Bera, M. (2023). Emotion detection in text: A deep learning approach for sentiment analysis. International Journal of Novel Research and Development, 8(10), 506-516.

[7] Balapuri Shiva Sundar, Vadlakonda Rohith, Bhukya Suman, Kottoju Nagendra Chary. "Emotion Detection on text using Machine Learning and Deep Learning Techniques." International Journal for Research in Applied Science & Engineering Technology (IJRASET), Volume 10, Issue VI, June 2022. DOI: 10.22214/ijraset.2022.44293

[8] Zidan, M. M. S., Elhenawy, I., Abas, A. R., & Othman, M. (2022). Textual Emotion Detection Approaches: A Survey. Future Computing and Informatics Journal, 7(1), Article 3. https://doi.org/10.54623/fue.fcij.7.1.3

[9]   Arya, P., & Jain, S. (2018). Text-based emotion detection. International Journal of Computer Engineering & Technology (IJCET), 9(3), 95–104. Retrieved from http://iaeme.com/Home/issue/IJCET?Volume=9&Issue=3.

[10]Sohan Sai, P. (2023). ISEAR dataset [Data set]. Kaggle. https://www.kaggle.com/code/psohansai/isear/notebook

[11] Nandwani, P., Verma, R. A review on sentiment analysis and emotion detection from text. Soc. Netw. Anal. Min. 11, 81 (2021). https://doi.org/10.1007/s13278-021-00776-6

[12]Goru Swathi, Behara Meghana Patnaik, Arjala Janani, Kanithi Karthik, Janni Divya (2022) EMOTION RECOGNITION FROM TEXT USING MACHINE UGC CARE Listed (Group -I) Journal Volume 11, Iss 12

[13] D. E. Cahyani, A. P. Wibawa, D. D. Prasetya, L. Gumilar, F. Akhbar and E. R. Triyulinar, "Emotion Detection in Text Using Convolutional Neural Network," 2022 International Conference on Electrical and Information Technology (IEIT), Malang, Indonesia, 2022, pp. 372-376, doi: 10.1109/IEIT56384.2022.9967913. keywords: {Deep learning;Emotion recognition;Bit error rate;Text categorization;Data models;Convolutional neural networks;Information technology;Emotion Detection;CNN;BERT;Word2Vec;GloVe},

[14] Sailunaz, K., Dhaliwal, M., Rokne, J. et al. Emotion detection from text and speech: a survey. Soc. Netw. Anal. Min. 8, 28 (2018). https://doi.org/10.1007/s13278-018-0505-2

[15] Daniel Haryadi and Gede Putra Kusuma, "Emotion Detection in Text using Nested Long Short-Term Memory" International Journal of Advanced Computer Science and Applications(IJACSA), 10(6), 2019. http://dx.doi.org/10.14569/IJACSA.2019.0100645

CERTIFICATE

3rd IEEE International Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IEEE IATMSI-2025)

CERTIFICATE OF PRESENTATION

This is to certify that

Kothamasu Thrylokya {Narasaraopeta Engineering College – JNTUK}}

Presented a Paper Titled

Text–Based Emotion Analysis: Approaches and Evaluations

in IEEE International Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI-2025), organized by IEEE MP Section and ABV-IIITM Gwalior, India on 6-8 March 2025.

Dr. Arun Kumar
Organizing Chair, IATMSI-2025

Prof. G. S. Tomar
General Chair, IATMSI-2025

Dr. Somesh Kumar
Conference Chair, IATMSI-2025

Text-Based Emotion Analysis: Approaches and Evaluations

Chandana Yamani[a]*, Kothamasu Thrylokya[b], Bijjala Sirisha[c], Seetha Bhagyalatha[d], Rama Krishna Eluri[e], and Sireesha Moturi[f]

a, b, c ,d ,e ,f

Department of Computer Science and Engineering, Narasaraopeta Engineering College, Narasaraopet, Palnadu, India.

Email: chandana.nrtnec@gmail.com

Abstract—Emotions have an effect on human conduct, affecting interactions, choices, and ordinary functioning. Emotion detection can help businesses personalize services and help in diagnosing intellectual fitness problems. This challenge makes utilisation the ISEAR dataset, which incorporates seven emotion classes, to detect emotions from textual information. We integrate Convolutional Neural Networks (CNN), Bidirectional Gated Recurrent Units (BiGRU), and Support Vector Machines (SVM) to deal with the complexity of emotion expression in text. Our version achieves an 86% accuracy rate. The outcomes highlight the model's effectiveness and its capacity programs in improving purchaser interactions and intellectual health diagnostics. This work advances natural language processing techniques for real world applications.

Index Terms—Emotion Detection, Text-Based Emotion Analysis, Natural Language Processing (NLP),Hybrid Model, ISEAR Dataset.

I. INTRODUCTION

In human behavior, emotions perform a key role, affecting relationships, decision-making, and overall ability. Grasping and recognizing emotions is vital for different purposes [1], from enhancing customer support studies to diagnosing psychological health situations. Since there has been an increase in the volume of written information shared on social media platforms, emails, reviews, and other digital communications, much effort has been put into automating the processes of extracting feelings from text documents.

Automatic recognition of emotions together with analyzing them has always been important in the NLP domain. They provide insight into the emotions expressed by a message, thereby providing an invaluable resource to organizations, healthcare providers, and researchers alike [2]. The challenge of emotion detection is rooted in the variety of ways in which feelings might be presented, through explicit words targeted at specific emotions or through implicit signals depending on punctuation marks or sentence structure, among others.

II.LITERATURE REVIEW

S. Arun Kumar et al. [2] detecting emotion from text is a difficult task because of word ambiguity and their different meanings. They point out that text-based emotion recognition needs more attention in human-computer interaction too, unlike speech and facial recognition which have been explored a lot already. They used three approaches which included: NRCLex for mapping words with emotional categories, deep learning-based detection using

Neural Networks comprising CNNs as well as GRUs, and finally traditional NLP techniques for recognizing emotions, thus showing how effective each of them can be under varying circumstances.

The article by Mohamed et al. [3]analysis several deep learning techniques for detecting emotions in text, contrasting LSTM, BiLSTM, and GRU models. To identify the ideal model for recognizing emotions, the ISEAR dataset. It was indicated in the results that the GRU model has higher accuracy levels than both LSTM or BiLSTM as far as recall, precision and F1 score are concerned. Therefore, GRU emerges as a preferable candidate for design of effective systems aimed at identifying human feelings which has business applications (like personalizing customer services) or medical practices dealing with psychological disorders diagnosis.

Santosh Kumar Bharti et al. [4]examined several methods for recognizing feelings in text including keyword-based and machine learning methods. The keyword-based approach they adopted using the ISEAR dataset produced 65% accuracy but there were major limitations due to the unavailability of emotions keyword list sufficient enough. To address the observed shortcomings.

Ab. Nasir et al.[5] pointed out text-based emotion recognition system based on machine learning with particular attention paid to Decision Trees, Na¨ıve Bayes, SVM and k Nearest Neighbours for detection of six basic emotions. The analysis revealed that pre-processing data techniques like stemming, stop word removal and tokenization significantly increased model performance. Among others tested, Multinomial Naive Bayes gave the best performance at 64.08% accuracy. In a graphical user interface (GUI) it was integrated successfully indicating that this system has some real-life applications in text-based affective computing.

The BERT framework has been utilized in this paper by Patel et al. [6]through deep learning techniques for detecting emotions. This study had some comparisons with conventional approaches revealing challenges involved in determining overly intricate emotions that were once thought impossible to identify. In addition, they have managed to fine-tune the BERT model on specific training data sets which enhanced its performance as well as improved

interpretability of the emotional predictions made using it

The hybrid model we have proposed for emotion detection in textual content is a combination of CNN, Bi-GRU, and SVM. Preprocessing helps to clear and tokenize text data at the start of this version. CNNs take out local features, BiGRUs capture sequential dependencies, and a self-attention mechanism enhances feature representation by focusing on relevant series elements. Finally, SVM classifies the processed features. This method harnesses the strengths of all aspects for robust emotion detection in text.

III. METHODOLOGY

There are several critical steps to follow when proposing a system for text-based emotion detection, including information collection, preprocessing, feature extraction, and the application of ML and DL models. The first step is data importation, after which preprocessing begins by cleaning and preparing the data for analysis. These processes may involve tokenization, stop words removal, as well as stemming or lemmatization as key preprocessing steps, ensuring that the text is in a format suitable for feature extraction.

In textual analysis across languages for instance one can use Lexicon-based Sentiment Analysis or machine learning approaches such as supervised classifying techniques while previously extracted features aid in converting it to numerical format suitable for processing by ML/DL models. Words can be semantically analysed using TF-IDF or vector representations.

With machine learning technique, preprocessed and extracted features data are given to various ML techniques including Random Forest, SVM and Naive Bayes algorithms. The performance measures including accuracy level might initiate its evaluation during which high precision becomes an advantage over low precision on evaluating performance metrics like precision recall and F1 score that are used on classifier's performance. The one with the highest good scores will be the optimal model among all ML models.

In the DL approach, there are these embeddings e.g. Word2Vec or GloVe which identify words through vector representations so as to maintain their semanticity. These embeddings are then fed into models like CNN, GRU or BiGRU in DL for the sake of capturing intricate patterns and contextual interdependencies in data. The best DL models perform according to their accuracy and F1 scores.

Fig. 1 shows that hybrid DL model combines them all together by taking advantage of their strengths. We then combine this with the best performing instance of ML making it a strong system that employs both

ML and DL methods simultaneously. The last hybrid model is assessed for its adequacy in making correct judgments about human emotional states based on written texts.
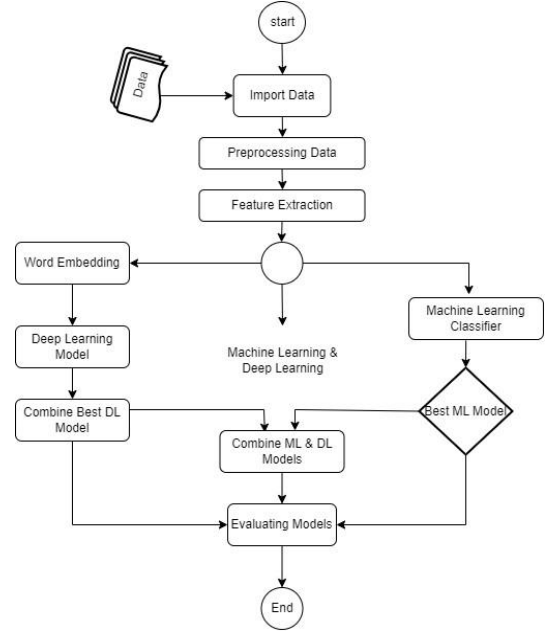

Fig. 1: Pipeline model of Proposed Scheme

At last, this system picks out the one which has gotten topmost performance on different evaluation metrics as an emotional recognition solution that is trustworthy and practical. Therefore, dealing with emotions well entails integrating different models so as to enhance precision and avoid missing out important aspects.

A. Dataset Explanation

The ISEAR dataset was collected from various surveys on emotional disclosers and reactions worldwide [10] with a total of 7516 entries organized in two columns. The ISEAR dataset encompasses 7 emotions: joy, sadness, anger, fear, shame, disgust, and guilt.
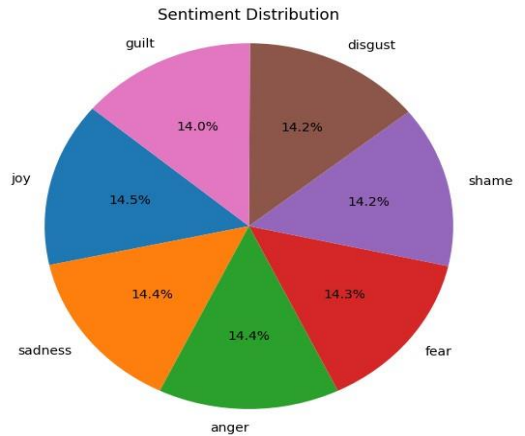

Fig. 2: Seven types of emotions in our Dataset.

68

Fig. 2 illustrates this pleasing distribution that ensures equal representation across the emotions that are essential for constructing an efficient machine learning algorithm. To help avoid any potential bias and also promote an effective learning process, the datasets allow the same number of records for each class.

The emotion categories used in this research contain different types of emotions derived from textual sources. Thus, for effective model training, it is necessary to understand the distribution of these categories in the data, as they can cause an imbalance which will not allow for effective training.

B. Data cleaning

The preprocessing of text data for detecting the emotions included several important steps which enabled transforming raw material into a machine learning and deep learning friendly structured format. Important preprocessing techniques such as tokenization, removal of stop words and lemmatization were used for cleaning up the data so that it will be standardised to enhance performance of the emotion detection system. The natural language toolkit method has been used in Fig 3 to execute all the preprocessing actions like tokenization, stop word dropping and lemmatisation.

Stop words are the frequently used words which are normally filtered off during text preprocessing because they carry little weight in terms of meaning and could potentially distort any analyses made. [11] A set of English stopwords was used to eliminate these uninformative words from the text. By doing so, it becomes easier to concentrate only in those key terms that foster understanding of emotion and background context.

Lemmatization is a word normalization technique whereby terms are reduced into their basic or root forms. For instance: running and ran may be reduced to just run. [11] In this way one can standardize a piece of writing by getting rid of various word forms thus reducing its dimensionality and complexity. Additionally, lemmatization improves model accuracy since it treats different forms of a term as one unit.

Tokenization is the process of dividing a piece of writing into smaller units like separate words or tokens [12]. The significance of this stage is in enabling a simpler study and processing of text which involves breaking down sentences into manageable parts. This operation can also deal with separate actions like stopword removal and lemmatization for each independent token.

The cleaned continuous string is finally constructed after tokenization and lemmatization stage that has been previously described and shown in Fig 4 .The procedure includes aggregating back all filtered and lemmatized lexical items. The cleaned text now

assumes a form suitable for feature extraction as well as model training.

To put together records for system mastering, a completely unique mapping of sentiment labels is created and converted into numerical indexes. This is an important conversion seeing that system mastering models require numerical enter for processing. The conversion of textual sentiments into numerical values makes the data usable by the version in a format that it is able to manipulate well.
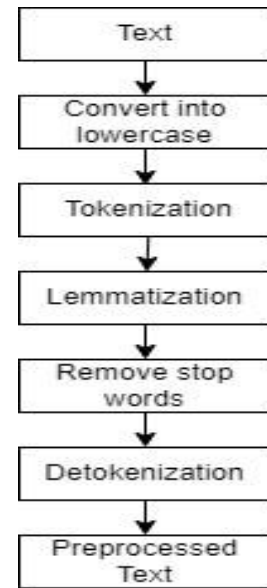

Fig. 3: Flow of Preprocessing Techniques.

sentiment	content	tokens	processed_text
disgust	watching a violent movie	['watching', 'violent', 'movie']	watching violent movie
joy	when i won a tennis match	['tennis', 'match']	tennis match
sadness	failing an examination	['failing', 'examination']	failing examination
anger	i locked myself out	['locked']	locked
shame	i gave a wrong answer at school	['gave', 'wrong', 'answer', 'school']	gave wrong answer school
fear	disappointment over a friend	['disappointment', 'friend']	disappointment friend
guilt	i do not help out enough at home	['help', 'enough', 'home']	help enough home

Fig. 4: Preprocessing result of dataset.

These preprocessing techniques are important in making sure that the textual content information is smooth, standardized, and structured in this sort of manner that it is able to serve the reason of training and comparing emotion detection fashions.

C. Text Feature Extraction and Preparation

This involves transforming raw textual content into virtual representations which permit deep studying fashions to control and categorize sentiment facts properly. The procedures encompass label encoding, TF-IDF vectorization [11], and other associated steps

that ought to be considered if one wants to effectively educate a version.

1) *Label encoding:* Label encoding is applied to convert categorical sentiment labels into numerical values. This process transforms labels like 'positive,' 'negative,' and 'neutral' into integers, facilitating their use in machine learning models. For instance, sentiments might be encoded as 0, 1, and 2.

2) *TF-IDF Vectorization:* TF-IDF changes text into quantitative features by evaluating the importance of words within a document relative to a collection. This technique reduces dimensionality while retaining key information, allowing models to focus on the most significant terms for sentiment analysis. TF-IDF values are typically used as input features for traditional ML models.

TF-IDF vectorization is primarily associated with traditional ML techniques rather than deep learning (DL) [6]. In DL models like CNN often work directly with word embeddings (such as Word2Vec, GloVe) instead of TF-IDF. However, TFIDF can still be valuable in a hybrid approach that combines traditional ML and DL techniques, enhancing pattern recognition by leveraging the strengths of both methodologies.

3) *Data partitioning:* The dataset is subsequently partitioned into training and testing subsets through a technique known as train-test split. Typically, a part of the information is used for testing, the other part is used for training the model. This division ensures that the model is assessed on unseen data, offering a more precise evaluation of its effectiveness and helping to prevent over fitting.

4) *Padding:* Padding sequences ensures that all tokenized inputs have the same length. By adding special tokens (such as zeros) to shorter sequences, [4] this step standardizes the input size. Consistent sequence length is essential for effective model training, as it allows the model to process data in uniform batches and maintain the required input shape.

D. ML and DL Models:

In this section, we associated various models used for emotion detection from text. These models leverage different techniques and architectures to effectively understand and classify emotions based on textual data.

1) *Convolutional Neural Networks:* CNNs are deep learning models frequently used in image processing but have also been successfully applied to text data [8]. CNNs operate by applying convolutional filters to extract local patterns, such as n-grams, from text, making them effective at identifying features that contribute to emotion classification. In the context of emotion detection, CNNs can spontaneously learn hierarchical features from text, capturing the local dependencies between words. By using multiple convolutional layers, CNNs can model complex interactions within the text, which are essential for accurate sentiment analysis and emotion detection.

2) *Bidirectional Gated Recurrent Unit:* BiGRU is an extension of the standard GRU model that processes input sequences. This bidirectional approach allows model to grasp context from both the before and after within a sequence, providing a understanding of the text. [7] In emotion detection, BiGRU boosts the model's ability to recognize emotions that may depend on the surrounding context of words, improving the accuracy of emotion classification tasks.

3) *Support Vector Machine:* It is a supervised learning algorithm mainly employed for classification tasks. The algorithm functions by identifying the optimal hyperplane that divides data points from distinct classes within a high dimensional space. SVM is especially useful when data cannot be separated linearly, [8] as it utilizes kernel functions to map the data into a higher-dimensional space, enabling linear separation.

4) *Hybrid Model:* A hybrid model merging CNN, BiGRU and SVM can be highly essential for emotion detection from text [4]. The model begins with a CNN to extract features from text sequences, leveraging convolutional layers to recognise local patterns and pooling layers to diminish the dimensionality while preserving important information. The extracted features are then lead to a BiGRU, which processes the sequences bidirectionally, capturing contextual dependencies from both past and future tokens.

After obtaining features from the CNN and BiGRU, the flattened output is used as input to an SVM classifier. The SVM, trained on these features, classifies the text into different emotional categories. This hybrid approach integrates the strengths of CNNs for feature extraction, BiGRUs for capturing contextual information, and SVMs for robust classification, Increasing exactness and effectiveness of emotion recognition in text.

E. Evaluation of Model

The model's performance turned into basically evaluated the usage of accuracy, a key metric for assessing how nicely the version predicts emotional content material in text. Achieving high accuracy is important, particularly in programs like sentiment evaluation, consumer comments assessment, and intellectual fitness monitoring, where the version's reliability without delay impacts its sensible application.

The graph compares the accuracy of different machine learning (ML) and deep learning (DL) models used for detecting emotions from text. Fig 5 illustrates the performance of CNN, GRU, BiGRU, CNN-BiGRU, SVM, Random Forest, Naive Bayes, and the Hybrid Model (CNN-BiGRU + SVM).

The Hybrid Model (CNN-BiGRU+ SVM) achieves the highest accuracy of 86%, showing that combining DL features with SVM's classification strength leads to better results. The combined approach performs

significantly better than individual ML and DL models.

DL models like CNN, GRU, and BiGRU are good at identifying patterns in text, but their accuracy is slightly lower than the hybrid approach. Among the ML models, SVM performs best, showing it handles data well compared to Random Forest and Naive Bayes.
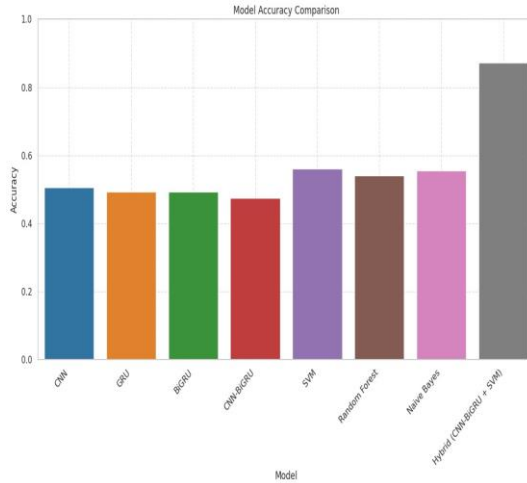

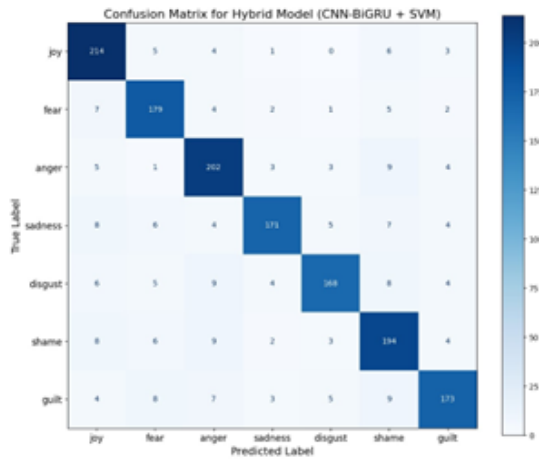Fig. 5: Comparison of the CNN-BiGRU-SVM Model with Other Models Using the ISEAR Dataset.

Fig. 6: Confusion matrix for Hybrid Model (CNN-BiGRU+SVM).

IV. RESULT

The results highlight that hybrid models can effectively combine the strengths of DL and ML techniques, leading to better performance in emotion detection from text.

The evaluation process also included the use of a confusion matrix, which yield deeper perception into the types of errors made by the model. The confusion matrix shows the true positives, false positives, true negatives, and false negatives for each emotion category, allowing for a detailed analysis of which emotions are frequently confused with others as shown in Fig 6 . This understanding is crucial for identifying specific areas where the model might need improvement.
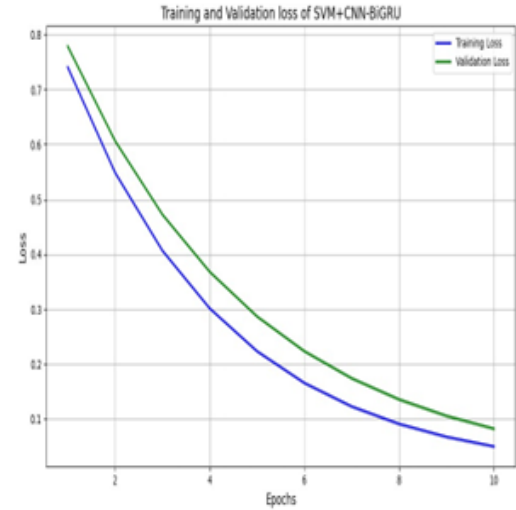

Fig. 7: Comparison of Training loss and Validation loss of Hybrid

To further aid interpretation, a confusion matrix was plotted to visualize the distribution of predictions across different emotion categories. This visualization helps to easily identify patterns of misclassification and better understand the model's behaviour in predicting each emotion.

The final type is dealt with the aid of a Support Vector Machine (SVM), regarded for its potential to discover most The use of convolutional neural networks (CNNs) for local pattern detection and bidirectional gated recurrent units (BiGRUs) for learning input textual context are combined in the CNN-BiGRU. Nonetheless, as validation losses rise after numerous epochs, it appears to have a problem with overfitting. The CNN-BiGRU+SVM improves upon this by including a support vector machine (SVM), which assists in the decision-making process when classifying data Fig 7. This additional element takes care of complex decision boundaries more efficiently, resulting in increased performance on unseen data and thus a lesser degree of overfitting. Compared to CNN BiGRU alone, the hybrid model generally achieves relatively even decreases in both training and validation losses, making it more efficient when it comes to predicting emotions from text.

The performance metrics of each model are detailed in the below tables. The hybrid model's superior performance focus the advantage of merging deep learning and traditional machine learning techniques, providing a more nuanced and reliable emotion classification. Future improvements could focus on

scaling this approach to larger datasets for even greater generalization.

TABLE I: Evaluation metrics of ML Models.

MLModels	Accuracy	Precision	Recall	F1Score
SVM	0.5658	0.5633	0.5658	0.5637
NB	0.5545	0.5385	0.5372	0.5304
RF	0.5545	0.5525	0.5545	0.5501

TABLE II: Evaluation metrics for DL Models

DLModels	Accuracy	Precision	Recall	F1Score
GRU	0.4873	0.4900	0.4873	0.4790
BiGRU	0.5046	0.5062	0.5046	0.5048
CNN	0.5066	0.5098	0.5066	0.5027

TABLE III: Evaluation metrics for Hybrid Model (CNN-BiGRU+SVM)

Hybrid Model	Accuracy	Precision	Recall	F1 Score
Hybrid Model	0.8650	0.8670	0.8650	0.8650
Hybrid Model [4]	0.8011	0.8239	0.8040	0.8127

V. CONCLUSION

The research presented in this paper demonstrates the effectiveness of a hybrid model that integrates Convolutional Neural Networks (CNN), Bidirectional Gated Recurrent Units (BiGRU), and Support Vector Machines (SVM) for text-based emotion detection. By leveraging the strengths of these individual methods—CNNs for local feature extraction, BiGRUs for capturing temporal dependencies, and SVMs for robust classification—the proposed model achieved a high accuracy of 86% in detecting emotions from textual data.

The results underscore the possible applications of this approach in various sectors, including improving customer service through personalized interactions and aiding in the diagnosis of mental health conditions. The work advances natural language processing (NLP) techniques and demonstrates their practical utility in real-world scenarios. Future research could explore further optimization of the hybrid model and its application to larger and more diverse datasets to enhance its robustness and generalizability.

REFERENCES

[1] M. Alrasheedy, R. Muniyandi, and F. Fauzi, "Text-based emotion detection and applications: A literature review," 6 Oct. 2022, pp. 1–9, doi: 10.1109/ICCR56254.2022.9995902.

[2] S. A. Kumar and A. Geetha, "Emotion detection from text using natural language processing and neural networks," *Int. J. Intelligent Systems and Applications in Engineering*, vol. 12, no. 14s, pp. 609–615, 2024. Available: https://ijisae.org/index.php/IJISAE/article/view/4707.

[3] D. Yohanesa, J. S. Putraa, K. Filberta, K. M. Suryaningruma, and H. A. Saputri, "Emotion detection in textual data using deep learning," *Elsevier*, vol. 227, pp. 464–473, 2023.

[4] S. K. Bharti, S. Varadhaganapathy, R. K. Gupta, P. K. Shukla, M. Bouye, S. Karanja Hingaa, and A. Mahmoud, "Text-based emotion recognition using deep learning approach," *Hindawi Computational Intelligence and Neuroscience*, vol. 2022, Art. ID 2645381, pp. 8, 2022. Available: https://doi.org/10.1155/2022/2645381.

[5] A. F. Ab. Nasir, E. Nee, C. S. Choong, A. S. Abdul Ghani, P. P. Abdul Majeed, A. Adam, and M. Furqan, "Text-based emotion prediction system using machine learning approach," *IOP Conference Series: Materials Science and Engineering*, vol. 769, no. 1, Art. 012022, 2020. Available: https://doi.org/10.1088/1757-899X/769/1/012022.

[6] M.Sireesha, Srikanth Vemuru, S.N.Tirumala Rao "Classification Model for Prediction Of Heart Disease Using Correlation Coefficient Technique" International Journal of Advanced Trends in Computer Science and Engineering, Vol. 9, No. 2, March - April 2020, Pages- 2116 – 2123.

[7] B. S. Sundar, V. Rohith, B. Suman, and K. N. Chary, "Emotion detection on text using machine learning and deep learning techniques," *Int. J. Res. Appl. Sci. Eng. Technol. (IJRASET)*, vol. 10, no. VI, pp. 1–9, June 2022. DOI: 10.22214/ijraset.2022.44293.

[8] M. M. S. Zidan, I. Elhenawy, A. R. Abas, and M. Othman, "Textual emotion detection approaches: A survey," *Future Computing and Informatics Journal*, vol. 7, no. 1, Art. 3, 2022. Available: https://doi.org/10.54623/fue.fcij.7.1.3.

[9] P. Arya and S. Jain, "Text-based emotion detection," *Int. J. Computer Engineering & Technology (IJCET)*, vol. 9, no. 3, pp. 95–104, 2018. Available: http://iaeme.com/Home/issue/IJCET?Volume=9 & Issue=3.

[10] P. Sohan Sai, "ISEAR dataset [Data set]," Kaggle, 2023. Available: https://www.kaggle.com/code/psohansai/isear/notebook.

[11] P. Nandwani and R. Verma, "A review on sentiment analysis and emotion detection from text," *Soc. Netw. Anal. Min.*, vol. 11, Art. 81, 2021. Available: https://doi.org/10.1007/s13278-021-00776-6.

[12] G. Swathi, B. Meghana Patnaik, A. Janani, K. Karthik, and J. Divya, "Emotion recognition from text using machine learning," *UGC CARE Listed (Group -I) Journal*, vol. 11, no. 12, 2022.

IEEE_Conference_Template.pdf

6% SIMILARITY INDEX	6% INTERNET SOURCES	4% PUBLICATIONS	1% STUDENT PAPERS

1	www.irjet.net Internet Source	1%
2	www.coursehero.com Internet Source	1%
3	www.mdpi.com Internet Source	1%
4	Arvind Dagur, Karan Singh, Pawan Singh Mehra, Dhirendra Kumar Shukla. "Artificial Intelligence, Blockchain, Computing and Security", CRC Press, 2023 Publication	1%
5	dokumen.pub Internet Source	1%
6	Submitted to Anatolia College Student Paper	<1%
7	aircconline.com Internet Source	<1%
8	journal.irpi.or.id Internet Source	<1%