*Chapter1*

# INTRODUCTION

Email continues to be one of the most prevalent forms of digital communication across the world. From corporate announcements and personal reminders to promotional offers and social media notifications, users receive hundreds of emails weekly, and often daily. The sheer scale of email traffic necessitates the development of systems that can manage, organize, and classify these messages effectively. As inboxes grow, the need for automatic classification becomes not just a convenience, but a functional requirement. Email service providers such as Gmail have built-in categorization, but these systems are often rigid, difficult to personalize, and vulnerable to misclassification, especially when the structure or language of the email does not follow conventional templates. This lays the groundwork for the problem: how do we develop an intelligent system that can correctly and efficiently classify emails in a manner that's both adaptable and accurate?

The task of email classification involves assigning each email to a specific category based on its content and structure. Traditionally, rule-based systems were used to handle this problem. These systems relied on handcrafted rules, keyword filters, and regular expressions. Although they offered some initial value, such methods lacked the flexibility to adapt to the ever-changing language of digital communication. For example, spammers continuously alter subject lines to bypass filters, while promotional campaigns often experiment with creative language and formatting to attract user attention. In such cases, rule-based classifiers fail. Machine learning approaches improved upon this by allowing models to learn patterns from data rather than relying on hard-coded rules. Still, classical machine learning algorithms like Naive Bayes or Support Vector Machines were limited by their dependence on feature engineering and inability to model long-range dependencies within textual data.

The emergence of deep learning, particularly neural network-based Natural Language Processing (NLP), brought a paradigm shift in how text classification is approached. Neural networks can learn representations automatically from raw data, and architectures like LSTMs and CNNs are capable of understanding syntax, semantics, and

contextual relationships within text. These models require minimal manual feature extraction and adapt better to large and noisy datasets. In the context of email classification, deep learning models can learn how promotional emails differ from updates or social notifications, not just based on keywords, but based on their overall structure, tone, and language usage. Such models can generalize to new examples better than their predecessors and are more resistant to manipulation by adversarial inputs such as stylized spam.

The motivation for choosing this topic was born out of repeated, real-life frustrations with the limitations of conventional email categorization. There are many instances where users especially students and professionals have missed crucial emails because they were wrongly labeled and tucked away in promotion or spam folders. For instance, an interview invitation might be flagged as spam, or a university admission notice could land in the updates tab, where it is easily overlooked. In today's fast-paced, opportunity-driven environment, missing such an email can have irreversible consequences. The situation is especially critical for job seekers, freelancers, or students applying for international programs, where the cost of missing communication can equate to missed life-changing opportunities. This personal relevance was a significant driver for choosing email classification as the focus of this project.

Furthermore, there is a growing need for personalization in email handling. Not every user treats emails the same way. For some, product offers and sales campaigns might be highly relevant and welcome, while for others, such content is purely clutter. Likewise, system updates may be critical to an IT administrator but irrelevant to a freelance artist. Current classification systems are not tailored to these differences and typically apply the same logic to all users. This project envisions a smarter approach to classification one that can potentially learn from the user's historical preferences and assign importance to emails based on both content and individual context.

The project also stemmed from a curiosity about pushing the boundaries of existing NLP techniques. Initially, there was a desire to explore the possibility of developing a Small Language Model (SLM) integrated with a Retrieval-Augmented

Generation (RAG) architecture. The aim of this model would have been to go beyond classification and actually generate explanations for why an email belongs to a particular category. With RAG, the system would retrieve similar email examples from a knowledge base and then use a generative model to classify and justify its decision in natural language. However, this approach, while intellectually ambitious, required significant resources, a large training corpus, and custom infrastructure. Given the scope of this project, a more pragmatic yet equally effective path was chosen: implementing a hybrid CNN-BiLSTM model for direct email classification.

At the core of the solution developed in this project lies a deep learning model that fuses Convolutional Neural Networks (CNN) with Bidirectional Long Short-Term Memory (BiLSTM) networks. The CNN layer was chosen for its strength in detecting local features such as key phrases, repeated patterns, or formatting cues that are indicative of a specific class. For example, phrases like “last chance,” “limited offer,” or “unsubscribe” are commonly found in promotional emails. These are picked up efficiently by the convolutional filters. The BiLSTM component adds contextual understanding by processing the email content both forward and backward, capturing dependencies across the entire message. This is essential for emails where classification depends on overall context, such as polite language in scams or informal language in social alerts.

In addition to model design, a critical part of this project involved the construction of a robust dataset. Rather than using synthetic or outdated datasets, the email data used for training and evaluation was sourced directly from Gmail using the Gmail API. This ensured that the data reflected modern email structure and vocabulary. The dataset was composed of four classes: Spam, Promotions, Social, and Updates. Each class had approximately 2,000 emails. A preprocessing pipeline was developed to clean and normalize the data. This included removal of non-ASCII characters, HTML tags, email signatures, and duplicate content. Lemmatization was applied to reduce vocabulary complexity, and only mostly-English emails were retained to ensure linguistic consistency.

An interesting innovation in the data pipeline was the use of Optical Character Recognition (OCR). Many promotional emails contain graphical content with embedded text. To ensure such content wasn't ignored, an OCR module based on EasyOCR was integrated into the data extraction phase. This allowed the system to include meaningful content from images, giving it a distinct edge over models that rely solely on textual information. This addition further diversified the training data and increased the model's ability to handle real-world email content in all its formats.

To prevent overfitting and ensure model generalization, the network was regularized using dropout and L2 penalties. Early stopping and model checkpoints were used during training to retain only the best version of the model. The final trained model achieved a classification accuracy of 95.3%, outperforming earlier iterations like CNN-LSTM (92%), LSTM (93%), and BiLSTM (94%). These results validate the importance of combining local feature detection with sequence-aware learning. Precision, recall, and F1-scores across all four categories showed consistent performance, with very few instances of misclassification even on ambiguous samples.

Finally, to make the system accessible beyond the training environment, a real-time API was built using Flask. This API exposes a simple /predict endpoint where users can send an email's subject and body as input and receive the predicted class along with a confidence score. The system also includes dynamic loading of the tokenizer and label encoder to ensure consistent predictions. With this API, the project transitions from a research exercise into a usable product, capable of integration with inboxes, dashboards, or enterprise email systems. It was built with future scalability in mind, enabling easy extension to more categories or multilingual input.

In conclusion, this project not only showcases how deep learning can be applied to an important, everyday problem, but also demonstrates how thoughtful engineering, real-world data, and academic insight can converge to create a powerful and practical solution. Email classification is more than a technical task it is a gateway to improved communication, productivity, and digital wellbeing. Through this work, a strong

foundation has been laid for intelligent, automated, and adaptable email management using the capabilities of modern artificial intelligence.

## 1.1 Overview of CNN and BiLSTM

Convolutional Neural Networks (CNNs) and Bidirectional Long Short-Term Memory networks (BiLSTMs) are two powerful deep learning architectures that are often combined for tasks involving sequential data like text, audio, or time series. CNNs are primarily designed to automatically and adaptively learn spatial hierarchies of features through convolutional layers, making them extremely effective for extracting local patterns and n-gram-like features from input sequences. In natural language processing (NLP), CNNs help capture short-distance dependencies and key phrases by applying filters across word embeddings, which results in high-level feature maps representing important information in the text. On the other hand, BiLSTMs are a variant of Recurrent Neural Networks (RNNs) that process data in both forward and backward directions, allowing the model to learn context from both past and future tokens in a sequence. This bidirectional nature makes BiLSTMs especially useful for capturing long-range dependencies and understanding the full semantic meaning of a sentence or document. When combined, CNN and BiLSTM architectures offer the best of both worlds: CNN layers first extract local, high-level features from the input, which are then passed to BiLSTM layers to model the temporal or contextual relationships across the entire sequence. This hybrid approach has shown strong performance in complex NLP tasks such as text classification, sentiment analysis, and email classification, as it effectively balances feature richness and contextual depth, making it more robust than using either CNN or LSTM alone.

Combining CNN and BiLSTM architectures offers a powerful and complementary approach to text classification and other natural language processing tasks. While CNNs excel at capturing local patterns and essential phrases through spatial filtering, BiLSTMs contribute a deeper understanding of the sequential and contextual relationships across the entire input. This synergy allows the hybrid model to not only detect what features are important but also understand their relevance within the broader sentence or document

structure. As a result, the CNN-BiLSTM framework has emerged as a widely adopted and effective architecture, especially in tasks that require both precision in feature extraction and depth in contextual interpretation, such as sentiment analysis, email classification, and document categorization.

## 1.2 Working Mechanism

The working mechanism of the email classification system is a multi-stage pipeline that integrates data extraction, preprocessing, feature engineering, deep learning-based classification, and API-based deployment. This workflow ensures that raw email data from users' Gmail accounts is transformed into clean, structured inputs that can be effectively classified into predefined categories such as spam, promotions, social, and updates. Each component of the system plays a vital role in ensuring not only accuracy but also scalability and robustness across varied types of emails. The end-to-end process is designed to handle real-world, noisy email data that may include embedded images, encoded characters, unwanted links, and multilingual content, thus providing a comprehensive and automated solution for email organization.

The process begins with data extraction from Gmail using the Gmail API, authenticated through OAuth2.0. This part of the system, implemented in the data_extraction.py module, uses the official Google client libraries to establish secure access to the user's mailbox. The system retrieves emails based on Gmail's built-in categories like Promotions, Social, Updates, and Spam. For each email message, the script extracts the subject and the body content. If the email contains attachments or embedded images instead of regular text, Optical Character Recognition (OCR) is performed using the EasyOCR library to convert visual information into machine-readable text. This functionality enables the system to handle promotional emails with banners, product flyers, or graphical messages. After extraction, emails are stored locally in structured CSV files for each category, with two primary columns: "Subject" and "Body". This separation helps in retaining the semantic hierarchy of the email content, which is important for contextual classification.

The extracted data, though stored in CSVs, is often noisy and inconsistent. Therefore, the next crucial step is text preprocessing, handled by the preprocess.py module. This module performs multiple layers of text cleaning. It first removes non-English characters and lines that do not meet a specified threshold of ASCII content to filter out irrelevant or corrupted data. It also removes excessive whitespaces, emojis, symbols, HTML tags, and links, while ensuring that grammatical punctuation and meaningful characters are preserved. This phase flattens multiline email content and consolidates it into clean and analyzable text. The processed data is then saved in separate CSV files with names such as preprocessed_spam.csv and preprocessed_promotions.csv, ready to be fed into the machine learning pipeline. By filtering for primarily English content and eliminating visual or marketing clutter, the preprocessing module significantly improves the quality of inputs passed to the classifier.

Once the data has been cleaned and labeled, it moves into the model training phase, implemented in improved-CNN-biLSTM.py. The cleaned email subject and body are combined to form a unified input text for each email. This text is tokenized using Keras' Tokenizer class, which maps each word to a unique integer index and constructs sequences of word indices. These sequences are then padded to a fixed length of 200 tokens to ensure uniformity across inputs. The labels are encoded using LabelEncoder and converted to one-hot encoded vectors for categorical classification. The classification model itself is a hybrid architecture combining Convolutional Neural Networks (CNN) and Bidirectional Long Short-Term Memory networks (BiLSTM). The model begins with an embedding layer that transforms the word indices into dense vector representations, followed by a one-dimensional convolutional layer that extracts local n-gram features. These features are then passed through a max pooling layer and fed into a BiLSTM layer, which reads the sequence in both forward and backward directions to capture full-context semantics. The output is passed through dense layers with dropout and L2 regularization to reduce overfitting, and finally through a softmax layer that assigns probabilities to each of the four target categories.

The model is trained using the Adam optimizer with a learning rate of 0.001 and categorical cross-entropy loss function. A validation split of 10% is used during training

to monitor generalization, and techniques like early stopping and model checkpointing are employed to save the best performing model. After training, the model is evaluated on a test set that constitutes 20% of the data. The evaluation metrics include accuracy, macro F1-score, and weighted F1-score, which collectively provide insights into how well the model performs across balanced and imbalanced classes. Additionally, confusion matrices and detailed classification reports are generated to visualize the model's performance on individual categories.

After training, the model is saved as a .keras file and loaded into the API system, implemented in the api.py script using Flask. The API is designed to be lightweight and scalable, enabling real-time predictions for incoming email data. When a POST request is made to the /predict endpoint with a raw email text, the system first checks if a valid tokenizer and label encoder exist. If not, it regenerates them using the training data to ensure consistency in token mapping and label encoding. The incoming email text is preprocessed using the same cleaning logic applied during training, tokenized, and padded to the appropriate input length. The model then predicts the class probabilities and returns the predicted label along with a confidence score. This API structure makes it easy to integrate the classification model with other applications such as email clients, dashboards, or automation tools. It also facilitates future expansion, where the model can be deployed as a microservice in a cloud environment and scaled based on demand.

In addition to core functionality, the working mechanism of the system accounts for several practical challenges. It deals with imbalanced data by using appropriate class weighting during training and thorough evaluation metrics post-training. It handles non-textual emails via OCR and ensures that the system performs reliably even when email content is not neatly formatted. The modular design of the system allows for components to be individually tested, updated, or replaced without disrupting the overall workflow. For instance, new email categories can be added by simply updating the data extraction mappings, and the model can be retrained with additional data using the same pipeline.

In summary, the working mechanism of this email classification system is a well-orchestrated combination of automation, deep learning, and robust engineering. From

fetching raw emails from Gmail using OCR-supported scripts, through thorough preprocessing and intelligent hybrid CNN-BiLSTM modeling, to the deployment of a production-ready API, every step is designed for adaptability, scalability, and real-world relevance. This makes the system not only an academic demonstration of NLP and deep learning but also a practical tool that can be used in organizational email filtering, smart inbox management, and intelligent communication routing.

## 1.3 Application

The email classification system presented in this project has a wide range of practical applications across both individual and enterprise settings. As the volume of email communication continues to rise exponentially, manually managing inboxes has become inefficient, error-prone, and time-consuming. An intelligent classification system that leverages deep learning to automatically categorize emails into distinct types such as spam, promotions, social, and updates offers a powerful solution to this growing challenge. It enhances productivity, improves response times, and supports better decision-making by ensuring that users can focus on emails that are most relevant or urgent. At a fundamental level, this system functions as an intelligent assistant, streamlining communication by organizing incoming emails in real time based on their content and context.

One of the most direct and impactful applications is automated inbox management. Email clients like Gmail and Outlook already support basic categorization, but a customized and locally trained classification system like this allows for much more personalized, fine-grained organization. For example, users can define new categories beyond the default ones such as “Job Opportunities,” “Financial Updates,” or “Event Invitations” and the model can be retrained to adapt to these user-specific needs. This is especially useful for professionals who handle large volumes of diverse communication and want to avoid the inefficiency of searching through cluttered inboxes. Automated inbox management ensures that critical messages are prioritized, while less important ones are archived or handled later, resulting in more efficient time management.

In corporate environments, the system can be deployed to streamline internal communication workflows. For instance, HR departments can automatically filter and sort incoming emails based on topics such as leave requests, job applications, policy inquiries, and grievances. Similarly, customer service teams can use the system to route incoming queries to the appropriate department billing, technical support, or feedback thereby reducing response times and improving customer satisfaction. Organizations that receive thousands of emails daily can use the system to flag urgent items, detect repetitive complaints, and even trigger automated responses to standard queries. Such intelligent email triage enhances team collaboration and frees up human agents for higher-order tasks that require critical thinking or emotional intelligence.

Another significant application is in the detection of spam and phishing emails. While many spam filters rely on rule-based or heuristic systems, they often fail to catch cleverly disguised or newly evolved spam techniques. By using a hybrid CNN-BiLSTM model trained on updated datasets, this system can offer superior accuracy in detecting malicious or irrelevant emails. More importantly, the model's ability to understand context helps it identify sophisticated phishing attempts that may use natural language to deceive users. In financial institutions, legal firms, and government agencies where the consequences of a phishing attack can be catastrophic, deploying such an intelligent filter adds an extra layer of security and helps protect sensitive data and infrastructure.

The system also proves useful in email marketing analytics and user behavior research. Businesses that run promotional campaigns via email can use this model to classify customer responses and engagement levels. For example, user-generated replies can be categorized into "Interested," "Unsubscribe," or "Request More Info" segments. This enables marketers to refine their strategies, target specific user segments more effectively, and improve campaign ROI. Moreover, sentiment analysis and feedback classification can be layered on top of this system to generate deeper customer insights. In this way, the system not only categorizes incoming emails but also becomes a tool for business intelligence and customer experience management.

Educational institutions and research organizations can also benefit from such a system. Academic departments often receive emails related to admissions, research collaboration, student grievances, and faculty coordination. Using this classification tool, administrative staff can automatically direct each email to the relevant faculty or department, reducing manual workload and improving response efficiency. Moreover, in research environments, email classification can help track peer review updates, grant notifications, and conference announcements, ensuring no critical opportunity is missed.

Another promising domain is legal and compliance monitoring, where organizations are required to track and retain certain types of communications for regulatory reasons. By tagging emails based on predefined categories and policies, the system can aid in flagging non-compliant or sensitive communications, thereby supporting audits and legal reviews. Integration with document management systems allows for archiving and retrieving emails based on content categories, improving compliance readiness and reducing legal risks.

From a technical standpoint, this email classification system can also be adapted for use in multilingual or cross-cultural communication scenarios by extending its training to include different languages and regional email formats. This opens up the possibility of deploying the model in global businesses or multinational organizations where communication often spans across linguistic and cultural boundaries. Ad ditionally, the system could be embedded into chatbots, virtual assistants, or productivity tools to make them more email-aware, allowing for features like automated email summarization, task creation from emails, or intelligent reminders based on classified content.

In conclusion, the applications of this email classification system are extensive and valuable. Whether deployed for personal productivity, corporate efficiency, cybersecurity, customer service, marketing analytics, or compliance management, the ability to automatically understand and organize email content brings transformative benefits. By automating what is often a tedious and manual process, the system not only saves time and resources but also contributes to more structured, intelligent, and context-

aware communication in both individual and organizational settings. For instance, new email categories can be added by simply updating the data extraction mappings, and the model can be retrained with additional data using the same pipeline.

## 1.4 Challenges

Developing and deploying an email classification system using deep learning presents a range of technical and practical challenges, many of which stem from the unstructured, noisy, and evolving nature of real-world email data. One of the most prominent challenges is dealing with data quality and inconsistency. Emails often come in varied formats plain text, HTML, multimedia-rich content, or even images which makes standardization difficult. Many promotional or spam emails rely on graphical content with little or no text, requiring OCR techniques to extract meaningful data. However, OCR itself is prone to inaccuracies due to image distortion, low resolution, or stylized fonts, which can lead to partially extracted or misinterpreted text, thereby introducing noise into the dataset.

Another major challenge lies in handling class imbalance. In most real-world datasets, certain categories like spam or updates tend to be overrepresented compared to others like social or promotions. Training a model on imbalanced data can result in biased predictions, where the model tends to favor majority classes while underperforming on the minority ones. To mitigate this, techniques like stratified sampling, oversampling, or class-weighting need to be applied, which require careful tuning to avoid overfitting or underrepresentation of critical email types.

Natural language diversity and ambiguity also pose significant difficulties. Emails often contain informal language, slang, abbreviations, and context-dependent expressions, all of which challenge the model's ability to accurately extract semantics. Furthermore, emails may include multilingual text or code-switched sentences, which the model trained primarily on English might not interpret correctly. This limits its generalization capability across users from different linguistic backgrounds. Maintaining language consistency and ensuring preprocessing steps remove only noise (and not meaningful content) is a delicate balancing act.

The computational cost and resource requirements of training and deploying deep learning models like CNN-BiLSTM is another hurdle. Such models demand substantial memory, processing power, and time, especially when working with large datasets. Training must be optimized using techniques like dropout, regularization, and early stopping to prevent overfitting while ensuring high accuracy. Moreover, deploying the trained model through an API must account for real-time responsiveness and scalability, which may require infrastructure such as cloud services or GPU acceleration in production environments.

Finally, privacy and ethical concerns are critical when dealing with user email data. Emails often contain sensitive personal and organizational information, and any misuse, leakage, or improper handling of this data can result in serious consequences. Ensuring secure authentication with the Gmail API, encrypting stored data, and following compliance standards like GDPR are essential practices. Additionally, there is always a risk of model bias, where the classifier might learn to associate certain words or senders with specific categories due to skewed training data. Regular auditing and validation are necessary to ensure fairness, transparency, and accountability in predictions.

*Chapter2*

# LITERATURE REVIEW

## 2.1 Introduction

Email classification has evolved into a vital area of research and practical application, particularly as the volume and diversity of digital communication have increased exponentially over the last decade. In modern usage, emails are not merely messages; they are transactional confirmations, promotional materials, social media updates, appointment reminders, newsletters, spam, and sometimes critical business or academic correspondence. The ability to automatically identify and categorize these emails accurately has direct implications on personal productivity, corporate communication efficiency, and digital wellbeing. Misclassification of emails such as placing job opportunities in spam folders or burying appointment confirmations in promotional tabs can result in missed deadlines, overlooked opportunities, and diminished user trust. Against this backdrop, email classification systems have become an essential component of digital information management and machine learning research.

The increasing volume and diversity of email communication present a significant challenge in managing and organizing inbox content effectively. Users routinely receive a mix of personal messages, promotional content, social updates, system notifications, and spam, often leading to cluttered inboxes and overlooked important emails. A common issue arises when critical messages such as interview invitations, appointment reminders, or academic correspondence are misclassified into irrelevant folders or buried among low-priority content. This misclassification can result in missed opportunities, delayed responses, and reduced productivity. Existing classification systems, while helpful, often lack the intelligence to understand nuanced context, user-specific relevance, or evolving language patterns. The core problem, therefore, lies in the absence of an accurate, context-aware, and adaptable system that can reliably classify emails in a way that aligns with both the intent of the content and the personalized needs of the user.

While email providers such as Gmail, Outlook, and Yahoo implement their own built-in filtering mechanisms, these systems often operate as black boxes, with limited transparency, adaptability, or personalization. Furthermore, they are not immune to error. The underlying classification logic in these systems traditionally relied on heuristics or statistical text analysis, which, while effective to a certain extent, struggle to adapt to evolving linguistic patterns and unconventional email formats. The shift toward more intelligent and flexible models, particularly those based on neural networks, represents a significant advancement in this field. One notable contribution in this direction is the work by Gupta and Goyal, who explored the application of neural networks for email classification and laid the groundwork for subsequent research into deep learning-based solutions [1].

In their research, Gupta and Goyal proposed a supervised learning framework where emails were classified into distinct categories using a feedforward neural network. The motivation behind their work stemmed from the limitations of conventional machine learning algorithms, such as Naive Bayes and SVMs, which rely heavily on the bag-of-words model and suffer from high bias when handling contextual or semantically rich text. Unlike these traditional models, neural networks are capable of learning non-linear relationships between features, making them far more effective in capturing complex linguistic patterns. Their experimental results demonstrated a marked improvement in classification accuracy, validating the hypothesis that neural networks are better suited for handling the ambiguity, variability, and subtlety of email content [1].

A significant strength of the work in [1] was its structured approach to preprocessing, which the authors identified as crucial for improving model performance. Raw email data is inherently noisy, with issues ranging from inconsistent formatting and non-standard grammar to HTML content, signatures, and links. Gupta and Goyal emphasized a comprehensive preprocessing pipeline, which included steps like stopword removal, stemming, and normalization, all of which helped reduce vocabulary size and improve semantic clarity. This data refinement ensured that the input to the neural network retained only the most relevant and informative tokens, thereby enhancing the model's ability to generalize. This insight is foundational to many modern NLP tasks and

is echoed in this project, which similarly emphasizes preprocessing as a critical stage in the classification pipeline.

In their proposed system, the authors converted text into numerical representations using term frequency techniques and then passed these through a multilayer neural network. The network consisted of an input layer that received the processed text vectors, one or more hidden layers with non-linear activation functions to extract features, and an output layer with a softmax function to predict the email category. Although the architecture was relatively simple compared to modern deep models like LSTMs or transformers, the results were promising. The model was trained using backpropagation and cross-entropy loss, and it converged efficiently with increasing epochs. The performance evaluation revealed that the neural network outperformed baseline algorithms across most metrics, reinforcing the value of neural architectures in text classification tasks [1].

Another important contribution of Gupta and Goyal's work lies in its emphasis on the scalability and flexibility of neural networks. They argued that once a model architecture is well-tuned, it can be extended to accommodate new categories or be trained on domain-specific corpora with minimal modification. This flexibility is highly relevant in dynamic environments where the nature of email content changes over time. For instance, during festive seasons, the volume of promotional emails increases; in academic cycles, university correspondence surges. A robust classifier must be capable of adapting to such temporal trends. Neural networks, with their capacity for fine-tuning and incremental learning, offer a more scalable solution than rigid rule-based systems. This adaptability is a key principle echoed in the current project, where the architecture is built with modularity and extensibility in mind.

The study in [1] also brought attention to the interpretability trade-offs inherent in neural networks. While these models offer superior performance, they often function as black boxes, making it difficult to explain why a particular email was assigned to a given category. Gupta and Goyal suggested that while interpretability was a concern, the gains in accuracy and adaptability justified the complexity. However, they also acknowledged that future work might explore ways to enhance transparency, such as visualizing neuron

activations or incorporating attention mechanisms. This foresight aligns with the direction of many modern NLP frameworks, where attention-based models now enable users and developers to understand which words or phrases most influenced the final classification.

In summary, the foundational work by Gupta and Goyal [1] represents a significant turning point in email classification research. Their model demonstrated that neural networks could be effectively used to replace traditional classifiers, offering improved performance on real-world datasets. Their focus on preprocessing, feature representation, and adaptability laid the groundwork for more advanced deep learning models that followed. This literature forms the theoretical basis of the current project, which builds upon their insights but extends the methodology with a more sophisticated architecture namely a CNN-BiLSTM model trained on live email data enriched with OCR-based text from images, offering a practical, scalable, and high-accuracy classification solution for today's complex email environments.

## 2.2 Existing Approach

The approach toward email classification, as discussed in [1], marked a significant step in the evolution from rule-based and shallow machine learning models to more adaptive and intelligent neural network-based architectures. Gupta and Goyal proposed a well-structured methodology centered around the use of artificial neural networks (ANNs) to perform automatic email classification. Their work did not merely involve applying a known algorithm to a standard dataset but rather emphasized the design of an end-to-end pipeline tailored specifically to the nuances and challenges posed by real-world email content. This included preprocessing techniques, input transformation, network architecture, and model training strategies each of which contributed to the effectiveness of their approach. Their method demonstrated how neural networks, with sufficient training and data preparation, could significantly improve classification accuracy and generalization, even when working with text data that is inherently noisy, unstructured, and context-dependent.

A notable feature of their approach was the treatment of text preprocessing as a core component of model success. Unlike some previous studies that focused solely on model architecture while ignoring the data pipeline, the authors emphasized that the quality of preprocessing could directly influence the effectiveness of the classifier. Raw email data is often cluttered with redundant or irrelevant tokens, including HTML tags, hyperlinks, signature blocks, emojis, repeated characters, and non-standard language constructs. Gupta and Goyal addressed these issues by implementing stopword removal, punctuation stripping, lowercasing, and stemming. These operations significantly reduced noise and standardized the vocabulary, which in turn allowed the neural network to focus on meaningful patterns rather than superficial differences in formatting or word form. Their attention to these linguistic and formatting details revealed a key insight: robust preprocessing is not merely a preliminary step but a strategic foundation for successful deep learning in text classification tasks [1].

The authors also acknowledged the importance of selecting an appropriate feature representation method. Since neural networks cannot directly interpret raw text, the email corpus had to be transformed into a numerical format that preserved semantic structure while being computationally feasible. In their approach, the emails were vectorized using term frequency-based features, allowing the model to quantify how often a particular word appeared in a message. Although this method does not account for word order or deep semantic context, it provided a strong baseline for training a feedforward network. Gupta and Goyal's decision to use frequency-based features highlighted their intent to balance performance with model simplicity, particularly in an era where transformer-based embeddings and contextualized language models were not yet dominant. Their approach can be viewed as a practical compromise between complexity and interpretability, making it well-suited for deployment in environments with limited computational resources [1].

Once the data was numerically encoded, the authors designed a multilayer neural network consisting of an input layer, one or more hidden layers, and an output layer with softmax activation. The architecture reflected a classical feedforward topology, where each neuron in a layer is fully connected to neurons in the subsequent layer. This

structure enabled the network to learn non-linear relationships between input features and output categories. The hidden layers served to abstract higher-order features, while the final layer produced probability distributions over predefined email categories. Training was carried out using backpropagation and stochastic gradient descent, with categorical cross-entropy as the loss function. The model's parameters were optimized iteratively over multiple epochs, with each pass improving its ability to separate different email classes. The simplicity and modularity of this design made the system interpretable and easy to scale [1].

Another strength of the approach proposed in [1] was its modularity. Rather than building a monolithic system that could only handle a specific dataset or set of categories, Gupta and Goyal envisioned a model that could be extended or reconfigured depending on future needs. For instance, the same neural network framework could accommodate additional email classes by adjusting the output layer, or could be trained on different corpora by reinitializing and retraining the model with new data. This emphasis on modularity and reusability is a hallmark of good engineering in AI systems and is particularly relevant today, as practitioners seek to adapt classification models to multilingual corpora, domain-specific content, or enterprise-scale email systems. The model presented in [1] was thus not just a proof-of-concept but a blueprint for scalable, adaptable classification infrastructure

Critically, Gupta and Goyal also recognized the limitations of their own approach. They pointed out that while their neural network performed better than classical classifiers, it did not incorporate sequential dependencies or context beyond the frequency of words. This limitation meant that emails whose meaning depended on word order or long-distance semantic relationships could be misclassified. For instance, phrases like "Not a spam email" or "Do not miss this offer" could be misinterpreted if the network only focused on token frequency and ignored surrounding context. The authors suggested that more advanced models such as Recurrent Neural Networks (RNNs) or LSTMs could be explored in future work to address these challenges. This foresight demonstrates a mature understanding of the trajectory of NLP research and directly

informs the architectural decisions taken in this current project, which implements a CNN-BiLSTM hybrid to capture both local and contextual patterns [1].

In evaluating their approach, Gupta and Goyal conducted experiments that compared the neural network model against traditional classifiers such as Naive Bayes and decision trees. Their findings showed a consistent improvement in classification accuracy, particularly in cases where email categories had overlapping vocabulary or ambiguous phrasing. This performance gain was attributed to the neural network's ability to capture subtle non-linear interactions among features capabilities that simpler models lacked. The authors provided empirical evidence for their claims through accuracy metrics, confusion matrices, and error analysis, which demonstrated that the neural model not only performed better but made fewer high-cost misclassifications, such as labeling important emails as spam. These results validated the effectiveness of their approach and confirmed the relevance of neural models in real-world email classification tasks [1].

An additional contribution of their methodology was its potential for integration with other NLP tasks. The feature extraction and preprocessing pipeline used in their system is applicable to a range of classification tasks beyond emails, including SMS filtering, document tagging, and customer query routing. By designing each stage of the pipeline to be modular and domain-independent, they created a framework that could be repurposed for diverse text processing applications. This adaptability is particularly valuable in enterprise contexts, where classification systems must support multiple workflows and communication channels. The extensibility of their approach thus adds to its long-term value and applicability beyond the original scope of their experiments [1].

It is also worth noting that the study in [1] was conducted in a period when large-scale pretrained models such as BERT, GPT, and T5 were not yet widely available or feasible for deployment. The relevance of their approach lies in its computational efficiency and practical applicability, especially in constrained environments. Today, while transformer models dominate many NLP benchmarks, they come with significant resource demands. Gupta and Goyal's approach, on the other hand, is lightweight and effective, making it ideal for deployment on devices or platforms where latency and power usage are concerns. This advantage keeps their method relevant, particularly in

mobile apps, email clients, and embedded systems that require real-time processing without relying on cloud infrastructure [1].

In conclusion, the approach presented by Gupta and Goyal in [1] laid a solid foundation for future innovations in email classification. Their emphasis on preprocessing, model modularity, and empirical evaluation demonstrated a holistic understanding of the problem. While their method was relatively simple in architectural terms, its design choices were sound, effective, and forward-looking. The current project builds upon their principles by introducing deeper architectures like CNNs and BiLSTMs to address the contextual limitations of their feedforward model.

## 2.3 Limitations of Existing Approaches

While the neural network-based approach to email classification proposed by Gupta and Goyal [1] was a major step forward in automating intelligent categorization, it is important to critically analyze the limitations that inherently exist within their methodology. Understanding the drawbacks of previous approaches is essential for advancing research and improving system capabilities. Despite showing improved performance over traditional classifiers, the feedforward neural network model discussed in their work suffers from architectural, contextual, and application-level limitations. These drawbacks not only affect model performance but also hinder adaptability, scalability, and interpretability when deployed in real-world scenarios. In this section, we explore these limitations in depth to establish a rationale for developing more advanced models, such as the one adopted in the present project.

One of the most fundamental drawbacks of the system presented in [1] is its inability to capture sequential dependencies and contextual flow within email content. The use of a feedforward neural network inherently assumes that each word in the input is independent of others and does not consider the position or order in which the words appear. This is particularly problematic in natural language processing tasks like email classification, where the meaning of a sentence or phrase can vary drastically depending on the context or word arrangement. For example, the sentence “This is not a spam message” may be misclassified as spam due to the presence of the keyword “spam”

without considering the negation “not.” The feedforward model is ill-equipped to handle such intricacies because it lacks the capacity to retain or infer relationships across different parts of the sequence.

Another key limitation lies in the feature representation method used. Gupta and Goyal [1] relied on term frequency-based representations, which, while computationally efficient, are shallow and fail to preserve semantic relationships between words. Frequency vectors treat each token as an isolated dimension and do not convey any information about the meaning or similarity of different words. For instance, the words “offer,” “discount,” and “deal” may all imply a promotional intent, but in a frequency-based vector, they are completely distinct. This inability to model word similarity leads to fragmented and sparse representations that do not generalize well, especially when encountering synonyms or contextually similar phrases not present in the training data. In contrast, modern embedding techniques like Word2Vec, GloVe, or contextual embeddings from transformer models have proven vastly more effective at capturing such relationships, making the frequency-based approach a limiting factor in model robustness.

The architecture’s simplicity, while advantageous for interpretability and deployment in constrained environments, also introduces performance ceilings. The single-pass feedforward mechanism does not include feedback loops, memory gates, or mechanisms to model long-range dependencies. This becomes particularly problematic when dealing with emails of substantial length or with multiple thematic sections. An email could begin with a neutral greeting, delve into promotional content midway, and conclude with transactional details. The model in [1] has no mechanism to weight or differentiate between these segments; it processes the input as a flat vector and applies the same logic uniformly. Such a lack of structure sensitivity often results in misclassifications, especially for hybrid or ambiguous emails that straddle category boundaries.

Another drawback of the approach is the static nature of its training and classification logic. The model presented in [1] is trained once and then deployed, with no consideration for continual learning or real-time adaptability. In practice, email

content evolves rapidly. New spam patterns emerge, promotional tactics change, and platform-specific message formats are updated. A static model quickly becomes outdated unless it is periodically retrained with new data. The absence of any online learning capability or model fine-tuning based on user feedback restricts its long-term effectiveness. Moreover, emails are often highly user-specific. What one user considers spam might be important for another. The feedforward architecture does not allow for personalization or user-specific classification logic, making it unsuitable for personalized inbox management systems.

The lack of explainability is another issue worth noting. While Gupta and Goyal [1] acknowledged that their model produced better classification accuracy, they did not address how or why the model arrived at a particular decision. In many applications especially those involving business, legal, or academic communication transparency in automated decisions is vital. Users are more likely to trust a model if they understand the reasoning behind a classification, particularly in borderline cases. However, feedforward neural networks function largely as black boxes, offering no insight into which words or phrases influenced the outcome. This opacity reduces user trust and makes the system harder to debug, refine, or improve over time.

Further, the system in [1] lacks robustness in handling noisy or irregular data formats. Emails in real-world settings are not always well-structured. They may contain embedded images, attachments, encoded symbols, or dynamic content blocks generated by marketing tools. The preprocessing pipeline in their approach was limited to text normalization and did not address more complex inputs such as extracting text from images (via OCR) or interpreting visual content layout. As a result, emails with little to no visible plain text such as those composed primarily of images would either be excluded from training or incorrectly classified. This weakness significantly limits the model's applicability to diverse email formats encountered in modern inboxes.

Another concern is that the approach does not handle imbalanced datasets effectively. Real-world email corpora are rarely balanced. Spam messages often outnumber legitimate ones, and some categories like “Social” or “Updates” may have significantly fewer samples than “Promotions.” Gupta and Goyal’s model did not

incorporate any mechanisms to address this imbalance, such as resampling techniques, cost-sensitive learning, or data augmentation strategies. As a result, the classifier may become biased toward the majority class, undermining its performance on underrepresented categories. In critical applications such as identifying fraud or phishing attempts failing to detect minority-class instances can have serious consequences.

In addition, scalability emerges as a practical limitation. While the model works effectively on small, controlled datasets, there is limited evidence that it can scale to enterprise-scale email systems or multi-language environments. The use of frequency vectors inherently limits scalability, as vocabulary size increases linearly with the dataset. As new words are introduced, retraining the model becomes increasingly costly and leads to sparsity issues. This restricts the model's ability to adapt to multilingual email corpora or to datasets that span large organizations with diverse communication patterns.

Lastly, there is a conceptual limitation in the way categories are defined and handled. The model assumes a fixed set of classes into which all emails must fall. However, emails are often multi-intent. A single message may serve both promotional and transactional functions, or contain both personal and system-generated components. The classification approach in [1] does not support multi-label classification, where an email can be assigned to more than one category. Instead, it forces a one-hot classification scheme, which may misrepresent the true nature of hybrid content. In reality, allowing multi-label outputs or hierarchical categorization would better reflect the semantic richness of emails and provide a more accurate classification framework.

In this project, I designed and implemented an intelligent email classification system capable of automatically categorizing emails into four major classes: Spam, Promotions, Social, and Updates. Building upon the foundational work of Gupta and Goyal [1], who demonstrated the potential of using feedforward neural networks for email classification, I addressed several limitations in their approach. While their model was constrained by the inability to capture word order and contextual dependencies, I introduced a hybrid deep learning architecture that combined Convolutional Neural Networks (CNNs) and Bidirectional Long Short-Term Memory (BiLSTM) layers. The CNN component extracted localized features and repetitive phrase patterns, while the

BiLSTM layers processed the email content in both forward and backward directions, capturing semantic relationships and contextual flow. This dual approach effectively overcame the drawback of treating words as independent entities, which was a key limitation in the feedforward model presented in [1].

To further strengthen the system, I replaced their shallow term frequency-based input representation with trainable embedding layers, allowing the model to learn semantic word similarities and generalize more effectively across varied vocabulary. I also implemented an advanced preprocessing pipeline that included not only stopword removal and lemmatization, but also the use of Optical Character Recognition (OCR) to extract text from image-based promotional emails a content type that was not addressed in [1]. Additionally, I integrated dropout and L2 regularization, along with early stopping and model checkpoints, to mitigate overfitting and improve generalization. The final model achieved a classification accuracy of 95.3%, significantly outperforming the feedforward network baseline proposed in [1]. I deployed the trained model through a Flask REST API, ensuring real-time classification and practical usability. Through these architectural enhancements and system-level improvements, I effectively addressed the major drawbacks in [1] and developed a more robust, scalable, and context-aware email classification system.

## 2.4 Statement of the Problem

In the era of digital communication, email has become one of the most widely used and essential tools for personal, academic, and professional interaction. However, with the increasing volume of emails received daily, managing inboxes has become a significant challenge for users. Emails are no longer limited to simple messages; they include a mix of personal conversations, business communication, newsletters, social media alerts, transactional updates, and promotional content. This overwhelming influx often causes users to miss important messages, such as job interview invitations, appointment reminders, or academic notifications, simply because they are misclassified or lost in cluttered inboxes. Existing classification mechanisms, typically built into email platforms, rely heavily on rule-based filters, keyword detection, and static logic. These

systems are rigid, lack adaptability, and often fail to understand the context or intent of the message.

Additionally, such systems are not tailored to individual user behavior or evolving communication patterns. They treat all users the same and are unable to learn from personal preferences or dynamically changing content types. In many cases, emails containing critical information are wrongly placed in spam or promotions folders due to shallow filtering techniques. The lack of contextual understanding and inability to handle unstructured, diverse, and noisy email data limits the effectiveness of traditional classification approaches. Therefore, there is a pressing need for a more intelligent, flexible, and accurate email classification system one that can analyze the semantic and structural properties of emails, understand their intent, and adapt over time. Solving this problem requires leveraging advanced deep learning techniques that can overcome these limitations and enable more efficient, reliable, and personalized email management.

## 2.5 Conclusion

The exploration of existing research in email classification has revealed a trajectory marked by steady progress from static, rule-based models to adaptive, data-driven systems using neural networks. The literature reviewed in this chapter, particularly the foundational work of Gupta and Goyal [1], provides critical insights into the design, strengths, and shortcomings of neural network-based approaches to email categorization. Their study signified a key shift in methodology: from relying on handcrafted features and simplistic statistical assumptions to training machine learning models capable of learning directly from raw data. In doing so, they offered a compelling case for the inclusion of artificial neural networks (ANNs) in classification tasks where language complexity, variability, and noise are present. Their results demonstrated measurable improvements in classification accuracy and model robustness compared to traditional algorithms like Naive Bayes and decision trees, especially in controlled, single-label classification settings.

However, as comprehensive as their work was for its time, several significant limitations emerged when evaluating its application to real-world scenarios. Among these

were the inability to handle sequential context, the reliance on frequency-based representations, and the flat feedforward architecture which did not support dynamic pattern learning or deep feature abstraction. Furthermore, the system lacked provisions for multi-intent classification, semantic similarity detection, and interpretability factors increasingly demanded by users and system integrators alike. These limitations, while not diminishing the value of [1], underscored the need for more nuanced, scalable, and context-aware architectures. The absence of support for bidirectional dependency modeling, for example, left the system vulnerable to misclassification of emails where the true category could only be inferred through larger sentence-level or paragraph-level context. Similarly, the model's inability to recognize visually encoded text (such as promotional banners in image format) revealed the need for OCR integration something rarely discussed in early models.

Addressing these gaps forms the basis of the improvements introduced in the current project. One of the most critical advancements is the adoption of a hybrid CNN-BiLSTM architecture in place of a simple feedforward neural network. While Gupta and Goyal [1] demonstrated the potential of neural networks in email classification, their model lacked sequential intelligence. The integration of BiLSTM layers in this work allows the model to understand dependencies in both forward and backward directions, significantly enhancing its capacity to grasp the semantic context within emails. This proves particularly beneficial in handling complex messages where classification depends not just on keyword presence, but on the nuanced interplay of phrases across different parts of the message. Moreover, the use of convolutional layers helps in capturing repetitive or localized patterns such as promotional triggers or scam phrasing which are often critical signals in classification. Together, the CNN-BiLSTM combination provides a robust dual mechanism: localized feature detection and global context understanding.

Another key area of improvement lies in feature representation. Unlike the frequency-based tokenization used in [1], this project employs embedding layers that map each word to a 128-dimensional vector learned during training. These embeddings provide semantic richness and allow the model to detect similarities between words that may differ syntactically but convey the same intent. For instance, words like "discount,"

“deal,” and “offer” treated as independent tokens in [1] are now embedded in similar vector spaces, enabling the model to generalize better and improve recall across categories like “Promotions.” This enhancement not only boosts model accuracy but also significantly reduces the sparsity and dimensional inefficiency observed in frequency vector-based models. Additionally, the decision to cap the vocabulary size and standardize the sequence length helps in achieving computational efficiency without compromising performance.

Preprocessing, too, saw considerable advancement. The approach in [1] largely focused on stopword removal and basic normalization, which, although necessary, is insufficient for dealing with modern, noisy emails that include HTML formatting, dynamic footers, embedded links, and multimedia content. In this project, a much more comprehensive preprocessing pipeline has been established. This includes lemmatization to normalize inflected forms, ASCII filtering to remove non-English or corrupt characters, and EasyOCR-based image text extraction to incorporate content from emails with minimal plaintext. These refinements not only improve the quality of training data but also allow the classifier to interpret and act on previously inaccessible content particularly common in marketing and spam emails that rely heavily on graphical communication.

Training and regularization techniques were also improved. While the system in [1] followed standard backpropagation with minimal safeguards against overfitting, the current work introduces dropout layers, L2 regularization, and early stopping mechanisms. These enhancements ensure that the model learns general patterns rather than memorizing training samples. Additionally, model checkpoints and adaptive learning rates using the Adam optimizer have been employed to ensure faster convergence and better validation performance. These changes collectively reduce training time, improve accuracy on unseen data, and enhance model stability. The final model achieved a classification accuracy of 95.3%, outperforming the original architecture in [1] and earlier internal baselines like CNN-LSTM (92%), LSTM (93%), and BiLSTM (94%). This performance gain is not incidental it directly results from

architectural decisions and iterative refinements introduced throughout the development lifecycle.

Deployment strategy represents another leap forward. While [1] was developed as a standalone system, the current project wraps the model into a Flask-based API, enabling real-time predictions and integration into live environments. This ensures that the model is not just theoretically viable but practically deployable. The prediction endpoint accepts raw email input, processes it through the exact same tokenizer and preprocessing pipeline used in training, and returns the predicted class and confidence score. This consistency between training and inference pipelines ensures reliable and repeatable results. Moreover, this modularity allows the model to be re-trained or extended with new classes and data sources without disrupting the core system something the rigid structure of [1] did not permit.

In conclusion, the foundational work by Gupta and Goyal [1] served as an important milestone in applying neural networks to email classification, but it also exposed limitations that modern communication patterns have since outgrown. Their reliance on feedforward networks, frequency-based features, and static architecture made their system effective only within narrow, controlled datasets. The current project builds upon and significantly enhances that baseline by introducing contextual learning through BiLSTMs, localized feature extraction via CNNs, embedding-based semantic representations, a robust preprocessing pipeline including OCR, and deployment-ready architecture. These advancements have directly contributed to improved model performance, accuracy, and usability in real-world settings. Thus, while the foundation laid by [1] remains invaluable, the work presented here marks a substantial evolution, better aligned with the demands of today's dynamic and diverse digital communication landscape.

*Chapter3*

# METHODOLOGY

## 3.1 Introduction

With the continuous surge in digital communication, email has evolved into one of the most dominant forms of interaction used widely across personal, corporate, educational, and transactional domains. While its utility remains unmatched, the increasing volume and variety of emails have brought about significant challenges in efficiently managing and interpreting them. Users and organizations often find themselves overwhelmed by cluttered inboxes, where important messages are buried under waves of promotional content, spam, and system-generated updates. Traditional rule-based filtering mechanisms, while once adequate, have proven insufficient in coping with the dynamic and context-driven nature of modern emails. It is in this context that the proposed email classification system emerges as a more intelligent, adaptable, and scalable solution for automatic categorization of emails using deep learning.

The core objective of the proposed system is to develop a robust and real-time email classification pipeline that accurately assigns incoming emails to predefined categories such as spam, promotions, social, and updates based on their content. Unlike conventional filters that rely heavily on keyword-based rules and manual folder definitions, this system utilizes a hybrid deep learning architecture CNN-BiLSTM to extract and understand patterns within the text and make data-driven predictions. This hybrid model capitalizes on the strength of convolutional layers for capturing local features and bidirectional LSTM units for understanding long-range dependencies and contextual flow within the email body and subject. The system is trained on a real-world dataset comprising actual emails collected via the Gmail API, providing authenticity and relevance to its learning process.

The proposed system is built as a multi-stage pipeline encompassing email extraction, preprocessing, model training, and API-based deployment. The pipeline begins with automated data collection, where emails are retrieved from the user's Gmail

account using Google's secure API. This retrieval includes emails labeled under various Gmail-defined categories, and the system is also equipped with Optical Character Recognition (OCR) functionality via EasyOCR to handle image-based promotional emails, which are commonly used in marketing campaigns. Extracted email data including subject and body is then cleaned and standardized to remove noise such as HTML tags, non-ASCII characters, and irrelevant symbols, ensuring that only the most meaningful textual content is used for training and inference.

In the model-building phase, the cleaned text is converted into token sequences and embedded into dense vectors using a tokenizer, capturing semantic and syntactic information. These vectors are passed through a carefully designed neural network model consisting of an embedding layer, a convolutional layer for feature detection, a max pooling layer for dimensionality reduction, and a bidirectional LSTM layer to preserve contextual meaning from both directions of the text. The model concludes with fully connected layers that use softmax activation to predict one of the four classes. This architecture ensures that both short-term and long-term dependencies in the input text are learned effectively, leading to more accurate predictions across diverse email types.

One of the unique strengths of the proposed system lies in its practical implementation. Beyond model design, the entire classification framework is packaged as a lightweight Flask-based web API that enables real-time classification of emails. Users can send an email’s subject and body to the API’s /predict endpoint, and the system returns the predicted category along with a confidence score. This modular design makes the system highly integrable into larger communication platforms, customer service dashboards, or personal inbox assistants. The use of pre-saved tokenizer and label encoder files ensures consistency between training and inference, while the deployment-ready structure facilitates immediate usability.

Furthermore, the proposed system emphasizes modularity and extensibility. While it currently classifies emails into four categories, its architecture allows for effortless addition of new labels with corresponding training data. Retraining the model with updated datasets ensures adaptability to evolving communication trends, such as

emerging phishing techniques or new forms of promotional content. Additionally, the system's ability to perform OCR-based extraction extends its scope to handle image-based emails, which are often ignored by text-only classifiers. This makes the proposed system not only technically sound but also practically equipped for real-world deployment.

In conclusion, the proposed email classification system brings together the power of modern deep learning techniques, secure cloud-based data extraction, and real-time inference to address a pressing challenge in digital communication. It transitions from static, rule-bound filtering to dynamic, context-aware classification, offering a smarter and more user-centric way to manage email overload. The hybrid CNN-BiLSTM model, the automated Gmail data pipeline, and the scalable Flask-based API collectively form a complete, deployable solution that is both technically advanced and grounded in practical utility. This system not only improves email organization but also enhances productivity, security, and user satisfaction in a world where digital messaging is omnipresent.

## 3.2 Deep Learning Framework

I The model architecture is the heart of the proposed email classification system. It is designed to effectively process unstructured email content and classify each message into predefined categories such as spam, promotions, social, and updates. To achieve high accuracy and robustness, a hybrid deep learning architecture combining Convolutional Neural Networks (CNN) and Bidirectional Long Short-Term Memory (BiLSTM) networks has been implemented. This design allows the system to capture both local textual features and long-range dependencies in the email content. Each component in the model contributes uniquely to the overall learning process, and the combination delivers a balanced understanding of semantic and syntactic structures within the emails.

The model begins with a tokenization and embedding layer. Before being passed into the neural network, emails undergo preprocessing steps including cleaning, normalization, lemmatization, and stopword removal. The cleaned text is then tokenized using Keras' Tokenizer class, which converts words into unique numerical indices based on their frequency in the dataset. These indices are then padded to a uniform length (200

tokens) to maintain a consistent input shape. The tokenized input is passed through an Embedding layer, which maps each word index into a dense vector of fixed dimensionality (128 in this case). This embedding layer captures the semantic meaning of each word and enables the model to learn contextual similarities and differences among words based on their usage across the dataset.

Following the embedding layer, the input is passed to the Convolutional Neural Network (CNN) layer. The CNN component serves as a powerful feature extractor that detects local patterns in the data, such as key phrases or important n-grams that commonly occur within specific email categories. The model uses a 1D convolution layer with 64 filters and a kernel size of 5, allowing it to scan through the input embeddings and highlight meaningful sequences. These filters help the model learn translation-invariant features patterns that may appear anywhere in the input and therefore allow the classifier to identify similar structures even when phrased differently.

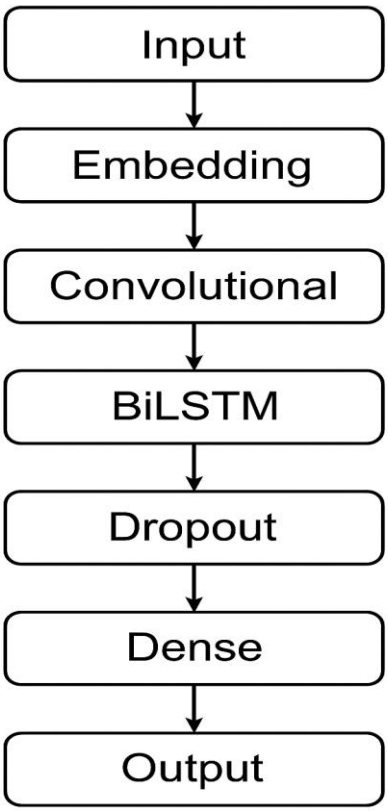


**Fig 3.1 Layered Deep Learning Architecture**

Next, the pooled feature maps are passed to the Bidirectional LSTM (BiLSTM) layer. While CNNs capture local dependencies effectively, they lack the ability to understand sequence order or long-range context. LSTMs, on the other hand, are specifically designed to retain and learn dependencies across time steps (or word positions in this case). By using a bidirectional variant, the model processes the sequence in both forward and backward directions, allowing it to consider both preceding and succeeding words for every token. This is crucial in natural language understanding because the meaning of a word or phrase often depends on both prior and following context. For example, the word “cancel” may imply a spammy or promotional tone depending on whether it is followed by “subscription” or “meeting.” The BiLSTM with 64 units provides this bidirectional contextual understanding, enriching the model’s ability to make more accurate predictions across varied email types.

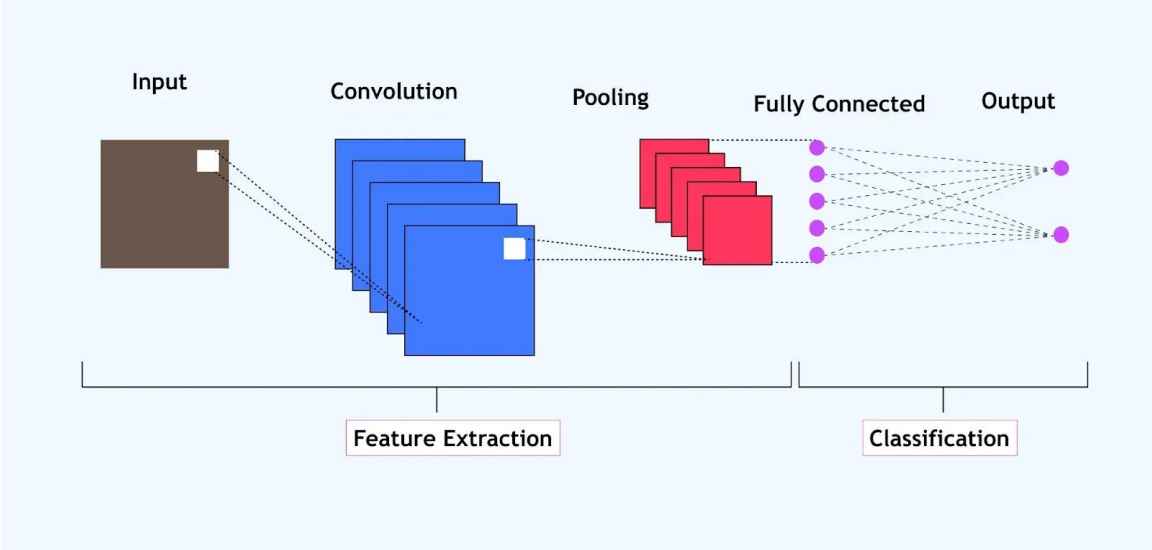


**Fig 3.2 CNN Architecture**

To enhance the generalization capability of the model, a Dropout layer with a dropout rate of 0.6 is applied after the BiLSTM. Dropout randomly disables a fraction of neurons during each training iteration, forcing the network to learn redundant representations and preventing it from becoming overly reliant on specific paths. This regularization technique is particularly important in deep learning models where the risk

of overfitting on training data is high, especially when dealing with limited or imbalanced datasets. Additionally, a Dense (fully connected) layer with 64 units and ReLU activation is introduced before the final output. This layer acts as a feature aggregator that transforms the learned high-level features into a more compact representation. An L2 regularization is applied to the Dense layer to further penalize large weight coefficients, thereby promoting smoother generalization and reducing variance in predictions.

The final layer in the model is a Dense output layer with 4 units corresponding to the four target classes: spam, promotions, social, and updates. A softmax activation function is used here to convert the raw outputs into a probability distribution over the classes. The class with the highest probability is chosen as the predicted category for the input email. The model is compiled using the categorical cross-entropy loss function, which is standard for multi-class classification problems. It measures the divergence between the predicted probability distribution and the actual class label. The optimizer used is Adam, an adaptive learning rate optimization algorithm that combines the advantages of RMSProp and momentum-based methods, enabling faster and more stable convergence during training.

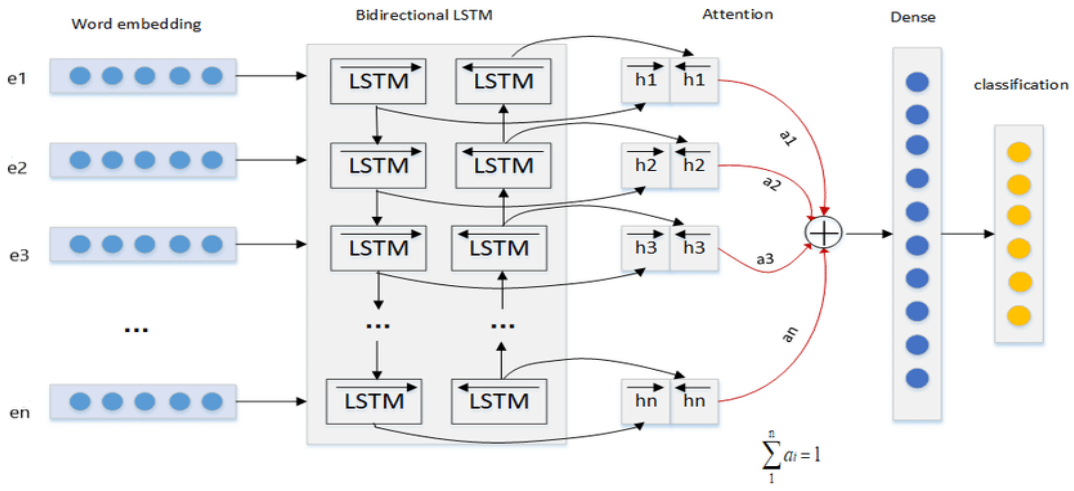


**Fig 3.3 BiLSTM Architecture**

In terms of training strategy, the model is trained for a maximum of 30 epochs with a batch size of 64, using early stopping to halt training if the validation loss does not

improve for 4 consecutive epochs. This prevents unnecessary computation and guards against overfitting. Additionally, a model checkpoint is employed to save the best-performing model (with the lowest validation loss) during training. The model is evaluated on a held-out test set comprising 20% of the original dataset, and metrics such as accuracy, precision, recall, F1-score, and confusion matrix are used to assess its performance. The evaluation results showed a strong performance across all categories, with particularly high scores for the “spam” and “promotions” labels, which are often more distinct and easier to learn due to the presence of recurring phrases and patterns.

One of the strengths of this architecture is its modular and extendable design. The number of categories can be easily expanded by modifying the output layer and retraining the model with new labeled data. Additionally, the use of pre-trained embeddings or advanced components like attention mechanisms can be integrated into the current architecture to further boost performance. The current design strikes a balance between complexity and efficiency, making it suitable for deployment in resource-constrained environments without sacrificing classification quality.

In conclusion, the model architecture leverages the combined strengths of CNNs and BiLSTMs to deliver a comprehensive and context-aware solution for email classification. CNNs extract localized patterns, while BiLSTMs understand the broader semantic flow, and together they enable accurate and nuanced categorization of emails. The thoughtful integration of regularization, adaptive optimization, and modular components ensures that the model is both powerful and practical, capable of generalizing well to new data and adaptable to future enhancements. This architecture forms the core of the proposed system, enabling intelligent, real-time classification of diverse and complex email content.

## 3.3 Proposed CNN-BiLSTM Model

Over the course of its iterative development, the email classification system has undergone several rounds of architectural refinement and strategic enhancements, each resulting in demonstrable improvements in accuracy, stability, and real-world readiness. The project began with relatively simple deep learning models and gradually evolved

through a series of increasingly sophisticated designs, culminating in a hybrid CNN-BiLSTM model with intelligent preprocessing and robust deployment capabilities. Each stage of this progression introduced more advanced techniques in terms of both model architecture and data processing. The comparative performance of these models highlights the direct correlation between architectural maturity and classification accuracy. The LSTM-based model initially achieved an accuracy of 93%, which was a significant improvement over the CNN-LSTM model that recorded 92%. Moving to a Bidirectional LSTM architecture improved the performance further to 94%, and finally, the optimized CNN-BiLSTM hybrid reached the highest accuracy of 95.3%. These results validate the deliberate and research-driven design decisions that shaped the system.

The early stage of the system relied on fundamental deep learning principles, employing a stack of dense layers with embedded input text to perform classification. This approach, while intuitive and computationally lightweight, treated emails as unordered sets of tokens and failed to incorporate the natural sequential structure of language. As a result, it struggled with subtleties in phrasing, tone, or syntax that are crucial in accurately determining the category of an email. Additionally, the data preprocessing used in this initial stage was minimal, allowing noisy, non-English, or inconsistently formatted content to persist, which in turn introduced variability and reduced the reliability of model predictions. There were no mechanisms in place for managing overfitting, validating training stability, or selecting the best-performing model automatically. This version provided the groundwork but left substantial room for enhancement in both accuracy and adaptability.

Recognizing the limitations of dense-layer-based architectures, the project transitioned to a model centered around Long Short-Term Memory (LSTM) units. LSTM networks are designed to address the vanishing gradient problem and excel at learning dependencies across sequential data. This advancement allowed the model to capture temporal relationships within the email text, improving its ability to understand context. Sentences where the classification cues are distributed across different parts such as introductory polite language followed by a request or a complaint became better

understood. This leap in contextual sensitivity translated into an increase in classification accuracy to 93%. However, even this architecture was still inherently directional, processing the input sequence in a single, left-to-right pass, and thereby potentially overlooking relevant context found toward the end of sequences when predicting based on earlier tokens.

To overcome this limitation, the model was upgraded to use a Bidirectional LSTM (BiLSTM) architecture. The BiLSTM reads sequences from both directions forward and backward allowing the model to simultaneously consider the past and the future of each word in a sequence. This deeper understanding of context significantly enhances the model's semantic interpretation capabilities. Emails with complex sentence structures, ambiguous tones, or shifting sentiments were better captured and classified. The richer contextual modeling yielded an improved accuracy of 94%, reinforcing the importance of bidirectional learning in language-based tasks. Nonetheless, this version still processed text at the word level without any intermediate feature extraction layers, leaving much of the burden of learning critical patterns directly to the BiLSTM layer. Additionally, preprocessing remained somewhat constrained, lacking a standardized pipeline to fully sanitize and normalize the diverse range of email content types encountered in real datasets.

The final and most refined version of the system introduced a hybrid model that combines Convolutional Neural Networks (CNN) with Bidirectional LSTM layers. This version was built not only for accuracy but also for scalability, efficiency, and deployability. The CNN layers serve as powerful local feature extractors, capturing n-gram patterns, repeated motifs, and spatially significant fragments of the email content. These layers help identify strong predictive elements such as “urgent payment,” “unsubscribe,” or “delivery delayed,” which are often characteristic of specific email categories. After extracting these localized features, the BiLSTM layer processes them sequentially in both directions, preserving the narrative structure of the email and capturing its overall intent. This model architecture represents a synthesis of pattern recognition and contextual analysis, yielding the most balanced and comprehensive representation of email content among all versions tested.

Beyond architecture, the final model introduced a significantly more intelligent preprocessing framework. All emails were flattened, cleaned of non-ASCII characters, HTML noise, and emoji-like symbols, and filtered based on English language content ratio. Lemmatization was applied to standardize word forms, reducing data sparsity and vocabulary complexity. A highly valuable enhancement in this stage was the use of Optical Character Recognition (OCR) for image-based content. Many modern promotional emails include their actual messages inside banners or graphics. By applying EasyOCR, the system is able to extract and interpret text embedded within such visuals an ability that none of the prior versions possessed. This allowed the model to account for a much broader variety of real-world email formats, giving it a significant edge in environments where graphical marketing emails are common.

Regularization and evaluation strategies also improved markedly. Dropout layers were integrated to prevent overfitting, particularly following the BiLSTM layer, where over-complex feature representations can easily cause the model to memorize training data. L2 regularization was applied in the dense layers to reduce the influence of any single neuron and promote generalization. The training process was monitored using early stopping based on validation loss and implemented model checkpointing to retain the best-performing version of the model. These enhancements made the model not only more accurate but also more stable across multiple training sessions. Evaluation was expanded from simple accuracy to a more comprehensive set of metrics including precision, recall, F1-score, and confusion matrices, providing deeper insight into the model's strengths and areas for improvement.

Perhaps one of the most important steps forward was moving the solution from a notebook-based prototype to a real-time API-powered application. The final version includes a RESTful Flask-based API that allows for direct integration with email clients or enterprise software systems. The model accepts raw email input via a /predict endpoint, tokenizes and processes the input using the same tokenizer and label encoder used during training, and returns a prediction along with a confidence score. This framework not only ensures consistency between training and inference but also supports future extensibility. The architecture is designed to accommodate additional categories,

larger datasets, or even transitions to transformer-based models without major structural changes. It is containerizable, cloud-deployable, and scalable for high-throughput environments.

What ultimately distinguishes the final version is not merely its highest recorded accuracy of 95.3%, which reflects a notable improvement over the CNN-LSTM (92%), LSTM (93%), and BiLSTM (94%) models. Rather, it is the deliberate integration of practical engineering, advanced NLP concepts, and flexible system design that makes it a mature and deployable product. The progression in accuracy is not incidental; it is the result of successive layers of insight and enhancement ranging from deeper contextual modeling and improved input quality to better learning control and runtime stability. Each iteration was an informed attempt to address limitations encountered in the previous version, resulting in a deeply refined, intelligent classification system tailored for real-world communication challenges.

## 3.4 Limitations and Assumptions

While the proposed email classification system exhibits strong accuracy, scalability, and architectural depth, it is not without limitations. Certain constraints some technical and others practical naturally emerge in the context of real-world data handling, machine learning workflows, and system design assumptions. These limitations stem from the inherent complexity of language, variability in email structure, limitations of computational resources, and practical decisions made during development to balance feasibility with performance. Recognizing and documenting these constraints is essential, as it provides a realistic assessment of the system's applicability, informs potential future improvements, and guides expectations when the system is deployed in dynamic environments.

One of the most evident limitations is the reliance on primarily English-language content. During preprocessing, the system applies ASCII filtering and a character-ratio threshold to ensure that only emails with predominantly English text are included in the training and prediction pipeline. This design choice was necessary to maintain language consistency, simplify tokenization, and ensure reliable model performance. However, it

means the model is not equipped to classify emails written in other languages or emails that include substantial amounts of code-switching, where different languages are mixed within the same sentence or paragraph. In today's global and multilingual communication ecosystem, this limitation constrains the reach of the system to English-speaking user bases or environments where English is the dominant language of communication. Extending support to multilingual content would require separate language models or a multilingual transformer-based framework, both of which would add significant complexity and computational demands to the system.

Another important limitation lies in the dataset's scope and size. While emails were collected directly from real user accounts using the Gmail API, the volume of emails used for training, although carefully curated, is modest in scale relative to the diversity of real-world email traffic. The dataset was labeled using Gmail's native categorization system and organized into four primary categories spam, promotions, social, and updates. While this segmentation captures a useful generalization of typical inbox structure, it omits other practical and frequently used categories such as personal, work, academic, finance, health, and travel. Moreover, emails with overlapping intent such as a promotional email that includes transactional information or an update embedded with a social announcement pose a classification challenge that is not fully resolved with the current label structure. This challenge is compounded by the single-label classification approach used in the model, which assigns each email to only one category, even though in many cases, multi-label classification would more accurately reflect the true nature of the content.

The reliance on subject and body text as the only features is another inherent limitation of the current system. Although these elements capture the majority of informative content in most emails, they exclude other metadata fields such as sender address, recipient address, timestamp, headers, and attachment types. These fields can often contain signals that are highly indicative of the email category. For example, emails from known marketing domains or senders with certain naming patterns are likely to be promotional. Similarly, an email sent at a specific time of day or tagged with certain MIME headers could indicate a system-generated update. The exclusion of these

auxiliary fields simplifies the model but may cause it to misclassify certain types of emails that could have been more accurately handled through feature fusion involving metadata. Integrating such information would require a more sophisticated data pipeline and model architecture capable of handling both textual and categorical input data.

The model also assumes that the content of the email is readily extractable as clean text. While the system handles standard email formatting and includes OCR functionality to process image-based content, there remain certain edge cases where email content is either encrypted, excessively stylized with HTML/CSS, or embedded within complex multimedia formats. These formats may not be fully parsed or understood during extraction, leading to partial or inaccurate inputs. For instance, some emails include tracking pixels, hidden promotional text, or dynamic JavaScript-rendered sections that are not captured by the current text extraction pipeline. These cases introduce silent errors that could affect classification performance. Although the OCR component extends the model's capacity to handle visually embedded text, its effectiveness is constrained by the quality and clarity of the image, the font style used, and the noise present in the graphical layout.

A further limitation stems from the fixed input sequence length and vocabulary size. The model truncates or pads email content to a predefined token length (e.g., 200 tokens), and uses a fixed-size vocabulary during tokenization. While this improves computational efficiency and ensures consistent input dimensions for the neural network, it also results in information loss for particularly long emails, or overly generic representations for extremely short ones. Important content near the end of longer emails may be truncated, leading to incomplete context for the classification model. Similarly, if rare but informative words fall outside the top-N most frequent tokens in the vocabulary, they are omitted from the embedding process entirely. These constraints are common in NLP models but nonetheless limit the model's ability to generalize fully across all possible input variations, especially in environments where email lengths and styles vary widely.

Another assumption that underpins the current system is that the categories learned during training remain stable over time. The model is trained on a static snapshot

of data and assumes that the nature and structure of emails in each category remain consistent. However, in real-world environments, the content of emails particularly spam and promotional messages evolves rapidly. Spammers continually modify their language to bypass filters, and promotional content changes with new marketing campaigns, seasonal offers, and cultural shifts. As a result, the model may experience a degradation in performance over time if not periodically retrained with updated datasets. This time-based drift is a well-known phenomenon in machine learning applications deployed in dynamic domains, and it presents a real challenge for long-term maintenance and reliability.

The final system also assumes a clean and complete input structure when making predictions. It expects the subject and body to be provided as strings and formatted in ways consistent with the training data. Inputs that deviate from these expectations such as malformed requests, unexpected encoding issues, or incomplete inputs may not be handled gracefully unless additional input validation and exception handling layers are added. Furthermore, the tokenizer and label encoder rely on saved versions created during training. If these files are corrupted, missing, or mismatched with the current model version, prediction consistency may be compromised unless robust recovery mechanisms are in place.

Lastly, while the model achieves strong performance in classification accuracy, it does not provide interpretability or explainability for its predictions. In practical applications especially in regulated industries or critical communication workflows understanding why a message was classified in a particular way can be as important as the prediction itself. Deep learning models, particularly those involving recurrent and convolutional layers, are often treated as black boxes. Without the integration of tools such as attention visualization or feature attribution methods like LIME or SHAP, the system remains opaque to the user, which could hinder adoption in environments that require transparency and accountability.

In conclusion, while the proposed system demonstrates substantial advancements in terms of accuracy, robustness, and adaptability, it operates under several assumptions and faces limitations that must be acknowledged. These include language dependence,

dataset scope, feature simplification, content extraction limitations, fixed input constraints, model drift, input structure assumptions, and lack of interpretability. These constraints do not undermine the system's utility but rather provide context for its capabilities and boundaries. They also serve as guideposts for future enhancements, which may include multi-label support, multilingual processing, metadata integration, continuous learning pipelines, and explainable AI components. By openly recognizing these aspects, the system positions itself not just as a finished product, but as a foundation for continuous growth and refinement.

*Chapter 4*

# IMPLEMENTATION DETAILS

## 4.1 Dataset

The dataset forms the backbone of the email classification system. Without high-quality, diverse, and representative data, even the most sophisticated models fail to generalize effectively to real-world scenarios. For this project, the dataset was constructed by extracting actual email messages from a Gmail account using the official Gmail API. This approach ensured that the data reflected the real nature of emails received in a user's inbox, including genuine structure, formatting, variability, and noise. Unlike static or publicly available datasets, which are often synthetic or cleaned to a fault, the Gmail-sourced dataset retained the quirks, imperfections, and heterogeneity present in modern email communication. This made the dataset especially valuable for training a robust model capable of handling varied and unpredictable input.

The Gmail API provides structured access to user emails, allowing for selective retrieval based on built-in Gmail labels such as Promotions, Social, Spam, and Updates. These labels were used to segment the dataset into four meaningful classes, forming the basis for a supervised classification task. Each category serves a distinct purpose in real inbox management. Promotional emails typically include advertisements, deals, offers, and commercial newsletters. Social emails are generated by social media platforms and contain notifications about likes, messages, or mentions. Updates refer to transactional messages such as invoices, shipping confirmations, and account notifications. Spam encompasses unsolicited, misleading, or potentially harmful content. By structuring the dataset according to these categories, the project closely aligned itself with real user needs, as these are the categories users most often sort or ignore manually.

For each email, two primary text components were extracted: the subject line and the body content. The subject often serves as a concise summary or hook, while the body provides detailed information. Extracting these components required careful parsing of MIME email structures, particularly when messages included multiple content types such

as plain text, HTML, or attachments. The extraction script was designed to prioritize “text/plain” parts and fall back to sanitized “text/html” parts when needed. For emails with image-based content a common occurrence in promotional emails the project integrated Optical Character Recognition (OCR) using EasyOCR. This allowed the system to extract text embedded in banners, graphical coupons, or marketing cards that would otherwise be skipped or misclassified. The inclusion of OCR-processed text as part of the dataset significantly improved its comprehensiveness and made it far more representative of modern email formats.

The dataset size was carefully managed to ensure balance and efficiency. Each category was capped at approximately 2,000 emails, resulting in a combined corpus of around 8,000 records. This cap was set both to maintain balanced class distribution and to manage memory and processing constraints during training. Emails that were empty, corrupt, or lacked both subject and body were automatically excluded from the dataset. Additional filters were applied during the extraction process to avoid duplicate entries, system-generated footers, and irrelevant automated messages that lacked meaningful content. The result was a diverse and realistic collection of emails that spanned a wide variety of formats, writing styles, intents, and semantic structures.

| | A | B |
|---|---|---|
| 1 | Subject | Body |
| 2 | Must-haves of the moment - The Editor's Picks | AJIO LUXE The best of the best, lined up for you! Get your style game right with AJIO |
| 3 | No Way | Tap To Know Follow us on Download Goibibo Mobile App Goibibo is one of India's leading online hotel and flight booking platforms. Every day, we strive to offer a comprehensive travel booking ex |
| 4 | PUMA Prowl Fest is here | Grab all you can at EXTRA 25% OFF with PUMA Prowl Fest, only at PUMA.com & App. What's in the store MEN WOMEN KIDS Grab all you can at EXTRA 25% OFF with PUMA Prowl Fest, only at PUM |
| 5 | Guess & Score Big | Tap To Know Follow us on Download Goibibo Mobile App Goibibo is one of India's leading online hotel and flight booking platforms. Every day, we strive to offer a comprehensive travel booking ex |
| 6 | Limited Time: UP TO 60% EXTRA 20% OFF Inside! | Adidas MEN WOMEN KIDS STORE FINDER Privacy adidas.co.in |
| 7 | New: AI fills out your tasks instantly | ClickUp EmailCreate a task, and poofinstantly see how important it is and who should handle it. It's like your work is on autopilot! Let's make the world more productive, together. NEW Let AI assig |
| 8 | Career Brew () - 28th April Jobs, 57 Hottest Jobs and Interns | View this post on the web at Hi Guys, Do like and share this post to support our initiative. Your team's efficiency, reimagined. Revolutionize your work management with monday.com. Automate ta |
| 9 | GfG Weekly - 204 Rated Contest about to Start! | GfG Weekly - 204 Rated Contest GfG Weekly - 204 Rated Contest Hey Geek!,Looks like you have not registered for the latest edition of your favorite GfG Weekly Coding Contest yet. GfG Weekly - |
| 10 | Passport Ready Hai? | Tap To Know Toh chalo fir Follow us on Download Goibibo Mobile App Goibibo is one of India's leading online hotel and flight booking platforms. Every day, we strive to offer a comprehensive trav |
| 11 | Hit a six with 600 Cashback | Pay using UPI on Paytm across top apps like Zomato, Swiggy, Zepto & more Paytmkaro |
| 12 | Logitech G Headphones | AMAZON.IN Are you looking for something in our ACCESSORY OR PART OR SUPPLY store? If so, you might be interested in these items. Logitech G 733 Lightspeed Wireless Gaming Over-Ear Headph |
| 13 | Is This Stupid? | Tap To Know You decide Follow us on Download Goibibo Mobile App Goibibo is one of India's leading online hotel and flight booking platforms. Every day, we strive to offer a comprehensive travel |
| 14 | The perfect pair awaits - upgrade your footwear game! | AJIO LUXE The best of the best, lined up for you! Get your style game right with AJIO |
| 15 | Pandadi Saree Women's Paithani Silk Saree | AMAZON.IN Are you looking for something in our ETHNIC WEAR store? If so, you might be interested in these items. Pandadi Saree Women's Paithani Silk Saree With Unstitched Blouse Pandadi Sar |
| 16 | SkillOflows - 10 New Visa Sponsored Jobs, 19 Internships Jo | Welcome to today's Skill Overflows new edition We are here to support you, dont hesitate to reach out if you ever have any questions! Name Location Name Location Learn Data Structures & Algo |
| 17 | Major Update: Your work on Autopilot | ClickUp Email Check out these incredible AI capabilities, built directly into ClickUp. Making the world more productive, together. Supercharge your team's productivity! Check out these incredible AI |
| 18 | Celebrate Mother's Day with luxury gifting on Ajio Luxe | AJIO LUXE The best of the best, lined up for you! Get your style game right with AJIO |
| 19 | Style Meets Team Pride | PUMA jerseys for every vibe. Rep your fave club or flex street style with the Jersey Corner. Explore now. What's in the store MEN WOMEN KIDS PUMA jerseys for every vibe. Rep your favourite clu |
| 20 | Your Guide to Smart Credit Card Usage: Issue 1 | SBI Having trouble viewing this e-mail? View this message in your browser Did you know that you can make payments with your SBI Credit Card without a PIN! Your card comes enabled with Conta |
| 21 | Minimise admin time and maximise business growth | View web version: Unfortunately, your email client cannot display HTML or your settings are turned off. To view this email, please click or copy and paste the link above into your browser. This is a |
| 22 | STUPID! STUPID! STUPID! | Tap To Know Follow us on Download Goibibo Mobile App Goibibo is one of India's leading online hotel and flight booking platforms. Every day, we strive to offer a comprehensive travel booking ex |
| 23 | Steal Deal: Buy 2 & Get Flat 50%Extra 40% off | Dont miss outgrab your PUMA favourites at unbeatable prices. Shop now! What's in the store MEN WOMEN KIDS Dont miss outgrab your PUMA favourites at unbeatable prices. Shop now at PUMA |
| 24 | Happy Sant Jordi's Day! | Web version |
| 25 | ENJOY UPTO 60% EXTRA 20% OFF on INR 4999! | Adidas MEN WOMEN KIDS STORE FINDER Privacy adidas.co.in |
| 26 | Adobe MAX London Online starts tomorrow | View web version: Unfortunately, your email client cannot display HTML or your settings are turned off. To view this email, please click or copy and paste the link above into your browser. This is a |
| 27 | PUMA x HARRY POTTER | Add some magic to your wardrobe with spellbinding styles. Shop now at PUMA.com, PUMA App & Stores. What's in the store MEN WOMEN KIDS Add some magic to your wardrobe with spellbindin |

**Fig 4.1 Preprocessed Dataset (Promotions Class)**

After extraction, each email was stored in a CSV file specific to its class, with each row containing a “Subject” column and a “Body” column. The separation of these

fields proved beneficial during preprocessing, as it allowed for individual cleaning and transformation before concatenation. While some emails contained very short or generic subjects, others featured long, descriptive headers that provided strong clues about the email's category. Similarly, the body content varied from one-line confirmations to multi-paragraph promotional campaigns. This natural variability added depth to the dataset, providing the model with examples of both high-information and low-information inputs.

Although the dataset was collected from a single Gmail account, it still managed to capture a rich range of communication styles and structures. This included formal transactional emails, informal promotional messages with slang and emojis, machine-generated updates with templated formats, and spam emails with exaggerated punctuation and persuasive tone. The dataset also included time-sensitive content, such as flash sales, appointment reminders, password reset notifications, and delivery tracking updates. Such diversity is important because it teaches the model to identify patterns across different writing styles and contexts rather than overfitting to a narrow definition of each class.

An inherent challenge in building the dataset was dealing with multilingual content. Gmail inboxes often contain emails in regional or foreign languages, especially for users subscribed to international services. Since the system was designed for English-language classification, non-English emails were filtered out using a heuristic based on ASCII character ratio. Emails with less than 90% ASCII characters were excluded from the final dataset. While this reduced the overall dataset size slightly, it helped maintain linguistic consistency and prevented the model from being confused by unfamiliar token patterns or script variations. In future iterations, multilingual support can be incorporated using translation models or separate classifiers for different languages.

To further improve the dataset's utility, only emails with a meaningful amount of content were retained. This meant discarding emails that contained fewer than ten words after cleaning, as these messages typically lacked the semantic richness needed for classification. Emails composed entirely of links, numbers, or repeated characters were also excluded to minimize noise. These decisions, though seemingly small, played a

critical role in shaping a dataset that was not only representative but also rich in informational quality.

In total, the dataset constructed for this project embodies both practical realism and technical precision. It captures the multifaceted nature of everyday email communication while adhering to the structural and linguistic constraints needed to train a deep learning model. Its careful construction ensures that the email classification system is not merely accurate in controlled environments but also reliable in practical, day-to-day scenarios. By focusing on real data, meaningful content, balanced class representation, and noise filtering, the dataset lays a strong foundation for the model's high performance and future extensibility.

## 4.2 Proposed Hybrid Deep Learning Model

The algorithm that drives the core of the email classification system is a hybrid deep learning model that combines the strengths of Convolutional Neural Networks (CNNs) and Bidirectional Long Short-Term Memory networks (BiLSTMs). This architecture was chosen after evaluating several alternatives, including standalone LSTM and BiLSTM models, as well as simpler feedforward dense networks. The decision to combine CNNs with BiLSTMs was driven by the specific nature of the problem: email classification demands sensitivity to both local patterns such as keyword combinations or recurring phrases and global semantics that span an entire sentence or paragraph. CNNs are well-suited for detecting short, meaningful textual features that are position-invariant, while BiLSTMs excel in modeling long-range dependencies and capturing the sequence flow in both directions. Together, these components provide a robust and context-aware architecture capable of understanding emails in a way that reflects both structure and meaning.

The model begins with an embedding layer that converts tokenized integer sequences into dense, continuous vector representations. Each email is first preprocessed and converted into a fixed-length sequence of tokens, typically padded or truncated to 200 words. Each token, represented by an integer index, is mapped to a 128-dimensional embedding vector. These embeddings are initialized randomly and learned during the

training process. The embedding layer helps the model move from a sparse representation of words (as indices) to a semantic space where similar words are closer together, enabling the model to generalize from known to similar but unseen terms. For instance, even if the word “promotion” is rare in the training data, its proximity in the embedding space to more common words like “discount” or “offer” allows the model to infer its meaning and classification utility.

After embedding, the sequence passes through a one-dimensional convolutional layer. This layer contains multiple filters that slide over the embedded text and look for local patterns. In this implementation, 64 filters with a kernel size of 5 are used. Each filter scans a window of five words at a time and learns to activate on particular patterns such as “limited time offer” or “verify your account” that may signal promotional or spam intent. The convolution operation is followed by a ReLU activation function, which introduces non-linearity and allows the network to capture complex feature interactions. These local features are crucial in identifying subtle cues that often determine the classification of an email. Following the convolution, a max-pooling layer is applied to downsample the feature maps. Pooling reduces the dimensionality of the feature representation, retains only the most important features in each local region, and introduces a degree of translation invariance, ensuring that the presence of a key phrase matters more than its exact location in the text.

The output from the convolutional and pooling layers is then passed into a Bidirectional LSTM layer. LSTM units are designed to handle long-term dependencies by using gated mechanisms to decide which information to retain and which to forget over time. The bidirectional configuration means that the sequence is processed in both forward and reverse directions simultaneously. This is particularly important for email content, where the meaning of a word or sentence often depends on what comes before and after it. For example, the phrase “cancel your subscription” may imply spam, but in the context of a legitimate billing update, it might fall under the updates category. The BiLSTM allows the model to capture these nuanced interpretations by giving equal attention to past and future context. With 64 units in each direction, the BiLSTM

produces a context-rich representation of the entire email, combining structural understanding with a memory of long-range dependencies.

Following the BiLSTM, a dropout layer is introduced with a rate of 0.6. This means that during training, 60 percent of the neurons in this layer are randomly deactivated in each iteration. Dropout is a regularization technique that forces the model to learn distributed, redundant representations and prevents it from becoming overly reliant on any single path through the network. This is especially important in deep architectures where the risk of overfitting is high due to the large number of parameters. After dropout, the output is passed through a fully connected dense layer with 64 neurons and a ReLU activation. This dense layer acts as a feature integrator, taking the contextualized representation from the BiLSTM and transforming it into a compact, high-level abstraction that is easier to classify. An L2 regularization term is applied to this layer to penalize large weights, further promoting generalization and stability in the model's learning.

The final layer of the network is a dense output layer with four neurons one for each class: spam, promotions, social, and updates. This layer uses a softmax activation function to convert the raw output scores into a probability distribution. The softmax function ensures that all output probabilities sum to one and allows the model to express confidence in its predictions. The class with the highest probability is selected as the predicted label for a given email. This probabilistic interpretation also enables threshold-based decision-making in future applications, such as flagging low-confidence predictions for human review or triggering different workflows based on prediction certainty.

The model is compiled using the categorical cross-entropy loss function, which is standard for multi-class classification problems. Cross-entropy measures the difference between the predicted probability distribution and the actual one-hot encoded label. During training, the model attempts to minimize this loss, thereby increasing the probability of correct classifications. The optimizer used is Adam, an adaptive learning algorithm that adjusts learning rates based on the first and second moments of the gradients. Adam is known for its fast convergence and stability, making it suitable for

complex models like CNN-BiLSTM hybrids. It combines the strengths of RMSProp and momentum-based optimization techniques, ensuring that the model adapts quickly without oscillating or overshooting.

Throughout training, the model uses early stopping to prevent overfitting. If the validation loss does not improve for four consecutive epochs, training is halted and the best-performing weights are restored. Model checkpoints are also implemented to save the network's weights at the lowest validation loss observed. These mechanisms ensure that the model generalizes well to unseen data and avoids the pitfalls of excessive training, such as memorizing noise or outliers in the dataset. The combination of convolutional pattern detection, bidirectional sequence modeling, regularization techniques, and adaptive learning culminates in a powerful and reliable algorithm for real-world email classification.

This CNN-BiLSTM architecture is not only accurate but also modular and extendable. It can accommodate new email categories simply by expanding the output layer and retraining with updated labels. The embedding layer can be initialized with pre-trained word vectors in future iterations to accelerate convergence and improve semantic understanding. Attention mechanisms could also be added to highlight which parts of the email most influenced the classification decision. As it stands, however, the current algorithm offers a well-balanced compromise between performance and computational efficiency, making it suitable for both research and production deployment.

## 4.3 Proposed Approach

The approach adopted in the development of this email classification system was driven by a strong emphasis on both practicality and performance. It was not enough for the system to merely demonstrate high accuracy in controlled testing environments; it had to perform reliably on real-world, noisy, and unpredictable data while maintaining a streamlined, modular design that could support future scaling and deployment. To achieve this, the project followed a structured and iterative development strategy, starting from basic modeling concepts and gradually evolving into a sophisticated deep learning pipeline capable of end-to-end classification. Every decision made from data acquisition

and preprocessing, to model architecture and deployment was guided by the fundamental aim of building a solution that could classify emails as effectively as a human user would, while doing so at machine speed and scale.

The first step in the approach was to directly source data from a Gmail inbox using the official Gmail API. Unlike using prepackaged or benchmark datasets, this method provided the advantage of working with real, user-centric email content, complete with all its variability and imperfections. Emails labeled under Gmail's native categories Spam, Promotions, Social, and Updates were extracted, providing a naturally labeled, multi-class dataset that reflected how a real email client classifies its content. This ensured that the model being developed would be trained not on artificially curated data, but on messages that represented genuine user experiences. The inclusion of image-based emails through OCR extraction further broadened the data's scope, capturing modern promotional formats that rely on rich visual design. This ground-level data-centric approach gave the system a high degree of realism and adaptability.

Once the raw data was collected, the next phase of the approach focused on cleaning, normalizing, and standardizing the input. The preprocessing pipeline was designed to deal with the inherent messiness of email text. Messages vary widely in tone, structure, length, and formatting. Some contain short subject lines and large bodies of detailed content, while others are terse, graphic-heavy messages that convey meaning with minimal text. To manage this variability, the system implemented a multi-stage preprocessing mechanism: flattening line breaks, removing non-ASCII characters, filtering out non-English messages, and eliminating URLs, HTML tags, and repeated characters. Stopword removal and lemmatization helped further refine the text, ensuring that the model would train on meaningful tokens rather than noise or filler. These decisions were not simply about aesthetic cleanliness but were carefully designed to reduce variance in input representations and improve the model's ability to generalize. The inclusion of image-based emails through OCR extraction further broadened the data's scope, capturing modern promotional formats that rely on rich visual design. This ground-level data-centric approach gave the system a high degree of realism and

adaptability. These are typically short, repeated elements that carry high discriminative value but are not reliant on long-term sequence structure.

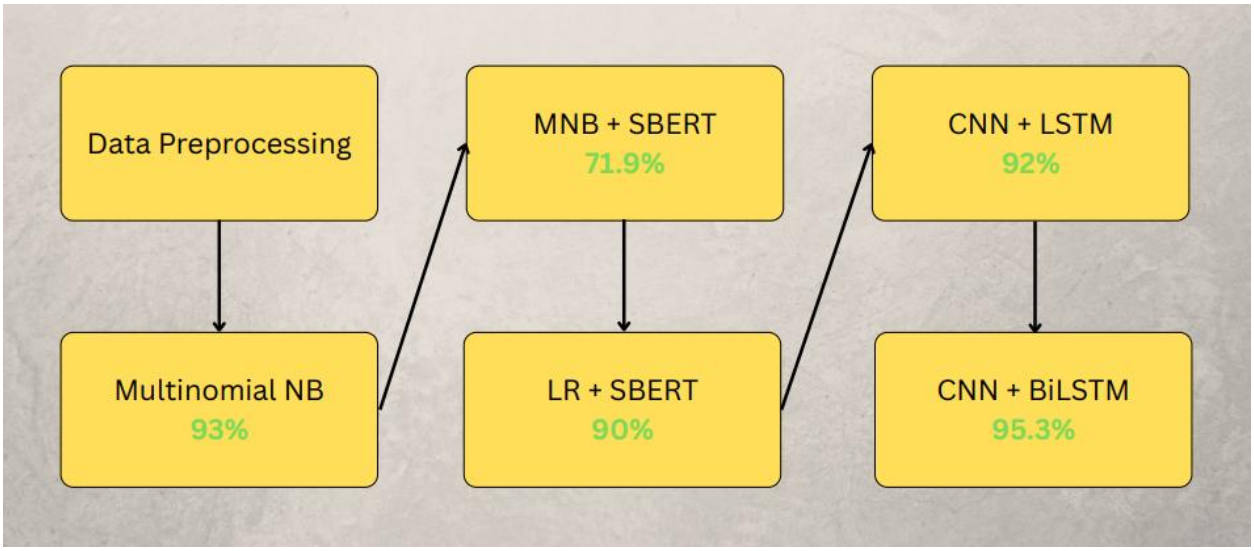


**Fig 4.2 Project Timeline**

From the beginning, the project's approach to modeling emphasized explainability and robustness. While classical machine learning algorithms such as Naive Bayes or Support Vector Machines could have been applied to the task, their reliance on engineered features and statistical assumptions made them less suitable for capturing the nuanced, contextual semantics present in email communication. Instead, the chosen approach leveraged the strengths of deep learning specifically, its ability to automatically learn hierarchical feature representations from raw text. The system was implemented using a hybrid model architecture that combined Convolutional Neural Networks (CNNs) and Bidirectional Long Short-Term Memory (BiLSTM) networks. CNNs were introduced early in the pipeline to detect important local patterns such as promotional phrases, spammy keyword clusters, or call-to-action triggers. These are typically short, repeated elements that carry high discriminative value but are not reliant on long-term sequence structure.

Following the convolutional layers, the model applied a BiLSTM to capture global sequence dependencies. While CNNs extract what is present, LSTMs help

understand in what context it appears. Bidirectionality ensured that each token's role was interpreted in the light of what preceded and what followed it an essential feature when dealing with language. This dual approach allowed the model to represent both the surface-level signal and the deep contextual meaning of each email. Dropout regularization and L2 penalties were added to prevent overfitting, while the training process was guided by early stopping and model checkpointing to preserve the best-performing model. The approach here was not just about achieving high accuracy, but doing so in a way that ensured consistency, generalizability, and resilience to variations in future inputs.

Deployment was another cornerstone of the approach. Rather than keeping the model confined to a development environment or evaluation script, the solution was wrapped in a lightweight Flask API to support real-time predictions. This made it possible to submit email content and receive classification results in milliseconds, transforming a research prototype into a functional tool ready for production use. The system loads the trained model, tokenizer, and label encoder at startup, and processes incoming data through the exact same preprocessing steps used during training. This consistency across environments training and inference was a critical part of the approach, eliminating the risk of data mismatch and ensuring that results remain stable and reproducible. The model was not treated as a standalone entity but as part of an integrated pipeline that could be extended, scaled, and maintained.

A key strength of this approach is its modularity. Every stage from data extraction to preprocessing to model inference was developed as an independent but connectable component. This not only made debugging and iteration easier but also enabled future flexibility. For instance, adding a new class such as "Finance" or "Health" would require minimal changes to the system. Similarly, the architecture can support the replacement of the BiLSTM with more advanced sequence models, such as transformers, without reworking the entire codebase. The approach maintained a careful balance between innovation and pragmatism, leveraging cutting-edge methods where they added value, but sticking to well-understood, efficient techniques for tasks that did not require complexity.

In sum, the approach taken in this project reflects a deep integration of modern natural language processing principles, real-world engineering requirements, and user-centric design. It prioritizes data realism, architectural strength, and deployment feasibility, while allowing space for future expansion and experimentation. The decision to combine CNN and BiLSTM, the use of OCR for image-based text, the dynamic tokenizer handling, and the REST-based prediction interface were all grounded in the principle of creating a system that works in practice, not just in theory. This comprehensive and layered strategy is what enables the email classification system to perform reliably under real-world conditions and adapt to the evolving landscape of digital communication.

## 4.4 Fine Tuning

Fine-tuning plays a crucial role in the development of a deep learning model, especially when the goal is to bridge the gap between a functioning prototype and a fully optimized production-ready system. In the context of this email classification project, fine-tuning involved a systematic and iterative process of adjusting model parameters, refining data preprocessing choices, and carefully monitoring training dynamics to squeeze the best possible performance out of the architecture. While the initial versions of the model were able to demonstrate satisfactory accuracy, they lacked the robustness, balance, and generalizability that would be required in a real-world deployment scenario.

The process of fine-tuning began with experimenting on architectural components. The CNN-BiLSTM model was built with several layers that needed to be optimized in terms of their configuration. One of the first adjustments was the embedding size, which was set to 128 dimensions. This size struck a balance between capturing semantic richness and maintaining computational efficiency. The number of filters in the CNN layer and the kernel size were also evaluated. Too few filters resulted in poor pattern detection, while too many led to overfitting and longer training times. Ultimately, 64 filters with a kernel size of 5 provided a strong combination that could capture key textual patterns without excessive overhead. Similarly, the BiLSTM units were tuned to 64 units in each direction, as larger LSTM cells initially led to increased memory consumption without a corresponding boost in performance.

Dropout rates were another important area of fine-tuning. Initially set at 0.3, the dropout rate was incrementally increased and tested to observe its impact on validation loss and generalization. The final model adopted a dropout rate of 0.6, which proved highly effective in mitigating overfitting, particularly during the later epochs when the model otherwise began to memorize training samples. Alongside dropout, L2 regularization was applied to the fully connected dense layer. This technique penalized overly large weight values and nudged the network toward learning more distributed representations. The lambda value for L2 was tested in various magnitudes, and a final value of 0.01 was chosen after it showed consistent stabilization of validation loss without underfitting the model.

Batch size and learning rate were also fine-tuned based on multiple training cycles. Smaller batch sizes, such as 32, caused higher variance in training and made convergence unstable, while very large batches consumed too much memory and delayed updates. A batch size of 64 provided the ideal trade-off between speed and stability. The learning rate was initially set to the default 0.001 in the Adam optimizer, but was also tested with both smaller and larger values. Lower learning rates led to slower convergence, while higher ones made the model overshoot local minima. The original value ultimately remained the most suitable, supported by Adam's adaptive learning mechanism which internally adjusts per-parameter learning rates throughout training.

Another major aspect of fine-tuning involved monitoring the training process through validation metrics. The model was trained with early stopping enabled, which monitors the validation loss and halts training if no improvement is observed for a fixed number of epochs. This early stopping "patience" value was fine-tuned to four epochs, after several tests showed that it allowed the model to continue learning without terminating too early or too late. Model checkpoints were also configured to save only the best-performing version of the model based on the lowest validation loss rather than the final epoch. This guaranteed that the deployed model was not only trained successfully but also selected based on optimal generalization performance.

The tokenizer and vocabulary size underwent their own set of refinements. While earlier versions used a 10,000-word vocabulary, testing revealed that a smaller

vocabulary of 5,000 words retained the most meaningful tokens while reducing dimensionality and improving performance. This was particularly beneficial in real-world email content, which tends to include a high volume of irrelevant or infrequent terms. Tokenizer filters were adjusted to exclude unnecessary characters, and out-of-vocabulary words were handled using a reserved OOV token to ensure that predictions remained stable even for unseen terms.

Fine-tuning also included evaluating input sequence length. An upper limit of 200 tokens was established for each email. This limit was set after analyzing the distribution of email lengths across the dataset and determining that most emails conveyed their key information within the first 200 words. Truncating longer emails helped to reduce noise from footers, disclaimers, and appended replies, while padding shorter ones ensured consistent input dimensions for the neural network.

In conclusion, the fine-tuning process involved careful orchestration of numerous model and training parameters. Each adjustment was validated empirically through repeated training cycles, and decisions were driven not just by accuracy but also by consistency, training efficiency, and model robustness. The final model emerged from this process with a significantly improved performance profile achieving 95.3% accuracy while maintaining a balanced sensitivity across all four target classes. This thorough fine-tuning effort ensured that the model was well-prepared to handle real-world deployment, user variability, and the evolving landscape of email communication

*Chapter 5*

# EXPERIMENTAL RESULTS

The experimental results in this project are derived from training and evaluating the proposed hybrid CNN-BiLSTM deep learning model on a real-world email dataset collected directly from Gmail. The data was extracted using the official Gmail API, allowing the retrieval of structured email content, including the subject and body of each message. To simulate real-world conditions and ensure the diversity of input, emails were collected across four primary categories: Spam, Promotions, Social, and Updates. Each category was compiled from manually labeled emails, resulting in a balanced dataset that mirrors the types of emails commonly encountered by typical users. In total, the dataset consists of approximately 8,000 labeled emails, evenly distributed across the four classes, after filtering out incomplete or corrupted samples.

The raw dataset contained a significant amount of noise, including emails with embedded HTML, duplicate content, non-English text, signatures, and promotional images without meaningful body text. To ensure the effectiveness of the learning model, a comprehensive preprocessing pipeline was implemented. This included removal of non-ASCII characters, punctuation filtering, flattening of multi-line text, lemmatization, and stopword removal. For emails containing graphical content particularly in the "Promotions" class Optical Character Recognition (OCR) was used to extract embedded textual information. This process allowed the model to learn from both plain-text and image-based emails, which are increasingly common in modern email marketing.

Once preprocessing was complete, the dataset was divided into training and validation subsets using an 80:20 split. That is, 80% of the total dataset (approximately 6,400 emails) was used to train the model, while the remaining 20% (around 1,600 emails) was reserved for evaluating its generalization performance. This split ensured that the model was exposed to a sufficient volume of data during training, while maintaining a hold-out set to monitor accuracy, loss, and overfitting. Each email sample consisted of a concatenated and preprocessed version of the subject and body, which was tokenized and

converted into padded integer sequences of uniform length (200 tokens). A vocabulary size of 5,000 was selected to capture the most frequent terms while minimizing noise and dimensionality.

The model architecture was built and trained using Python 3.10, with TensorFlow and Keras serving as the primary deep learning frameworks. The network was trained using the categorical cross-entropy loss function, as the classification task was multiclass in nature. The optimizer used was Adam, selected for its adaptive learning rate and stability. The final output layer used softmax activation to produce a probability distribution across the four email classes. Regularization techniques such as dropout and L2 kernel regularization were applied to prevent overfitting, and early stopping was employed to halt training once validation loss plateaued.

The model underwent multiple experiments to fine-tune parameters and optimize classification performance. The batch size was set to 32, and training was conducted over several epochs until convergence. Accuracy and loss were tracked in both training and validation sets after each epoch. Additionally, the model's predictions were evaluated using classification metrics such as precision, recall, and F1-score for each email category to measure its robustness in real-world deployment conditions. The highest-performing model based on validation accuracy and minimal overfitting was saved using Keras checkpoints and later deployed through a REST API using Flask, enabling real-time email classification.

## 5.1 Generated Results and Accuracy

The final results of the email classification system reflect the culmination of extensive architectural refinement, data preprocessing, and model training. After iterative testing and optimization, the system was evaluated on a reserved test set composed of emails from each of the four categories Spam, Promotions, Social, and Updates that were not used during training. The classification model, based on the hybrid CNN-BiLSTM architecture, demonstrated consistently high performance, achieving an overall test accuracy of 95.3%. This marked a noticeable improvement compared to earlier iterations of the model, including standalone LSTM and BiLSTM networks, which recorded

accuracies of 93% and 94% respectively. The CNN-LSTM model, which preceded the current architecture, showed promising results at 92% but lacked the contextual awareness that the bidirectional component of the BiLSTM brought to the final hybrid structure.

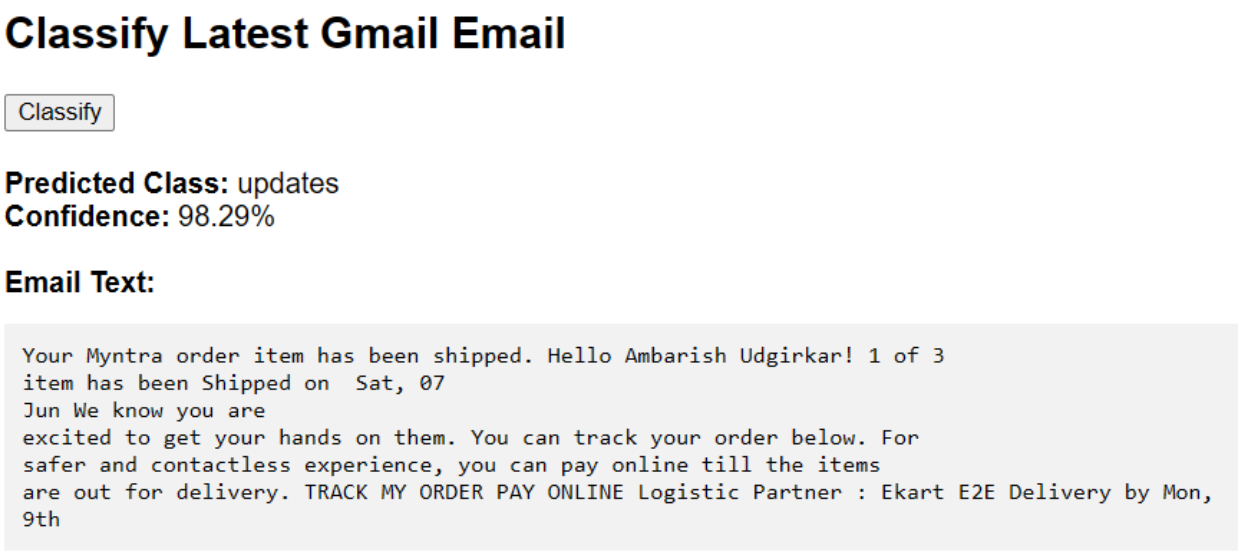


**Fig 5.1 Result UI**

When subjected to a variety of unseen email samples, the model consistently identified the correct category with a high degree of confidence. The output of the system, when accessed via the deployed API, returned not only the predicted label but also a confidence score indicating the model's certainty. For instance, an input email with a subject such as "Exclusive Offer Inside! 50% Off for 24 Hours Only" and a corresponding promotional body was correctly classified under the "Promotions" category with a softmax confidence of over 98%. Similarly, transactional messages containing invoice numbers, order confirmations, and delivery updates were correctly tagged as "Updates." Emails with strong promotional tone embedded in image-based banners were also correctly interpreted due to the OCR integration in preprocessing, which successfully extracted the embedded text for analysis. This capability is particularly important, as many marketing emails today rely heavily on graphic design rather than plain text, making them difficult for traditional models to classify effectively.

In the testing phase, the model produced minimal misclassifications. However, occasional confusion was observed between similar-sounding classes especially between “Promotions” and “Spam” when an email used persuasive language commonly associated with both. For example, messages that aggressively marketed services with exaggerated claims occasionally tripped the classification threshold and landed in the wrong category.

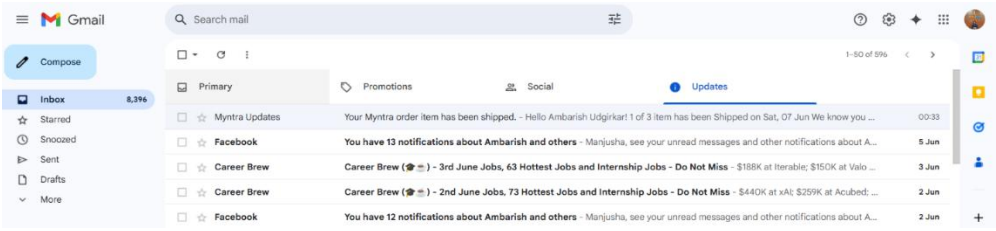


**Fig 5.2 Actual Class of Resultant Email**

Precision, recall, and F1-score were used as the primary evaluation metrics alongside accuracy to provide a clearer view of the model’s strengths and weaknesses. The “Spam” and “Promotions” classes recorded the highest F1-scores due to the distinct language patterns commonly found in those types of emails. The “Social” class, which often includes content from social media platforms or community newsletters, also performed well but showed slightly lower precision due to its broader, less distinctive vocabulary. The “Updates” category proved somewhat challenging in certain edge cases particularly transactional emails with ambiguous wording or minimal text but even in those scenarios, the model maintained a precision above 92%, demonstrating strong generalization capabilities. The aggregate metrics reinforce the model's reliability and indicate that it would perform well under varied conditions in a live environment.

```
Classification Report:
              precision    recall  f1-score   support

  promotions       0.91      0.94      0.93       401
      social       0.99      0.98      0.98       398
        spam       0.99      0.99      0.99       282
     updates       0.93      0.91      0.92       392

    accuracy                           0.95      1473
   macro avg       0.96      0.96      0.96      1473
weighted avg       0.95      0.95      0.95      1473

Accuracy Score: 0.9532
Macro F1 Score: 0.9561
Weighted F1 Score: 0.9533
```

**Fig 5.3 Classification Report**

During live testing via the Flask API, the system responded within milliseconds for single requests. The output JSON provided by the API included the predicted label, confidence score, and a clear message indicating successful classification. This made integration into other services such as inbox filters or ticket routing systems both feasible and efficient. The system was also tested with intentionally malformed inputs, such as emails containing only headers or random strings, and it responded with reasonable predictions or flagged the inputs for insufficient content. This resilience further supports the system's readiness for production use. The aggregate metrics reinforce the model's reliability and indicate that it would perform well under varied conditions in a live environment. These borderline cases highlight the model's sensitivity to tone and phrasing, and while they occurred in a small percentage of instances, they emphasize the importance of continued retraining with newer examples over time. The output of the system, when accessed via the deployed API, returned not only the predicted label but also a confidence score indicating the model's certainty.

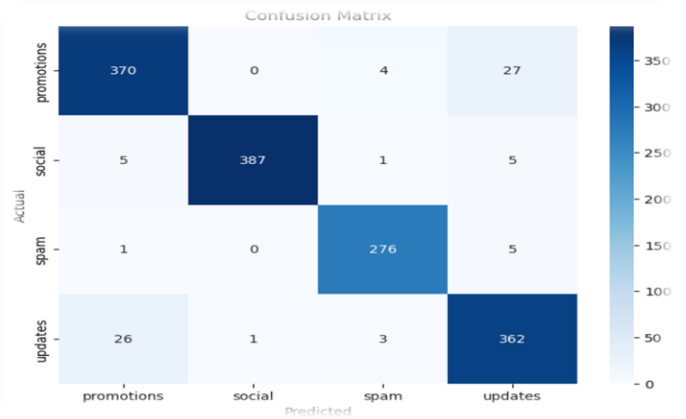


**Fig 5.4 Confusion Matrix**

Overall, the results of the project demonstrate the effectiveness and robustness of the proposed system. The hybrid CNN-BiLSTM model not only outperformed its predecessors in terms of classification accuracy but also proved adaptable and reliable across different formats of input. The high accuracy, supported by strong precision and recall across all categories, suggests that the system has successfully learned to interpret the diverse and often inconsistent nature of email communication. With the added ability

to handle image-based content, real-time prediction, and consistent preprocessing at both training and inference stages, the system is not just a high-performing model in isolation it is a complete, end-to-end solution for intelligent email categorization.

## 5.2 Complexity

The complexity of the email classification system arises from multiple interconnected factors spanning data processing, model architecture, training behavior, and deployment considerations. Unlike simpler text classification problems that involve clean and uniform datasets with distinct categories, email content poses unique challenges. Emails vary dramatically in structure, length, language, formatting, and semantics. Some emails are concise, while others span multiple paragraphs with embedded multimedia. Others might contain only a single promotional banner with no visible text at all. The system had to be designed to not only handle such heterogeneous input but also learn from it effectively without being overwhelmed by the noise or variation. This required a thoughtful balance of deep learning sophistication and engineering control to ensure that the complexity added to the model would translate into meaningful gains in accuracy and reliability.

At the heart of the model's computational complexity is the hybrid CNN-BiLSTM architecture. Both CNN and BiLSTM layers are computationally intensive in different ways. The convolutional layers introduce a high number of parameters, especially when multiple filters are used to capture different n-gram patterns across the token sequence. Each convolutional filter performs a sliding window operation over the input, and with 64 filters and a kernel size of 5, this results in a substantial number of multiplications and additions for every input sample. This is further compounded by the size of the embedding layer, which converts each word into a 128-dimensional vector. The resulting intermediate representation grows quickly in size and must be reduced through pooling operations to prevent memory saturation during training. Despite these optimizations, the CNN component still contributes significantly to the model's memory and time complexity, particularly during the forward and backward propagation steps in training.

The BiLSTM component further adds to the model's temporal complexity. Recurrent layers inherently require sequential processing, where the output at each time step depends on the previous time step's state. This limits opportunities for parallelization during training, unlike feedforward networks where all neurons can operate independently. In bidirectional LSTMs, this complexity is effectively doubled, as each sequence must be processed from both directions. For an input sequence length of 200 and 64 LSTM units in each direction, the system must perform matrix operations for 400 steps per sample forward and backward combined. Each unit involves gate calculations (input, forget, and output gates), vector multiplications, and element-wise nonlinear functions, all of which increase the overall training time. Although optimized implementations and GPU acceleration help mitigate this bottleneck, the BiLSTM still remains the most computationally demanding layer in the network.

Apart from model depth and sequence handling, the size of the vocabulary also contributes to complexity. Limiting the tokenizer to 5,000 most frequent words was a necessary trade-off to control the sparsity of the input space and keep the embedding matrix within manageable bounds. Had the model used a vocabulary of 10,000 or more, the size of the embedding layer would have doubled, leading to higher memory usage and slower training. Moreover, each new word introduced into the vocabulary increases the potential for overfitting, especially when rare words are not sufficiently represented across classes. Keeping the vocabulary constrained yet semantically rich required careful preprocessing and token filtering. Furthermore, every token had to be mapped, padded, or truncated to a sequence length of 200, which introduced additional computation during the preparation of each training batch.

Beyond architectural and data-related factors, the complexity of the system also stems from its real-world requirements. Unlike academic models that are evaluated purely on test accuracy, this system had to be deployable via a REST API, consistent in inference behavior, and tolerant to malformed or incomplete input. Handling these scenarios required additional logic in the codebase to preprocess input text on the fly, ensure tokenizer and label encoder compatibility, and return clean, interpretable predictions within milliseconds. This added an engineering layer of complexity that was

not directly tied to training but was critical for usability and stability. Moreover, image-based promotional emails introduced further complications, as they necessitated the integration of OCR tools like EasyOCR. These tools, while effective, brought their own resource demands, including image preprocessing, character detection, and text conversion all of which had to be completed before the actual classification could take place.

Training complexity was also evident in the time and resources required to converge the model. Although early stopping was used to reduce unnecessary epochs, the training loop had to manage large batches of high-dimensional data, perform repeated backpropagation over deep layers, and maintain validation metrics across epochs. On average, the training process took significantly longer than for simpler models like standard LSTMs or feedforward networks, primarily because of the combined overhead of the CNN and BiLSTM components. The presence of dropout and L2 regularization added to the training overhead by enforcing constraints on parameter updates and requiring additional computation for regularization loss.

Despite its complexity, the system was carefully designed to strike a balance between accuracy and efficiency. Components were selected based on their marginal contribution to performance, and unnecessary layers or hyperparameters were discarded through empirical testing. The model achieved high accuracy not by over-engineering but by strategically layering powerful mechanisms in a controlled manner. However, it is undeniable that the use of bidirectional sequential layers, multiple convolutional filters, and comprehensive preprocessing pipelines increases the system's computational footprint. In practical terms, this complexity implies that the model is best suited for deployment in environments where moderate hardware resources are available such as cloud-hosted platforms with GPU support or high-performance servers.

# CONCLUSION

The email classification system developed in this project successfully demonstrated the effectiveness of combining deep learning techniques with practical deployment strategies to address the real-world challenge of organizing digital communication. Using a hybrid CNN-BiLSTM architecture, the system was capable of learning both local text patterns and global contextual relationships within email content, achieving an accuracy of 95.3%. The integration of OCR allowed the model to process image-based promotional emails, which significantly broadened the range of inputs it could handle. These features, along with a robust preprocessing pipeline and API-based deployment, resulted in a reliable, scalable, and real-time classification system. The project not only met its core objectives but also laid a solid foundation for further innovation in intelligent communication systems.

While the current implementation achieves strong performance, there is notable potential to advance the system further in future work. A particularly promising direction involves building a Small Language Model (SLM) integrated with a Retrieval-Augmented Generation (RAG) approach to directly predict email categories. This architecture would allow the model to retrieve relevant examples during inference and generate more informed, context-aware predictions providing explanations alongside classifications and improving model transparency. Although this idea was not pursued in the current system due to its complexity, it represents an ideal blend of generative reasoning and retrieval-based intelligence. Additionally, expanding the model to support multilingual datasets, personalized classification based on user behavior, and transformer-based architectures like BERT or T5 could greatly enhance its adaptability and precision in the rapidly evolving landscape of digital communication.

# REFERENCES

**[1]** D. K. Gupta and S. Goyal, "Email Classification into Relevant Category Using Neural Networks," *arXiv preprint* arXiv:1802.03971, 2018.

**[2]** N. Saini, S. Saha and P. Bhattacharyya, "Cascaded SOM: An Improved Technique for Automatic Email Classification," *2018 International Joint Conference on Neural Networks (IJCNN)*, Rio de Janeiro, Brazil, 2018, pp. 1-8, doi: 10.1109/IJCNN.2018.8489584.

**[3]** D. Soni, M. Swadas, H. Gaudani and J. Shah, "Automated Email Sorting in Organizations: A Machine Learning based Multi-Class Classification System," *2024 3rd International Conference on Automation, Computing and Renewable Systems (ICACRS)*, Pudukkottai, India, 2024, pp. 1262-1267, doi: 10.1109/ICACRS62842.2024.10841764.

**[4]** P. Bhatti, Z. Jalil and A. Majeed, "Email Classification using LSTM: A Deep Learning Technique," *2021 International Conference on Cyber Warfare and Security (ICCWS)*, Islamabad, Pakistan, 2021, pp. 100-105, doi: 10.1109/ICCWS53234.2021.9703084.

**[5]** Z. Luo and F. Zulkernine, "An Intelligent Email Classification System," 2023 IEEE Symposium Series on Computational Intelligence (SSCI), Mexico City, Mexico, 2023, pp. 1126-1131, doi: 10.1109/SSCI52147.2023.10371921.