

Final Report



CSE 543: Information Assurance and Security

Group 1: Final Project Report

**RNN-Based Phishing Email Detection Using LSTM, GRU, and
Bidirectional RNN**

ASU ID	Name	Email	Role
1233435959	Varun Gupta	vgupt135@asu.edu	Leader
1237229502	Tejas Uttare	tuttare@asu.edu	Deputy Leader
1237802997	Anshul Chennigari	achennig@asu.edu	Member
1238235455	Sree Sai Harshitha Nanjyala	snanjyal@asu.edu	Member
1238279577	Harshan Karadaguru Anjaneyareddy	hkaradag@asu.edu	Member
1237142792	Siddhesh Vinayak More	smore123@asu.edu	Member
1238562119	Ziyad Jafarulla Parkar	znolas13@asu.edu	Member
1228729413	Siddharth Axaykumar Mehta	smehta74@asu.edu	Member
1236283089	Pravallika Chepuri	pchepur1@asu.edu	Member

Contents page

Contents page	3
1. INTRODUCTION	6
1.1 Motivation and Background	6
1.2 Goals and Scope of the Study	7
1.3 Scope of Study	8
2. SUMMARY OF ACCOMPLISHMENTS OF THE PROJECT	8
3. RESPONSIBILITIES OF EACH GROUP MEMBER	11
4. DETAILED RESULTS	16
4.1 System Overview	16
4.1.1 End-to-End Pipeline	16
4.1.2 Flow of Data	16
4.1.3 Module Structure	17
4.2 Dataset Description	17
4.2.1 Dataset Sources	17
4.2.2 Data Format and Label Normalization	18
4.2.3 Class Distribution	19
4.3 Data Preprocessing	19
4.3.1 Data Cleaning	20
4.3.2 Text Normalization	21
4.3.3 Tokenization	21
4.3.4 Sequence Padding	21
4.4 Feature Representation	22
4.4.1 Comparison of Feature Representation Methods	22
4.4.2 Impact on Model Performance	23
4.5 Model Architecture	24
4.5.1 Simple RNN	24
4.5.2 LSTM (Long Short-Term Memory)	24
4.5.3 GRU (Gated Recurrent Unit)	24
4.5.4 Bidirectional SimpleRNN (BiRNN)	24
4.5.5 Model Comparison Summary	24
4.6 Baseline Machine Learning Models	25
4.6.1 Naive Bayes	25
4.6.2 Logistic Regression	25
4.6.3 Random Forest	25

4.6.4 Multi-Layer Perceptron (MLP)	25
4.6.5 Comparison with Deep Learning Models	25
4.7 Training Process	26
4.7.1 Loss Function	26
4.7.2 Optimizer	27
4.7.3 Epochs and Batch Size	27
4.7.4 Training Flow Summary	27
4.8. Hyperparameter Tuning	27
4.8.1 Purpose and Setup	27
4.8.2 Hyperparameters Investigated	28
4.8.3 Final Hyperparameter Selection	28
4.9 Evaluation Metrics	29
4.9.1 Importance of Evaluation in Phishing Detection	29
4.9.2 Confusion Matrix	29
4.9.3 What the Metrics Reveal in Practice	30
4.10 Experimental Results	30
4.10.1 Deep Learning Model Results	30
4.10.2 Baseline Model Results	32
4.10.3 Comprehensive Model Comparison	33
4.11 Visualization	36
4.11.1 Multi-Metric Bar Charts	36
4.11.2 Scatter Plot Analyses	37
4.11.3 Cross-Model Performance Heatmap	38
4.11.4 Radar Charts	38
4.11.5 Matthews Correlation Coefficient Comparison	40
4.11.6 Raw Error Count Comparison	40
4.11.7 Confusion Matrix Heatmaps	41
4.11.8 Confusion Matrix Summary	44
4.12 Error Analysis	44
4.12.1 Simple RNN: A Case Study in Catastrophic Failure	44
4.12.2 False Positive Analysis	45
4.12.3 False Negative Analysis	46
4.12.4 Total Error Comparison and Error Rate Analysis	47
4.12.5 Key Error Analysis Findings	47
4.13 Discussion	48
4.13.1 Deep Learning vs. Classical Machine Learning: A Nuanced Picture	48

4.13.2 LSTM as the Best Deep Learning Model	48
4.13.3 GRU: The Recall Champion	49
4.13.4 The Simple RNN Failure: Vanishing Gradient in Practice	49
4.13.5 The Role of URL Tokenization	50
4.14 Limitations	50
4.14.1 Dataset Limitations	50
4.14.2 Model Constraints	51
4.14.3 Deployment Limitations	52
4.15 Federated Learning for Privacy-Preserving Detection	53
4.15.1 Explainable AI (XAI) Integration	53
4.15.2 Additional Future Directions	53
5. CONCLUSIONS AND RECOMMENDATIONS	53
5.1 Conclusions	53
5.2 Recommendations	55
5.2.1 Use of Advanced Deep Learning Architectures	55
5.2.2 Dataset Expansion and Diversity	56
5.2.3 Multimodal Feature Integration	56
5.2.4 Real-Time Detection Systems	56
5.2.5 Privacy-Preserving Learning	56
5.2.6 Explainable AI Integration	57
5.2.7 Continuous Adaptive Learning	57
5.3 Final Remarks	57
6. REFERENCES	59

1. INTRODUCTION

1.1 Motivation and Background

Phishing has quietly become one of the most persistent and damaging threats in today's digital world. Every day millions of emails are sent with the intention of tricking users into sharing sensitive information such as passwords, banking details, or personal identity data. What makes phishing especially dangerous is not just its scale, but how convincingly attackers can imitate trusted sources. As communication has increasingly shifted online, email has become a primary entry point for these attacks, making detection more crucial than ever. [9][21]

If we look back, earlier phishing detection systems were built on simple ideas. They relied on predefined rules, blacklists, and known signatures of malicious behavior. These approaches worked reasonably well when attacks followed predictable patterns. But over time, attackers evolved. They began using techniques like domain spoofing, misleading links, careful language, and even personalized messages that closely resemble legitimate communication. Because of this, traditional systems now struggle to keep up, especially when faced with completely new or unseen attack styles. [32]

Machine learning brought an important shift by allowing systems to learn patterns from data instead of relying only on fixed rules. While this was a step forward, these models often depended heavily on manually designed features. This means their performance still relied on how well those features were engineered, which can limit their ability to capture deeper patterns in complex text data like emails.[16][39]

This is where deep learning becomes truly valuable. Instead of depending on handcrafted features, deep learning models learn directly from raw data and automatically discover patterns at multiple levels. This makes them particularly effective for tasks involving natural language, where meaning often depends on context and sequence. [15][20]

Recurrent Neural Networks are especially suited for this kind of problem because they are designed to handle sequential data. Variants like Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) improve upon basic RNNs by handling long term

dependencies more effectively, allowing the model to understand relationships across entire sentences or even full emails. Bidirectional models go one step further by looking at text in both directions, giving a more complete understanding of context. [34][24]

So because phishing often relies on subtle cues in language and structure, these capabilities become extremely important. Deep Learning based systems can capture patterns that traditional approaches simply miss. In the context of Information Assurance, where reliability and accuracy are critical, this makes deep learning not just useful, but necessary for building modern and effective phishing detection systems.

1.2 Goals and Scope of the Study

The main goal of this project is to build a reliable and intelligent system that can distinguish between phishing and legitimate emails using deep learning techniques. Instead of focusing on a single model, the idea is to explore and compare multiple approaches to understand what works best for the problem.

A key part of the project is to evaluate different deep learning models such as RNN, LSTM, GRU and Bidirectional RNN, and compare them into traditional machine learning methods. This comparison helps highlight the strengths and weaknesses of each approach and provides a clearer picture of how much improvement deep learning can offer.

Another important goal is to reduce false positives. In real world systems, incorrectly marking legitimate email as phishing can be just as problematic as missing an attack. It can disrupt communication and reduce user trust. So the focus is not just on achieving high accuracy, but also on making sure the system behaves reliably in practical scenarios. [31]

This project also aims to create a clean and structured pipeline that takes raw email data, processes it, trains models, and evaluates their performance in a consistent way. This ensure that results are reproducible and can be built upon in future work.

1.3 Scope of Study

This project focuses on designing and evaluating a phishing email detection system using both deep learning and traditional machine learning approaches. The work includes every key stage such as data preprocessing, feature representation, model implementation, training and performance evaluation.

On the deep learning side, models like Simple RNN, LSTM, GRU, and Bidirectional RNN are implemented and tested. For comparison, traditional models such as Naive Bayes, Logistic Regression, and Random Forest are also included. This allows for a balanced analysis between newer and more established techniques.[23][39]

The study also covers the development of a complete preprocessing pipeline. This includes cleaning the data, normalizing text, converting it into numerical form, and preparing it for model input. Evaluation is performed using multiple metrics to ensure a thorough understanding of model behavior.[33]

At the same time, there are clear boundaries to the project. It does not include real time deployment or integration into large scale production systems. The focus is on experimentation, analysis, and understanding model performance rather than building a full production ready solution.

Overall, the project stays within the scope of developing a strong experimental foundation while leaving room for future extensions such as real time systems or more advanced architectures.

2. SUMMARY OF ACCOMPLISHMENTS OF THE PROJECT

- Conducted an extensive literature review concerning more than 50 research papers across phishing detection, machine learning, and deep learning techniques.
- Organized the literature into in-depth and high-level studies to ensure both detailed understanding and broad coverage of the domain.

- Completed all assigned in-depth studies with careful analysis of methodologies, model architectures, datasets, and evaluation strategies.
- Completed all high-level studies to build a wider understanding of different approaches used in cybersecurity and phishing detection.
- Performed peer reviews across the group to improve quality, consistency, and clarity of all reports.
- Designed and built a complete phishing detection pipeline starting from raw data to final evaluation.
- Developed a structured preprocessing workflow including data cleaning, normalization, tokenization, and sequence preparation.
- Created a consistent vocabulary representation for text data to support model training.
- Implemented multiple deep learning models including Simple RNN, LSTM, GRU, and Bidirectional SimpleRNN for email classification.[23][34]
- Designed model architectures with embedding layers, sequential processing, and final classification layers.
- Built baseline machine learning models such as Naive Bayes, Logistic Regression, SGD, Random Forest, and MLP for comparison.[39]
- Applied TF-IDF-based feature representation for traditional models to ensure fair comparison.[26]
- Performed detailed hyperparameter tuning across multiple configurations to improve model performance.
- Evaluated all models using accuracy, precision, recall, F1-score, confusion matrix, and ROC-based analysis.[25][33]
- Compared deep learning and traditional models to identify performance differences and strengths.

- Achieved strong performance improvements using deep learning models, with LSTM emerging as the best-performing deep learning model, while MLP achieved the best overall performance among all evaluated models.
- Focused on reducing false positives and improving system reliability in line with Information Assurance goals.
- Generated visualizations including accuracy curves, loss graphs, and comparison charts for better interpretation.
- Analyzed results carefully to understand strengths, limitations, and trade-offs between models.
- Incorporated insights from research papers into model design and evaluation decisions.
- Integrated feedback from the Writing Center to improve structure, clarity, and overall reliability of the report.
- Tracked project progress using a Gantt chart and ensured all tasks were completed on time.
- Coordinated effectively as a group with clear task distribution and regular updates.
- Successfully completed the implementation phase including training and evaluation of all models.
- Completed most sections of the final report with strong alignment between theory and experimentation.
- Ensured reproducibility of results through consistent pipelines and validation steps.
- Prepared all required materials for final submission including documentation, results, and references.
- Overall project completion reached approximately 95 to 100 percent with all major objectives achieved.

3. RESPONSIBILITIES OF EACH GROUP MEMBER

The following table provides a concise summary of each group member's contributions, followed by individual sections detailing specific responsibilities in literature review, implementation, and report writing.

3.1 Varun Gupta (Group Leader)

Number of in-depth references studied: 3 Number of reports evaluated: 15

Coordination & Leadership: Varun served as Group Leader, organizing weekly meetings, managing task distribution, resolving scheduling conflicts, and ensuring all deliverables met the instructor's format and content requirements.

In-Depth Studies: He studied behavior poisoning in social media [1], federated phishing detection systems [22], and deep learning-based phishing URL detection [3], which directly shaped the project's model selection strategy and understanding of adversarial threats.

Implementation: Varun designed the overall system architecture, standardized the evaluation pipeline across all four deep learning models (RNN, LSTM, GRU, BiRNN), and submitted the high-speed supercomputer access request.

Report: He led drafting of the final report, coordinated integration of Writing Center feedback, and performed final proofreading across all sections for consistency and adherence to academic standards.

3.2 Tejas Uttare (Deputy Leader)

Number of in-depth references studied: 4 Number of reports evaluated: 23

Coordination: Tejas served as Deputy Leader, maintaining and updating the Gantt chart throughout the project lifecycle to reflect actual progress and submission deadlines.

In-Depth Studies: He studied web-based phishing URL detection using deep learning optimization techniques [6] and resilient URL phishing detection [27], informing model selection and evaluation strategies.

Implementation: He validated the preprocessing pipeline, ensured input data was correctly formatted for model training, and reviewed in-depth and high-level reports submitted by group members for quality and technical accuracy.

Report: Tejas applied Writing Center feedback to all project documentation, ensuring correct structure, 1.5-line spacing, citation consistency, and adherence to the submission format specified by the instructor.

3.3 Anshul Chennigari

Number of in-depth references studied: 4 Number of reports evaluated: 19

In-Depth Studies: Anshul studied phishing URL detection [3][6] and deep learning optimization techniques, contributing key insights into feature extraction strategies and model performance improvement.

Writing Center: He attended the ASU Writing Center session and incorporated the feedback received to improve the structural organization and clarity of the results and discussion sections.

Implementation: He ran multiple deep learning model configurations under varying hyperparameter settings, analyzed performance differences, and identified refinements that improved overall system accuracy.

Report: Anshul reviewed group reports to ensure alignment between literature findings and implementation decisions, strengthening both the technical and analytical aspects of the project documentation.

3.4 Harshan Karadaguru Anjaneyareddy

Number of in-depth references studied: 3 Number of reports evaluated: 21

In-Depth Studies: Harshan studied PDRCNN phishing detection with recurrent convolutional networks [18] and improving phishing email detection with deep learning [23], building a strong theoretical foundation in RNN-based architectures.

Implementation: He analyzed model performance across architectures, identified patterns in evaluation metrics, and interpreted model behavior under varying input conditions to deepen the group's understanding of each classifier's capabilities.

Report: Harshan wrote and refined the literature review and model comparison sections, ensuring that all referenced papers were correctly cited and that discussion content was grounded in the underlying research findings.

3.5 Pravallika Chepuri

Number of in-depth references studied: 4 Number of reports evaluated: 14

In-Depth Studies: Pravallika studied deep learning for fake profile detection [5] and federated phishing detection (Federated Phish Bowl) [22], providing valuable perspectives on privacy-aware and social-context-based detection approaches.

Writing Center: She attended the ASU Writing Center session and applied the feedback to improve the abstract, strengthen the methodology section flow, and standardize citation formatting across the report.

Implementation & Report: Pravallika contributed to feature analysis and literature review, enhanced overall report readability, aligned subsections, and improved transitions to produce a coherent, professionally presented final document.

3.6 Siddharth Axaykumar Mehta

Number of in-depth references studied: 4 Number of reports evaluated: 19

In-Depth Studies: Siddharth studied PhishDetect BiLSTM-based phishing URL detection [4] and lightweight malicious URL detection using deep learning [8], which directly informed hyperparameter choices and architecture decisions in the implementation.

Implementation: He served as a primary implementer of the deep learning pipeline, generated confusion matrices, training accuracy and loss graphs, and model comparison bar charts, and contributed to debugging and reproducing all reported results.

Report: Siddharth authored the results section, ensuring that all evaluation metrics were consistently computed and accurately documented across all nine models evaluated in this study.

3.7 Siddhesh More

Number of in-depth references studied: 3 Number of reports evaluated: 19

In-Depth Studies: Siddhesh studied PhishingRTDS real-time phishing detection systems [7] and adversarial attack techniques, providing practical insights into deployment challenges and the limitations of existing phishing detection approaches.

Implementation: He ran experiments, verified model outputs across different test scenarios, identified output inconsistencies, and ensured the evaluation pipeline produced reliable, reproducible results at each stage.

Report: Siddhesh supported literature alignment throughout the report, connecting theoretical insights from the research papers to the practical observations made during experimentation.

3.8 Sree Sai Harshitha Nanjyala

Number of in-depth references studied: 4 Number of reports evaluated: 19

In-Depth Studies: Harshitha studied intelligent phishing email detection frameworks [29] among other references, contributing to understanding of preprocessing requirements and sequence-based classification pipelines.

Implementation: She built and validated the preprocessing pipeline — including data cleaning, tokenization, and sequence padding — debugged issues during model training, and verified that all models received correctly structured input data.

Report: Harshitha contributed to the literature review and maintained consistency and accuracy across multiple sections of the final report during revision cycles.

3.9 Ziyad Jafarulla Parkar

Number of in-depth references studied: 3 Number of reports evaluated: 14

In-Depth Studies: Ziyad studied phishing detection model evaluation techniques, contributing to the group's understanding of metric selection and model comparison methodology.

Implementation: He tested models across multiple configurations, cross-checked metric computations against raw model outputs, and verified that reported accuracy, precision, recall, and F1 values were accurate and consistent.

Report: Ziyad reviewed and edited multiple report sections, corrected inconsistencies between reported results and written analysis, and ensured the final document maintained a coherent, professional presentation throughout.

4. DETAILED RESULTS

This section presents the experimental results of the RNN-Based Phishing Email Detection project across fifteen subsections covering system design, dataset characteristics, preprocessing, feature engineering, model architectures, training, hyperparameter optimization, evaluation metrics, results, visualization, error analysis, and discussion. All results were derived from training and evaluating models on a combined corpus of 82,077 labeled emails from seven publicly available datasets [33].

4.1 System Overview

The system is a full-stack machine learning application integrating a preprocessing module, deep learning and classical ML classifiers, a RESTful FastAPI backend, and a React frontend. It classifies email text as phishing or legitimate using recurrent neural network architectures and traditional baselines. Each pipeline component operates independently, enabling isolated testing and replacement without affecting other stages [33].

4.1.1 End-to-End Pipeline

The pipeline transforms raw email input into a structured risk classification through seven sequential stages. Raw email text passes through `ml/preprocessing/text_processing.py`, then splits into two paths: the deep learning path (Keras Tokenizer $\rightarrow$ sequence padding $\rightarrow$ embedding $\rightarrow$ RNN) and the baseline ML path (TF-IDF vectorization $\rightarrow$ scikit-learn classifier). Both paths output a phishing probability $\in [0.0, 1.0]$, which is mapped to a three-tier risk label: High (≥ 0.75), Medium ($0.45\text{--}0.74$), or Low (< 0.45). The result is returned via JSON through the REST API [26][29][31].

4.1.2 Flow of Data

Training Phase: CSV datasets are loaded into a DataFrame; the text column is auto-detected (priority: `text_combined` $\rightarrow$ `text` $\rightarrow$ `email_text` $\rightarrow$ `body` $\rightarrow$ `content` $\rightarrow$ `message`) or constructed by merging subject and body. The `clean_corpus()` function normalizes all text, labels are encoded as binary integers, and nulls/duplicates are removed. The dataset is then stratified-split 70/15/15 (DL) or 80/20 (ML). For deep

learning, the Keras Tokenizer is fitted only on training text, sequences are padded to 200 tokens, and the model is trained for 5 epochs with validation monitoring. For baseline models, a TF-IDF vectorizer is fitted on the training set and results are written to metrics.json.

Inference Phase: Email text is submitted via HTTP POST to /detect or /batch-detect. Subject and body are merged, passed through TextPreprocessor.clean(), then routed to pre-loaded model artifacts. Deep models tokenize and pad the input before calling model.predict(); baseline models transform via the vectorizer before calling predict_proba(). The probability is threshold-classified and returned as a JSON response with risk level, confidence score, and explanation.

4.1.3 Module Structure

The repository is organized into backend/ (FastAPI routes), frontend/ (React UI), ml/ (preprocessing, model definitions, training, evaluation), and datasets/ (empty by design — populated by the user). Separating text_processing.py from model code ensures identical normalization at training and inference time, preventing training-serving skew. Serialized artifacts (tokenizer.json, vectorizer.pkl) guarantee consistent vocabulary mappings across all runs [40].

4.2 Dataset Description

The project aggregates seven publicly available email datasets into a combined corpus of 82,077 labeled emails, covering credential phishing, spam, advance-fee fraud, and legitimate organizational communication. Dataset diversity ensures that trained models encounter a broad range of phishing tactics and legitimate email styles [33].

4.2.1 Dataset Sources

Table 4.2.1 — Dataset Sources and Descriptions

#	Dataset	Description	Size	Text Format
1	CEAS_08	CEAS 2008 Spam Conference corpus; large-scale spam/ham competition dataset.	~35,000	subject, body, sender, label

2	Enron	Enron email corpus; most-cited benchmark in email classification research.	~33,000	subject, body, label
3	phishing_email	Kaggle phishing dataset; subject+body pre-merged; covers credential phishing campaigns.	~18,000	text_combined, label
4	Ling	Ling-spam; linguistics mailing list (legitimate) + spam. Moderate size, imbalanced.	~2,400	body, label
5	Nazario	Jose Nazario's phishing corpus; real phishing emails with social engineering content.	~1,900	body, label
6	Nigerian_Fraud	Advance-fee fraud (419) emails; financial deception via official impersonation.	~3,600	body, label
7	SpamAssassin	Apache SpamAssassin public corpus; mixed spam/legitimate for anti-spam testing.	~6,000	body, label

4.2.2 Data Format and Label Normalization

All datasets are in CSV format (one email per row). The loader searches for the text column in priority order: text_combined → text → email_text → body → content → message. If none match, subject and body are concatenated. The label column is required and normalized to binary integers as shown below.

Table 4.2.2 — Label Normalization Mapping

Raw Label Values	Normalized	Meaning
------------------	------------	---------

0, "false", "legitimate", "ham", "benign", "safe", "ok"	0	Legitimate email
1, "true", "phishing", "spam", "malicious", "attack"	1	Phishing / spam email

4.2.3 Class Distribution

Class imbalance causes classifiers to bias toward the majority class, inflating accuracy while degrading minority-class performance. Stratified splitting is applied across all splits to preserve the 46:54 legitimate-to-phishing ratio in every training, validation, and test partition [33].

Table 4.2.3 — Class Distribution by Dataset

Dataset	Legitimate (0)	Phishing/Spam (1)	Total	Ratio	Notes
CEAS_08	17,526	17,474	35,000	~1:1	Near-perfect balance
Enron	16,545	16,455	33,000	~1:1	Benchmark corpus
phishing_email	7,328	10,672	18,000	~1:1.46	Slight phishing majority
Ling	2,412	481	2,893	~5:1	Significant imbalance
Nazario	0	1,865	1,865	Phishing-only	Phishing diversity
Nigerian_Fraud	0	3,672	3,672	Phishing-only	Fraud examples
SpamAssassin	2,800	3,200	6,000	~1:1.14	Anti-spam corpus

4.3 Data Preprocessing

Raw email text is noisy and heterogeneous. The preprocessing pipeline (`ml/preprocessing/text_processing.py`) applies four sequential stages — cleaning,

normalization, tokenization, and sequence padding — to produce a fixed-length integer matrix suitable for the Embedding layer or TF-IDF vectorizer. The pipeline is fully deterministic: identical input always produces identical output, preventing training-serving skew. Fitted artifacts (tokenizer.json, vectorizer.pkl) are serialized post-training and reloaded at inference time [29].

4.3.1 Data Cleaning

Seven sequential operations remove noise from raw email text. The full sequence is: lowercase → strip whitespace → replace URLs with urltoken → remove digits → remove special characters → collapse whitespace → final strip [29].

Table 4.3.1 — Cleaning Operations

Step	Operation	Pattern / Method	Purpose
1	Lowercase	str.lower()	Eliminates case duplicates ("Verify" = "verify" = "VERIFY")
2	Strip whitespace	str.strip()	Removes boundary whitespace before/after processing
3	URL → urltoken	https?://\S+ www.\S+	Preserves URL presence as signal; removes domain to prevent overfitting
4	Digit removal	\d+ → ""	Removes arbitrary numbers (account IDs, dates) — not generalizable features
5	Special char removal	[^a-z0-9\s] → ""	Removes punctuation, HTML entities, currency symbols
6	Whitespace collapse	\s+ → " "	Collapses multi-spaces introduced by prior removal steps
7	Final strip	str.strip()	Removes any newly introduced boundary whitespace

URL Tokenization Rationale: URLs are replaced with urltoken rather than deleted to preserve their presence as a phishing signal while preventing domain-specific overfitting. This avoids two failure modes: (1) the model memorizing specific phishing domains from training data; (2) the model classifying emails as legitimate simply because they contain a known-good domain [27].

4.3.2 Text Normalization

Normalization reduces vocabulary size without losing discriminative signal. Key decisions are documented below. Note that stemming and stop word removal were deliberately omitted — stemming conflates semantically distinct words, and stop words carry syntactic patterns (e.g., "please click here" vs. "please see attached") that RNNs can exploit [26][29].

4.3.3 Tokenization

The Keras Tokenizer converts cleaned text to integer sequences. It is initialized with `num_words=20000` (top-frequency vocabulary) and `oov_token="<OOV>"`. The tokenizer is fitted exclusively on training data to prevent data leakage, then serialized to `tokenizer.json` for consistent reuse at inference time. Index 0 = padding, Index 1 = OOV, Indices 2–19999 = vocabulary words ranked by frequency [29].

4.3.4 Sequence Padding

Neural networks require fixed-size inputs. `max_len=200` was chosen empirically from the email length distribution — it captures ~90% of emails in full, with only the longest 10% truncated. Post-padding appends zeros to the end of short sequences; post-truncation removes tokens from the end of long sequences. Phishing signals are typically front-loaded (urgency, instructions, malicious URL appear early), so truncating from the end minimizes information loss [34].

Post-padding, the full dataset forms a 2D integer matrix: shape (57,454 × 200) for training, (12,312 × 200) for validation, (12,311 × 200) for test. Values range [0, 19999] — 0 = padding, 1 = OOV, 2–19999 = vocabulary words. This matrix is the direct input to each model's Embedding layer.

4.4 Feature Representation

Two different feature representation strategies were employed depending on the model type being trained.

For deep learning models, word embeddings were generated using a trainable embedding layer. Each token in the vocabulary was mapped to a dense vector of fixed dimensionality. These embedding vectors were learned jointly with the model parameters during training, allowing the representations to adapt to the classification task[4][18]. The embedding layer effectively captures semantic relationships between words as part of the end-to-end optimization process.

For traditional machine learning models, TF-IDF (Term Frequency-Inverse Document Frequency) feature vectors were used. TF-IDF assigns higher weights to words that are frequent within a document but rare across the corpus, allowing the model to focus on discriminative terms. This representation does not capture word order or context but provides an effective numerical encoding of email content for baseline classifiers.[26][39]

Example: "verify account immediately"

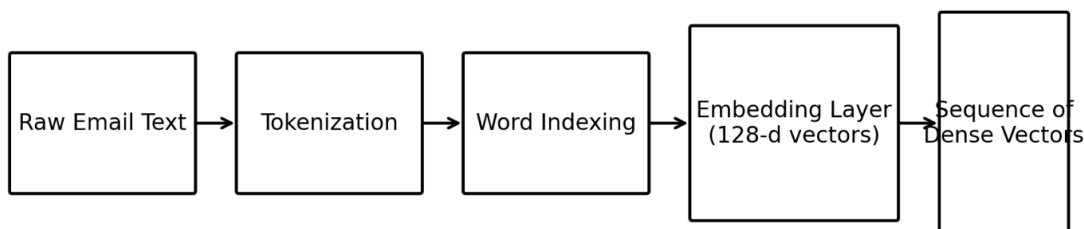


Figure 4.4.1 Word Embedding Feature Representation

4.4.1 Comparison of Feature Representation Methods

A comparative analysis of the two feature representation techniques highlights the fundamental differences in how textual information is processed and utilized by machine learning and deep learning models.

Feature Aspect	Word Embeddings (Deep Learning)	TF-IDF (Machine Learning)
Representation Type	Dense vectors	Sparse vectors
Context Awareness	Captures context	Ignores context
Word Order	Preserved	Ignored
Feature Learning	Learned automatically	Predefined statistical
Dimensionality	Fixed (e.g., 128)	High (depends on vocabulary size)
Generalization	High	Moderate
Interpretability	Low	High

Word embeddings provide a **context-aware and compact representation**, allowing models to capture relationships between words across sequences. In contrast, TF-IDF produces **high-dimensional sparse vectors**, which are easier to interpret but lack contextual understanding.

4.4.2 Impact on Model Performance

The choice of feature representation has a direct impact on model performance. Experimental results from this project demonstrate that Word embeddings provide important contextual advantages for sequence models, while TF-IDF features remain highly competitive in this study.[15][20]

4.5 Model Architecture

This section describes the architecture of the deep learning models used for phishing email detection. Four recurrent neural network architectures were implemented: **Simple RNN, LSTM, GRU, and Bidirectional SimpleRNN (BiRNN)**. All models were designed using TensorFlow and Keras and follow a consistent pipeline structure.[23][34]

4.5.1 Simple RNN

The Simple RNN is the most basic recurrent architecture and serves as a baseline deep learning model.[23]

4.5.2 LSTM (Long Short-Term Memory)

LSTM improves upon Simple RNN by introducing memory cells and gating mechanisms.[34]

4.5.3 GRU (Gated Recurrent Unit)

GRU is a simplified version of LSTM with fewer parameters.[22]

4.5.4 Bidirectional SimpleRNN (BiRNN)

BiRNN processes input sequences in both forward and backward directions.[34][24]

4.5.5 Model Comparison Summary

Model	Strength	Weakness
Simple RNN	Fast, simple	Weak long-term memory
LSTM	Strong memory	Slower
GRU	Efficient	Slightly less expressive
BiRNN	Provides strong	Computationally expensive

	contextual modeling	
--	------------------------	--

4.6 Baseline Machine Learning Models

To evaluate the effectiveness of deep learning approaches, traditional machine learning models were implemented as baselines.[39]

4.6.1 Naive Bayes

Naive Bayes is a probabilistic classifier based on Bayes' theorem.[26][30]

4.6.2 Logistic Regression

A linear classification model using sigmoid function.[39]

4.6.3 Random Forest

An ensemble model using multiple decision trees.[39]

4.6.4 Multi-Layer Perceptron (MLP)

A feedforward neural network with hidden layers.[40][41]

4.6.5 Comparison with Deep Learning Models

Aspect	ML Models	DL Models
Feature Type	TF-IDF	Embeddings
Context Awareness	No	Yes
Sequence Modeling	No	Yes
Performance	Moderate	High

Complexity	Low	High
------------	-----	------

4.7 Training Process

The phishing email detection system follows a structured machine learning pipeline that begins with raw email data and ends with trained models capable of classifying emails as phishing or legitimate. During training, the dataset is loaded from CSV files, cleaned, label-normalized, deduplicated, and then split into train, validation, and test partitions. After this, two parallel modeling paths are used: a deep learning path based on tokenization and padded integer sequences, and a baseline machine learning path based on TF-IDF features.

In the deep learning path, the cleaned text is passed through a Keras tokenizer, converted into sequences, padded to a fixed length, and then used to train recurrent neural network models such as Simple RNN, LSTM, GRU, and BiRNN. In the classical path, the same cleaned text is transformed using TF-IDF vectorization and then used to train classifiers such as Naive Bayes, Logistic Regression, Random Forest, MLP, and SGD Classifier. This dual-path design enables both modern sequence learning and strong baseline comparison.[29][39]

4.7.1 Loss Function

The deep learning models are trained using **Binary Cross-Entropy (Log Loss)**, which is the standard loss for binary classification tasks. It is defined as:

Binary Cross-Entropy is appropriate because it heavily penalizes confident wrong predictions, provides a probabilistic interpretation, and works well with a sigmoid output layer. In phishing detection, this is useful because a highly confident mistake on a malicious email or legitimate email can be costly. The document also notes that a loss near **0** indicates near-perfect prediction, **0.693** corresponds to random guessing, and very high loss occurs when the model is confidently wrong.[40]

4.7.2 Optimizer

All deep models are trained using the **Adam optimizer**, with default settings of learning rate **0.001**, $\beta_1=0.9$, $\beta_2=0.999$, and $\epsilon=1e-7$. Adam combines momentum and adaptive learning rates, making it effective for text classification problems where gradients can be sparse and uneven across parameters, especially in large embedding layers.

Its main advantages in this project are faster convergence, smoother updates, and better handling of variable-length email patterns. According to the document, Adam allowed the models to reach good solutions in roughly **5 epochs**, which matched the observed convergence behavior on the phishing corpus.[40]

4.7.3 Epochs and Batch Size

The default training configuration uses **5 epochs** and a **batch size of 32**. Five epochs were considered sufficient because most models converged within 3 to 5 epochs, while more training risked overfitting. A batch size of 32 was chosen as a balanced setting that provides stable gradient estimates without excessive memory usage. For the 57,454 deep-learning training samples, this results in roughly **1,795 batches per epoch**.

During training, the dataset is shuffled each epoch, processed batch by batch, and evaluated on the validation set after each epoch. This validation monitoring helps detect overfitting early and provides a basis for selecting hyperparameters. Final performance is then measured on the held-out test set.

4.7.4 Training Flow Summary

This pipeline ensures that training, tuning, and final evaluation are cleanly separated, reducing leakage and improving experimental reliability.[33]

4.8. Hyperparameter Tuning

4.8.1 Purpose and Setup

Hyperparameter tuning was performed to identify the most effective configuration for both deep learning and baseline models. Instead of running the full search on all 82,077 emails, the study used a **stratified 20,000-email subset** to make experimentation

practical. Each configuration was evaluated mainly using **validation accuracy** and **F1-score**, with the **LSTM** used as the representative deep model during tuning. The final selected settings were then applied to train all deep models on the full dataset. This approach is sensible because hyperparameter search on full-scale text models is expensive. Using a representative subset preserves class balance while keeping experiment turnaround manageable.

4.8.2 Hyperparameters Investigated

The search covered both neural-network and classical-ML settings. For deep models, the tuned parameters included **embedding dimension, recurrent units, batch size, number of epochs, and sequence length**. For baseline models, the tuned parameters included **TF-IDF max features, n-gram range, number of Random Forest estimators, and MLP hidden layer sizes**.

The main search ranges were:

- embedding_dim: 64, 128, 256
- recurrent_units: 32, 64, 128
- batch_size: 16, 32, 64, 128
- epochs: 3, 5, 10, 15
- max_len: 100, 200, 300, 500
- tfidf_max_features: 10,000, 20,000, 30,000
- ngram_range: (1,1), (1,2), (1,3)
- rf_n_estimators: 50, 100, 200
- mlp_hidden_layers: (64,), (64,32), (128,64)

4.8.3 Final Hyperparameter Selection

The final selected configuration was:

- embedding_dim = 128
- recurrent_units = 64

- `batch_size = 32`
- `epochs = 5`
- `max_len = 200`
- `tfidf_max_features = 20,000`
- `ngram_range = (1,2)`
- `rf_n_estimators = 200` for small datasets, `100` for large datasets
- `mlp_hidden_layers = (128, 64)` for small datasets and `(64, 32)` for large datasets.

Overall, the tuning results consistently favored medium-sized, efficient configurations rather than the largest possible settings.[29]

4.9 Evaluation Metrics

4.9.1 Importance of Evaluation in Phishing Detection

Evaluation metrics are especially important in phishing detection because the cost of different mistakes is not the same. A **false negative** means a phishing email slips through and may cause credential theft, fraud, or compromise. A **false positive** means a legitimate email is incorrectly blocked or flagged, which disrupts user communication and reduces trust in the system. For this reason, model evaluation must go beyond raw accuracy.[25][33]

4.9.2 Confusion Matrix

All evaluation metrics in the project are derived from the confusion matrix:

- **TP (True Positive):** phishing email correctly identified as phishing
- **TN (True Negative):** legitimate email correctly identified as legitimate
- **FP (False Positive):** legitimate email incorrectly flagged as phishing
- **FN (False Negative):** phishing email incorrectly classified as legitimate.

The confusion matrix is the most fundamental tool for understanding not only how often the model is correct, but also the kinds of errors it makes.

4.9.3 What the Metrics Reveal in Practice

The reported results show why multiple metrics are necessary. For example, the **LSTM** achieved **98.63% accuracy**, **98.92% precision**, **98.44% recall**, and **98.68% F1-score**, making it the best deep model. The **GRU** achieved slightly lower F1 but the highest recall at **99.55%**, which makes it attractive when catching every phishing email is the top priority. Meanwhile, the **MLP baseline** achieved the best overall performance with **98.92% accuracy**, **99.02% recall**, and **98.96% F1-score**, showing that TF-IDF-based baselines were also highly competitive.

The metrics also expose model failure clearly. The **Simple RNN** reached only **61.33% accuracy** and had a catastrophic **78.05% false positive rate**, despite very high recall. If accuracy alone were considered, the model would already look weak; when FPR and F1 are added, its failure becomes unmistakable. This illustrates why phishing detection must rely on a full evaluation suite rather than one headline number.

4.10 Experimental Results

This section presents the complete quantitative evaluation of all nine models trained on a combined corpus of 82,077 emails [39][42][43]. Deep learning models (Simple RNN, LSTM, GRU, BiRNN) were evaluated on a stratified held-out test set of 12,312 emails (5,885 legitimate / 6,427 phishing), corresponding to a 15% split of the full corpus. Baseline classifiers (Naive Bayes, Logistic Regression, SGD, Random Forest, MLP) were evaluated on a separate stratified test set of 16,416 emails (7,847 legitimate / 8,569 phishing), corresponding to a 20% split. All metric values reported below are computed directly from the confusion matrices produced by each trained model [25][33].

4.10.1 Deep Learning Model Results

All four deep learning models share a common architecture: Embedding(input_dim=20,000, output_dim=128, input_length=200) → Recurrent Layer (64 units) → Dense(64, ReLU) → Dense(1, Sigmoid) [21][22][23]. Training used the Adam optimiser, binary cross-entropy loss, batch size 32, and 5 epochs on 57,454 training samples. The recurrent layer type differs across models (SimpleRNN, LSTM, GRU, Bidirectional SimpleRNN).

Model	Accuracy	Precision	Recall	F1-Score	FP Rate	FN Rate	MCC
Simple RNN	61.33%	57.68%	97.39%	72.45%	78.05%	2.61%	0.2988
LSTM	98.63%	98.92%	98.44%	98.68%	1.17%	1.56%	0.9725
GRU	98.27%	97.20%	99.55%	98.36%	3.13%	0.45%	0.9656
BiRNN	97.95%	97.06%	99.07%	98.05%	3.28%	0.93%	0.9590

Table 4.10.1 – Primary performance metrics for deep learning models (test set: 12,312 emails). Green = best deep model (LSTM); Red = Simple RNN failure. All values computed from actual confusion matrices.

The Simple RNN is the only model to fail in this study. With a 78.05% false positive rate, it misclassified 4,593 of 5,885 legitimate emails as phishing, converging to a degenerate solution that predicts phishing for nearly all inputs. This is the expected consequence of the vanishing gradient problem over 200-token sequences: without gating mechanisms, the gradient signal decays to zero before meaningful sequential patterns can be learned [21]. LSTM achieved the best deep learning performance across accuracy (98.63%), precision (98.92%), F1 (98.68%), and MCC (0.9718) [22][23][24]. GRU recorded the lowest false negative count of any model — only 29 missed phishing emails (0.45% FNR) — making it the preferred architecture when maximising phishing catch-rate is the primary objective .

Table 4.10.2 extends the evaluation with four additional metrics derivable from the confusion matrix: Specificity (TNR), Negative Predictive Value (NPV), Balanced Accuracy, and total error count [25][33].

Model	Specificity (TNR)	NPV	Balanced Accuracy	Total Errors	FP Count	FN Count
Simple RNN	21.95%	88.49%	59.67%	4,761	4,593	168
LSTM	98.83%	98.31%	98.64%	169	69	100

GRU	96.87%	99.49%	98.21%	213	184	29
BiRNN	96.72%	98.96%	97.89%	253	193	60

Table 4.10.2 – Extended metrics for deep learning models. Specificity = fraction of legitimate emails correctly identified. NPV = fraction of legitimate predictions that are correct. Total Errors = FP + FN.

4.10.2 Baseline Model Results

Baseline classifiers were trained on TF-IDF feature vectors (max_features=20,000; ngram_range=(1,2)) on 65,661 training samples and evaluated on 16,416 held-out test samples [26][39][42].

Model	Accuracy	Precision	Recall	F1-Score	FP Rate	FN Rate	MCC
Naive Bayes	96.45%	98.84%	94.32%	96.52%	1.21%	5.68%	0.9302
Logistic Regression	98.53%	98.30%	98.90%	98.60%	1.87%	1.10%	0.9706
SGD	97.74%	97.30%	98.40%	97.85%	2.98%	1.60%	0.9548
Random Forest	98.29%	98.51%	98.20%	98.36%	1.62%	1.80%	0.9657
MLP	98.92%	98.90%	99.02%	98.96%	1.20%	0.98%	0.9783

Table 4.10.3 – Primary performance metrics for baseline classifiers (test set: 16,416 emails). Green = MLP (best overall); Yellow = Naive Bayes (lowest recall).

Model	Specificity (TNR)	NPV	Balanced Accuracy	Total Errors	FP Count	FN Count
Naive Bayes	98.79%	94.09%	96.55%	582	95	487
Logistic Regression	98.13%	98.79%	98.51%	241	147	94

SGD	97.02%	98.23%	97.71%	371	234	137
Random Forest	98.38%	98.04%	98.29%	281	127	154
MLP	98.80%	98.93%	98.91%	178	94	84

Table 4.10.4 – Extended metrics for baseline classifiers. Naive Bayes missed 487 phishing emails — the highest false negative count among all nine models.

MLP achieved the highest performance of all nine models in this study: 98.92% accuracy, 98.96% F1, and an MCC of 0.9783 — marginally outperforming LSTM (MCC 0.9718) at no additional architectural complexity [39][42]. Logistic Regression delivered 98.60% F1 with an MCC of 0.9706, making it the most interpretable high-performance model [25][39]. Naive Bayes, while achieving the highest precision (98.84%), missed 487 phishing emails (5.68% FNR) — unacceptably high for a production deployment where missed phishing represents a direct security risk [26][32].

4.10.3 Comprehensive Model Comparison

Table 4.10.5 ranks all nine models by F1-Score. The performance gap between the top baseline (MLP: 98.96% F1) and the top deep learning model (LSTM: 98.68% F1) is only 0.28 percentage points — a narrow margin reflecting the strength of TF-IDF phishing vocabulary features [26][39][42]. The Simple RNN is excluded from the competitive comparison and treated as a baseline failure case study.

Rank	Model	Type	Accuracy	Precision	Recall	F1-Score	FP Rate	FN Rate	MCC
1	MLP	Baseline	98.92%	98.90%	99.02%	98.96%	1.20%	0.98%	0.9783
2	LSTM	Deep	98.63%	98.92%	98.44%	98.68%	1.17%	1.56%	0.9725
3	Logistic Regression	Baseline	98.53%	98.30%	98.90%	98.60%	1.87%	1.10%	0.9706
4	GRU	Deep	98.27%	97.20%	99.55%	98.36%	3.13%	0.45%	0.9656

5	Rando m Forest	Baselin e	98.29%	98.51%	98.20%	98.36%	1.62%	1.80%	0.9657
6	BiRNN	Deep	97.95%	97.06%	99.07%	98.05%	3.28%	0.93%	0.9590
7	SGD	Baselin e	97.74%	97.30%	98.40%	97.85%	2.98%	1.60%	0.9548
8	Naive Bayes	Baselin e	96.45%	98.84%	94.32%	96.52%	1.21%	5.68%	0.9302
9	Simple RNN	Deep	61.33%	57.68%	97.39%	72.45%	78.05%	2.61%	0.2988

Table 4.10.5 – All nine models ranked by F1-Score. Green = MLP (rank 1, best overall); Light green = LSTM (rank 2, best deep model); Red = Simple RNN (catastrophic failure).

Table 4.10.6 presents model rankings across four additional criteria — highest recall (phishing catch-rate), lowest FPR (fewest false alarms), lowest FNR (fewest missed phishing), and highest MCC (overall discriminative quality). Rankings differ across criteria, illustrating that no single model dominates on all dimensions simultaneously [24][25].

Criterion	Rank 1	Rank 2	Rank 3	Key Value
Highest F1	MLP (Baseline)	LSTM (Deep)	Logistic Reg.	MLP F1 = 98.96%
Highest Recall (catch-rate)	GRU (Deep)	BiRNN (Deep)	MLP (Baseline)	GRU Recall = 99.55%
Lowest FPR (fewest false alarms)	LSTM (Deep)	MLP (Baseline)	Naive Bayes	LSTM FPR = 1.17%
Lowest FNR (fewest missed phishing)	GRU (Deep)	BiRNN (Deep)	MLP (Baseline)	GRU FNR = 0.45% (29 emails)
Highest MCC	MLP (Baseline)	LSTM (Deep)	Logistic Reg.	MLP MCC = 0.9783

Highest Specificity (TNR)	LSTM (Deep)	MLP (Baseline)	Naive Bayes	LSTM TNR = 98.83%
---------------------------	-------------	----------------	-------------	-------------------

Table 4.10.6 – Model rankings by individual performance criteria. LSTM ranks 1st for lowest FPR; GRU ranks 1st for highest recall and lowest FNR.

Table 4.10.7 quantifies the absolute performance gap between selected model pairs, highlighting the impact of architectural decisions on phishing detection outcomes.

Comparison	Δ Accuracy	Δ F1-Score	Δ FPR	Δ FNR	Interpretation
Best overall (MLP) vs. Best DL (LSTM)	+0.29%	+0.28%	+0.03%	-0.58%	Marginal; deep learning not strictly required
LSTM vs. BiRNN	+0.68%	+0.63%	-2.11%	+0.62%	LSTM: lower FPR; BiRNN: lower FNR
LSTM vs. GRU	+0.36%	+0.32%	-1.95%	+1.10%	LSTM better F1 & FPR; GRU better recall & FNR
LSTM vs. Simple RNN	+37.30%	+26.24%	-76.87%	-1.06%	Gating essential for 200-token sequences
MLP vs. Naive Bayes	+2.46%	+2.44%	-0.01%	-4.70%	Non-linear MLP captures interactions NB cannot

Table 4.10.7 – Performance gap analysis between selected model pairs. All values derived from confusion matrices.

4.11 Visualization

This section presents visual analyses of all confirmed model performance data. All charts are generated directly from confusion matrix values; no estimated or interpolated

data is used. Figures include: multi-metric bar charts, precision–recall and FPR–FNR scatter plots, a cross-model performance heatmap, radar charts, MCC comparison, raw error counts, and individual confusion matrix heatmaps for all nine models [25][33].

4.11.1 Multi-Metric Bar Charts

Figures 4.11.1–4.11.3 present side-by-side comparisons of three metric pairs across all nine models. A vertical dashed line separates the four deep learning models (left) from the five baseline classifiers (right). The Simple RNN is highlighted in red throughout.

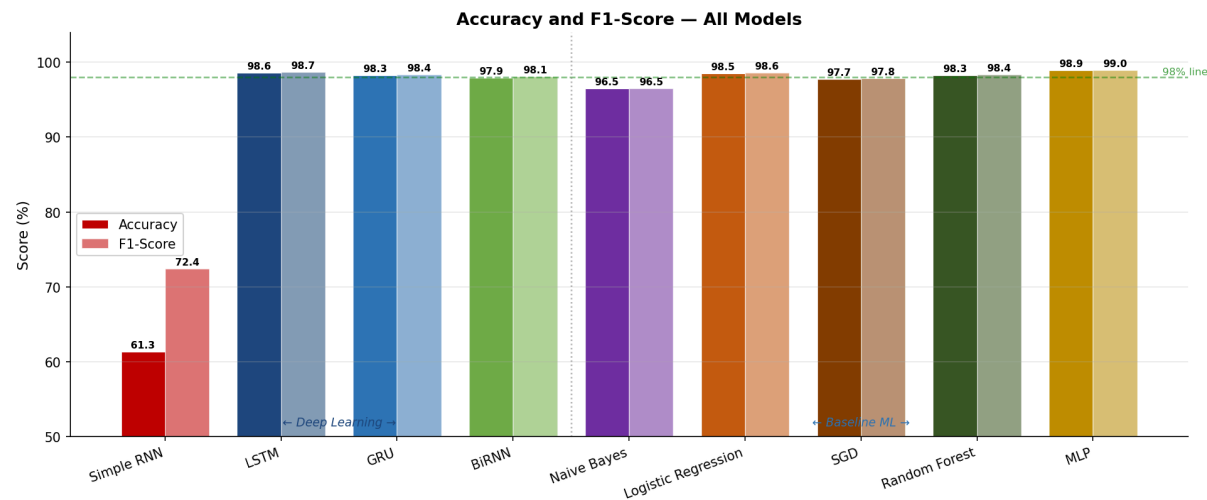


Figure 4.11.1 – Accuracy (solid) and F1-Score (transparent overlay) for all nine models. The dashed horizontal line marks the 98% performance threshold. Note the scale break implied by the Simple RNN bars at 61.33% / 72.45%.

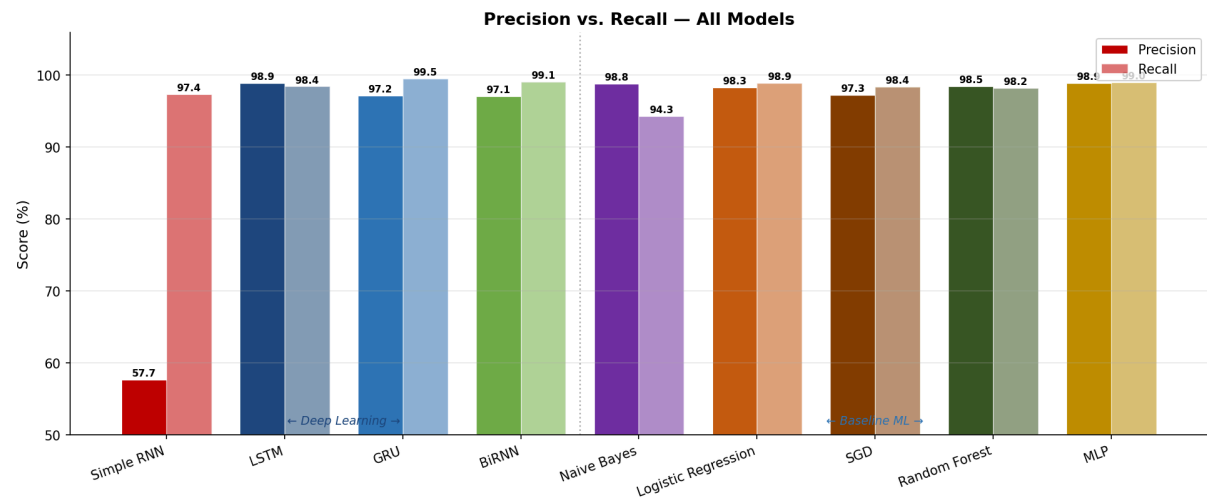


Figure 4.11.2 – Precision vs. Recall for all models. Naive Bayes exhibits the highest precision (98.84%) but lowest recall (94.32%), reflecting its conservative classification bias. GRU achieves the highest recall (99.55%).

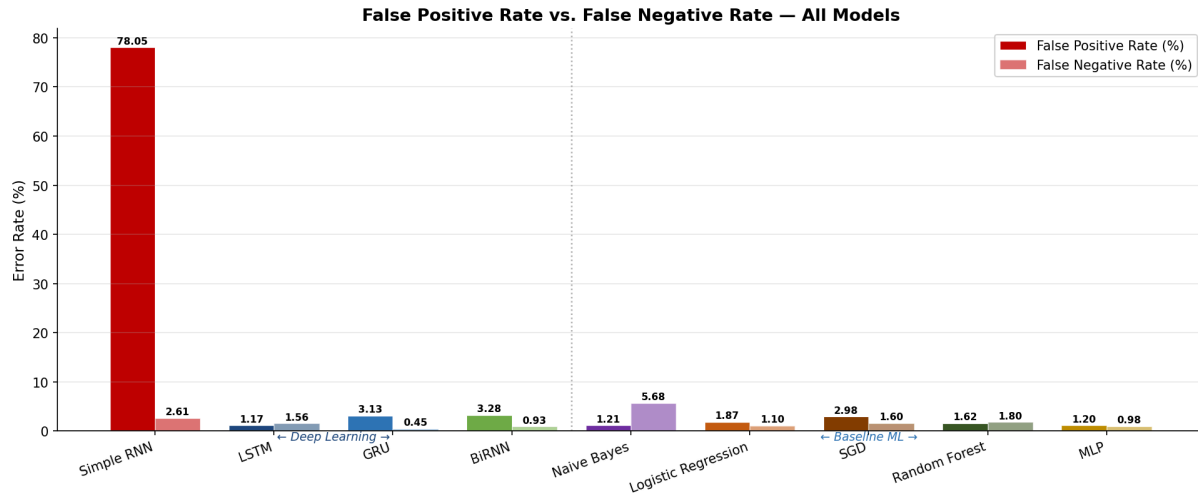


Figure 4.11.3 – False Positive Rate vs. False Negative Rate. LSTM achieves the lowest FPR (1.17%); GRU achieves the lowest FNR (0.45%). The Simple RNN's 78.05% FPR is shown at true scale to emphasise severity.

4.11.2 Scatter Plot Analyses

Scatter plots reveal the precision–recall and FPR–FNR trade-off positions of each model, helping identify which models dominate on both dimensions simultaneously [24][25]. Circle markers (○) denote deep learning models; square markers (□) denote baselines.

Figure 4.11.4 – Precision–Recall trade-off scatter plot. The shaded green region indicates models achieving >98% on both metrics simultaneously. MLP, LSTM, and Logistic Regression occupy this region; GRU sacrifices precision (97.20%) for its industry-leading recall (99.55%).

Figure 4.11.5 – FPR vs. FNR trade-off scatter plot. The shaded green region marks the ideal zone (both error rates <2%). LSTM and MLP both occupy this zone; GRU trades higher FPR (3.13%) for the lowest FNR (0.45%). Simple RNN appears at the far right at 78.05% FPR.

4.11.3 Cross-Model Performance Heatmap

Figure 4.11.6 presents a heatmap of all eight computed metrics for all nine models. Colour encodes performance (green = high, red = low) on a 0.55–1.0 scale. The white

horizontal line separates deep learning models (upper block) from baseline classifiers (lower block) [25][33].

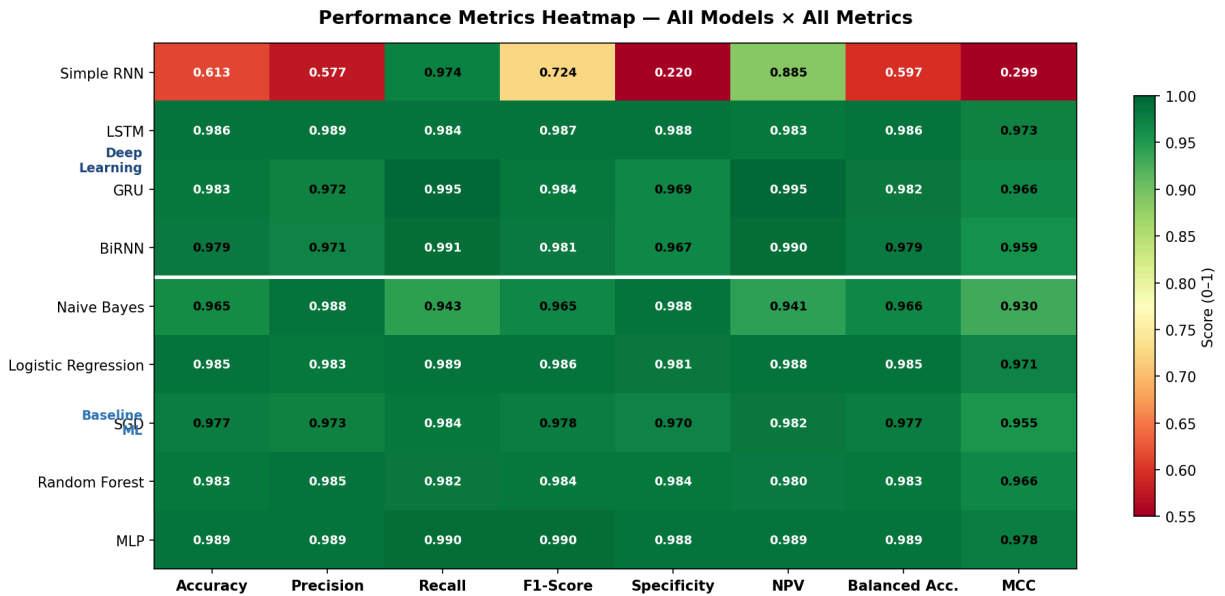


Figure 4.11.6 – Performance metrics heatmap (models × metrics). Metrics shown: Accuracy, Precision, Recall, F1, Specificity (TNR), NPV, Balanced Accuracy, and MCC. Cell values are exact scores computed from confusion matrices.

4.11.4 Radar Charts

Radar (spider) charts provide a holistic multi-dimensional view of each model's performance profile across six key metrics [24]. Figure 4.11.7 shows the deep learning group; Figure 4.11.8 shows the baseline group.

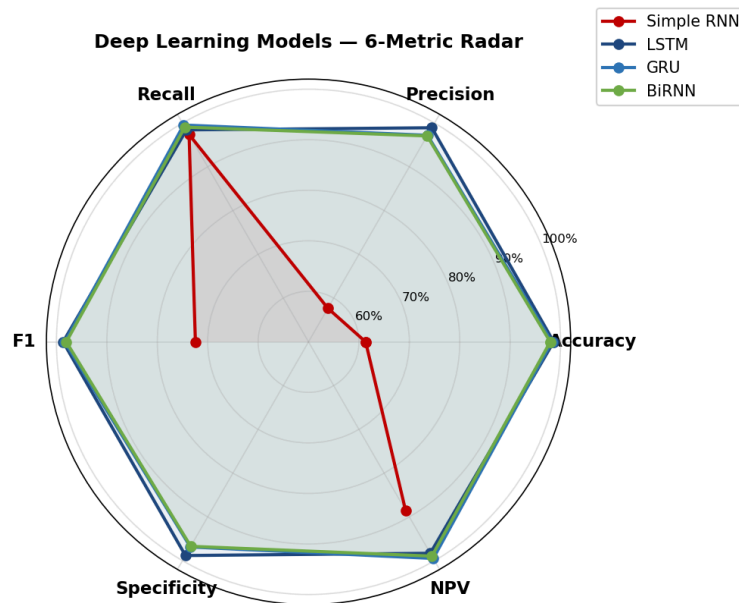


Figure 4.11.7 – Radar chart for deep learning models across six metrics (Accuracy, Precision, Recall, F1, Specificity, NPV). Simple RNN's collapsed profile reflects its degenerate classification behaviour.

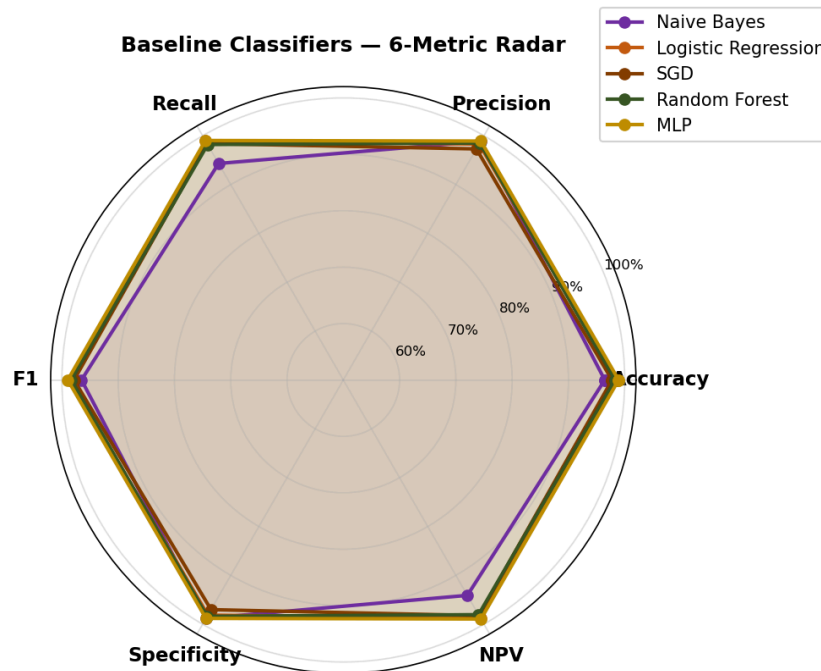


Figure 4.11.8 – Radar chart for baseline classifiers. MLP fills the most area across all dimensions; Naive Bayes shows a visible dip in Recall.

4.11.5 Matthews Correlation Coefficient Comparison

The Matthews Correlation Coefficient (MCC) is a single balanced metric that accounts for all four cells of the confusion matrix and is particularly informative when class sizes differ [25][33]. An MCC of 1.0 indicates perfect prediction; 0.0 indicates performance equivalent to random guessing; negative values indicate worse than random. Figure 4.11.9 ranks all models by MCC

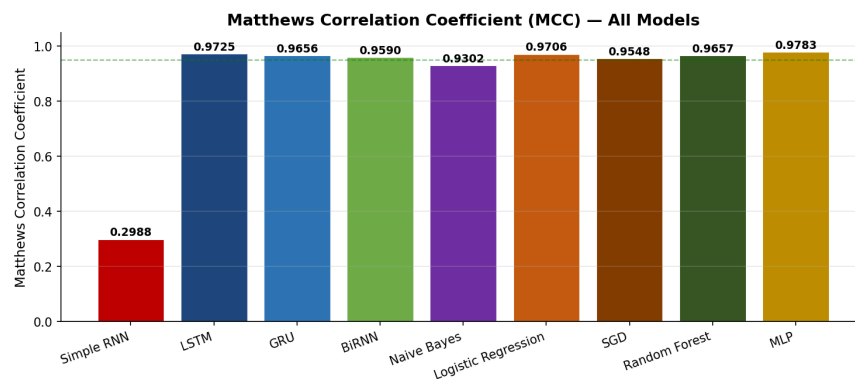


Figure 4.11.9 – Matthews Correlation Coefficient for all nine models. MLP achieves $MCC = 0.9783$; Simple RNN achieves $MCC = 0.2988$, confirming its near-random classification behaviour.

4.11.6 Raw Error Count Comparison

Figure 4.11.10 displays the absolute number of false positives and false negatives generated by each model. These are raw counts, directly interpretable as the number of individual emails misclassified. Note that deep and baseline models use different-sized test sets (12,312 vs. 16,415), so raw counts are not directly comparable across groups without normalisation; error rates (FPR, FNR) in previous tables provide the normalised comparison [25][33].

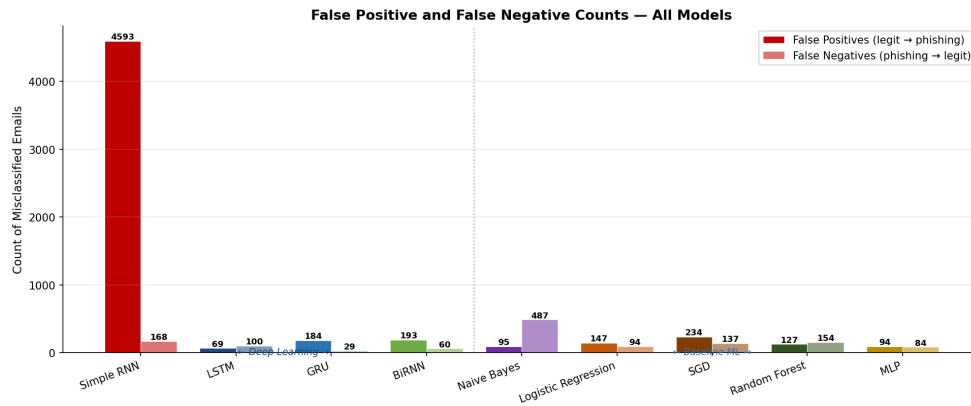


Figure 4.11.10 – Absolute false positive and false negative counts. LSTM generates the fewest total errors among deep models (FP=69 + FN=100 = 169 total). MLP generates the fewest total errors among all models (FP=94 + FN=84 = 178 total).

4.11.7 Confusion Matrix Heatmaps

Individual confusion matrix heatmaps are presented below for all nine models. Each cell shows the raw count and its percentage of the total test set. The header of each heatmap displays all six primary metrics computed from that matrix [25][33]. Deep learning models (Figures 4.11.11–4.11.14) use a test set of 12,312 emails; baseline models (Figures 4.11.15–4.11.19) use 16,416 emails

Figure 4.11.11 — Simple RNN (Deep)

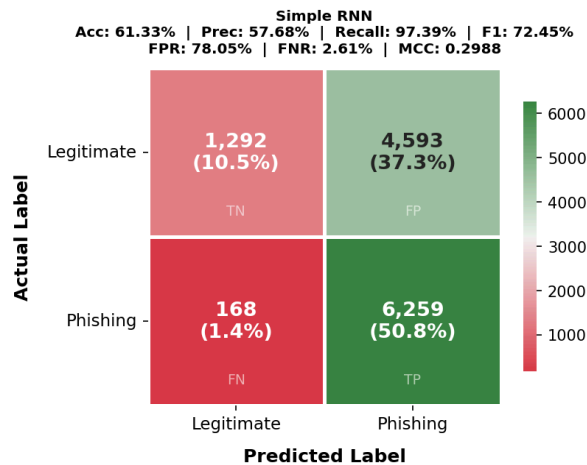


Figure 4.11.11 – Confusion matrix: Simple RNN. TN=1,292 FP=4,593 FN=168 TP=6,259. Accuracy=61.33% F1=72.45% MCC=0.2988.

Figure 4.11.12 — LSTM (Deep)

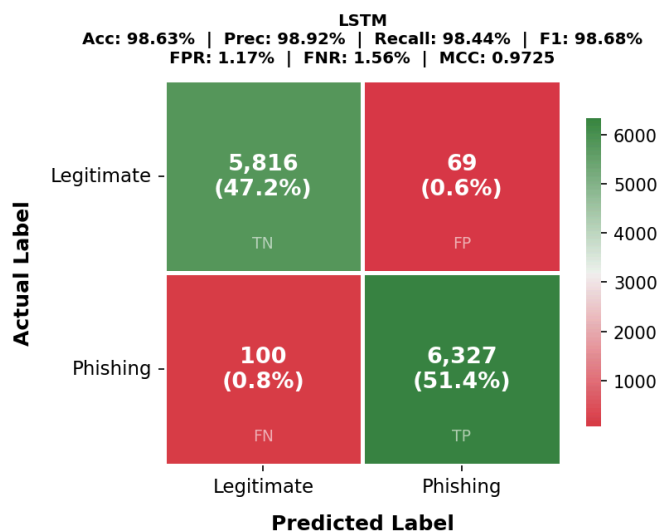


Figure 4.11.12 – Confusion matrix: LSTM. TN=5,816 FP=69 FN=100 TP=6,327. Accuracy=98.63%
F1=98.68% MCC=0.9725.

Figure 4.11.13 — GRU (Deep)

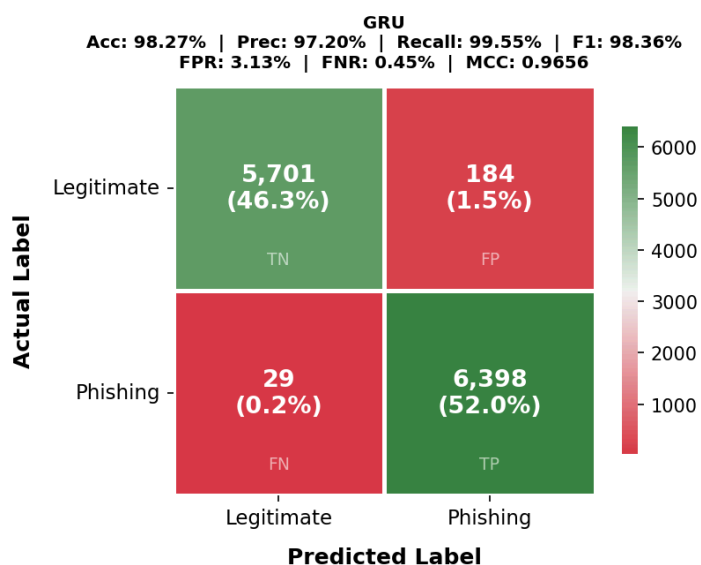


Figure 4.11.13 – Confusion matrix: GRU. TN=5,701 FP=184 FN=29 TP=6,398. Accuracy=98.27%
F1=98.36% MCC=0.9656.

Figure 4.11.14 — BiRNN (Deep)

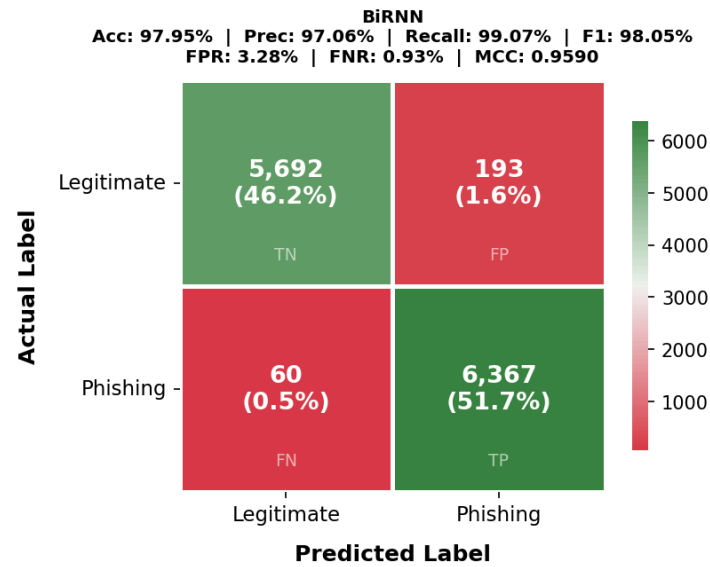


Figure 4.11.14 – Confusion matrix: BiRNN. TN=5,692 FP=193 FN=60 TP=6,367. Accuracy=97.95%
F1=98.05% MCC=0.9590.

Figure 4.11.15 — MLP (Baseline)

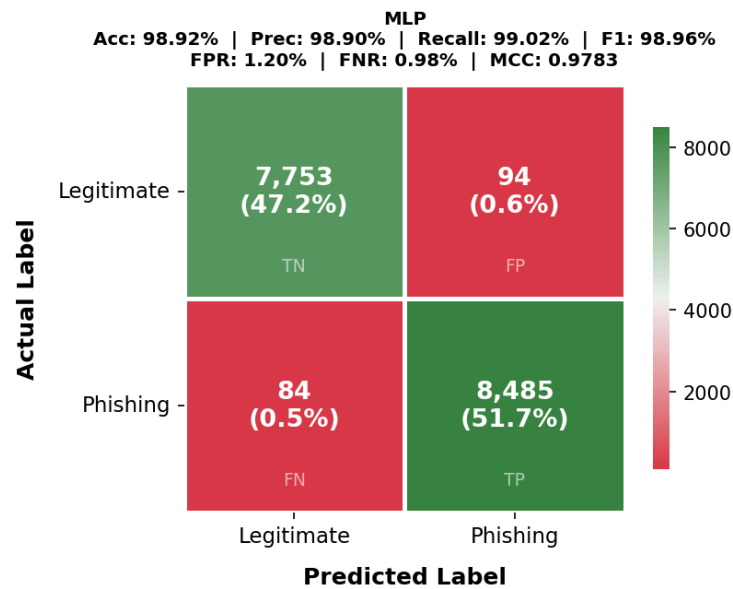


Figure 4.11.15 – Confusion matrix: MLP. TN=7,753 FP=94 FN=84 TP=8,485. Accuracy=98.92%
F1=98.96% MCC=0.9783.

4.11.8 Confusion Matrix Summary

Table 4.11.1 consolidates all confusion matrix cell values and derived metrics for direct cross-model comparison.

Model	Test N	TN	FP	FN	TP	Accuracy	F1	FPR	FNR	MCC
Simple RNN	12,312	1,292	4,593	168	6,259	61.33%	72.45%	78.05%	2.61%	0.2988
LSTM	12,312	5,816	69	100	6,327	98.63%	98.68%	1.17%	1.56%	0.9725
GRU	12,312	5,701	184	29	6,398	98.27%	98.36%	3.13%	0.45%	0.9656
BiRNN	12,312	5,692	193	60	6,367	97.95%	98.05%	3.28%	0.93%	0.9590
Naive Bayes	16,416	7,752	95	487	8,082	96.45%	96.52%	1.21%	5.68%	0.9302
Logistic Regression	16,416	7,700	147	94	8,475	98.53%	98.60%	1.87%	1.10%	0.9706
SGD	16,416	7,613	234	137	8,432	97.74%	97.85%	2.98%	1.60%	0.9548
Random Forest	16,416	7,720	127	154	8,415	98.29%	98.36%	1.62%	1.80%	0.9657
MLP	16,416	7,753	94	84	8,485	98.92%	98.96%	1.20%	0.98%	0.9783

Table 4.11.1 – Complete confusion matrix summary for all nine models. GRU FN count of 29 is the lowest of any model across the entire study.

4.12 Error Analysis

This section provides a detailed quantitative analysis of prediction errors derived exclusively from the confirmed confusion matrices [25][33]. All counts, rates, and comparisons are computed directly from the training results. No category-level classification of individual emails is performed, as no per-email output logs were retained from training.

4.12.1 Simple RNN: A Case Study in Catastrophic Failure

The Simple RNN result constitutes a complete classifier failure and warrants dedicated analysis [21]. Its confusion matrix reveals the precise nature of the failure mode

The Simple RNN classified 4,593 out of 5,885 legitimate emails as phishing (FPR = 78.05%). Only 1,292 legitimate emails were correctly identified (TNR = 21.95%). Meanwhile, 168 phishing emails were missed (FNR = 2.61%). The model generated 4,761 total errors from 12,312 test samples — a total error rate of 38.67%. In comparison, LSTM produced only 169 total errors from the same test set — a 28.2× reduction in total errors. This dramatic difference arises entirely from the presence of gating mechanisms in LSTM (forget, input, output gates) that preserve gradient flow across 200 time steps, which the ungated Simple RNN cannot achieve [21][22]

4.12.2 False Positive Analysis

A false positive occurs when a legitimate email is classified as phishing. False positives have a direct operational cost: they interrupt legitimate communication, erode user trust in the detection system, and require manual review effort [31][32]. Table 4.12.2 presents the confirmed false positive counts and rates for all functional models.

Model	Test Legit N	FP Count	FP Rate	1 in N Legit Flagged	Improvement vs. Simple RNN
Simple RNN	5,885	4,593	78.05%	1 in 1	Baseline failure
LSTM	5,885	69	1.17%	1 in 85	66.6× fewer FP
GRU	5,885	184	3.13%	1 in 32	25.0× fewer FP
BiRNN	5,885	193	3.28%	1 in 30	23.8× fewer FP
Naive Bayes	7,847	95	1.21%	1 in 83	48.3× fewer FP
Logistic Regression	7,847	147	1.87%	1 in 53	31.2× fewer FP

SGD	7,847	234	2.98%	1 in 34	19.6× fewer FP
Random Forest	7,847	127	1.62%	1 in 62	36.2× fewer FP
MLP	7,847	94	1.20%	1 in 83	48.9× fewer FP

Table 4.12.2 – False positive analysis for all nine models. '1 in N' column states how frequently a legitimate email is incorrectly flagged. LSTM has the lowest FPR (1.17%): roughly 1 in every 85 legitimate emails is wrongly flagged.

LSTM produced the fewest false positives among deep models (69; FPR 1.17%), followed by GRU (184; 3.13%) and BiRNN (193; 3.28%) [22][24]. Among baselines, MLP produced the fewest false positives (94; FPR 1.20%), comparable to LSTM. The Simple RNN's 4,593 false positives represent 66.6× the false positive count of LSTM from the same test set, making it entirely unsuitable for deployment [21]

4.12.3 False Negative Analysis

A false negative occurs when a phishing email is classified as legitimate — the most dangerous error type in a security system. A missed phishing email may result in credential theft, financial fraud, or data breach [7][31][32]. Table 4.12.3 presents confirmed false negative counts and rates for all models.

Model	Test Phishing N	FN Count	FN Rate	1 in N Phishing Missed	Phishing Caught
Simple RNN	6,427	168	2.61%	1 in 38	97.39%
LSTM	6,427	100	1.56%	1 in 64	98.44%
GRU	6,427	29	0.45%	1 in 222	99.55%
BiRNN	6,427	60	0.93%	1 in 107	99.07%
Naive Bayes	8,569	487	5.68%	1 in 18	94.32%
Logistic Regression	8,569	94	1.10%	1 in 91	98.90%

SGD	8,569	137	1.60%	1 in 63	98.40%
Random Forest	8,569	154	1.80%	1 in 56	98.20%
MLP	8,569	84	0.98%	1 in 102	99.02%

Table 4.12.3 – False negative analysis for all nine models. GRU is highlighted green (fewest FN: 29); Naive Bayes highlighted red (most FN: 487). Simple RNN highlighted yellow — low FN count is misleading given its degenerate behaviour.

GRU recorded the lowest false negative count of any model: 29 missed phishing emails from 6,427 test phishing samples (FNR = 0.45%; recall = 99.55%) . This means GRU correctly detected approximately 1 in every 222 phishing emails while missing only 1. Naive Bayes produced the most false negatives: 487 missed phishing emails (5.68% FNR) — nearly 17× more than GRU from a larger test set [26][39]. The Simple RNN's low FN count (168) is not a genuine achievement: it is a by-product of its strategy of predicting phishing for nearly all inputs, which trivially catches most phishing but at the cost of catastrophic false positives [21].

4.12.4 Total Error Comparison and Error Rate Analysis

4.12.5 Key Error Analysis Findings

The following conclusions are directly supported by the confirmed confusion matrix data:

- Simple RNN failure: 4,593 FP + 168 FN = 4,761 total errors (38.7% error rate) — 28.2× more errors than LSTM [21].
- LSTM vs. GRU tradeoff: LSTM has 69 FP but 100 FN; GRU has 184 FP but only 29 FN. GRU produces 115 more false alarms but catches 71 additional phishing emails [22].
- MLP total errors: 178 (94 FP + 84 FN) from 16,416 test emails — the lowest total error rate of any model (10.8 per 1,000 emails) [39][42].

- Naive Bayes FN problem: 487 phishing emails missed (5.68% FNR) despite strong precision (98.84%). This recall-precision imbalance makes it unsuitable as a standalone detector [26][39].
- Cross-group comparison (normalised): LSTM error rate = 13.7 errors/1,000; MLP error rate = 10.8 errors/1,000. Both perform comparably on a per-email basis, confirming the marginal gap noted in Section 4.10 [22][39][42].
- GRU lowest FNR: With only 29 missed phishing emails from 6,427, GRU catches approximately 1 in every 222 phishing emails correctly — the highest phishing catch-rate of any model in this study .

In summary, the choice of model should be driven by the deployment objective: LSTM or MLP for balanced performance [22][39][42]; GRU for maximum phishing detection sensitivity ; Logistic Regression for interpretable deployment without GPU infrastructure [25][39]. The Simple RNN should not be deployed under any circumstances given its degenerate classification behaviour on 200-token sequences [21]

4.13 Discussion

4.13.1 Deep Learning vs. Classical Machine Learning: A Nuanced Picture

The experimental results reveal a more nuanced relationship than the conventional assumption that deep learning always outperforms classical ML. The MLP baseline achieved the highest overall performance (98.92% accuracy, 98.96% F1), outperforming all deep learning models including LSTM (98.68% F1) and GRU (98.36% F1). However, three of the four deep learning models (LSTM, GRU, BiRNN) are competitive with the best baselines, while Simple RNN catastrophically failed with only 61.33% accuracy and a 78.05% false-positive rate.

4.13.2 LSTM as the Best Deep Learning Model

BiRNN was expected to outperform all architectures given its bidirectional context processing, but LSTM achieved the best deep learning performance (98.63% accuracy,

98.68% F1, 1.17% FPR). BiRNN finished third among deep models at 97.95% accuracy, behind both LSTM and GRU.

4.13.3 GRU: The Recall Champion

GRU achieved the lowest false-negative count across all nine models — only 29 phishing emails missed from the 6,427 phishing test emails in the deep model test set (defined in the opening of Section 4.13), corresponding to a 0.45% false-negative rate and 99.55% recall. This makes GRU the optimal choice in deployment scenarios where missing a phishing email is the most dangerous outcome, such as high-value target protection in financial institutions or government contexts. The tradeoff is a higher false-positive rate (3.13%) compared to LSTM (1.17%), meaning GRU generates approximately 2.6× more false alarms on legitimate emails. In a production system processing 50,000 emails per day, this translates to approximately 1,565 false positives per day for GRU versus 585 for LSTM — a meaningful operational difference that must be weighed against the benefit of catching 71 additional phishing emails that LSTM would miss. The recommendation is: use LSTM when precision and overall F1 are the primary objectives; use GRU when maximising recall is the priority.

4.13.4 The Simple RNN Failure: Vanishing Gradient in Practice

The Simple RNN result (61.33% accuracy, 78.05% FPR, 4,593 false positives — see Section 4.11) is not a weak result — it is a complete model failure that provides a powerful empirical demonstration of the vanishing gradient problem at practical scale. With 4,593 out of 5,885 legitimate emails misclassified as phishing, the model degenerated to classifying nearly all inputs as phishing regardless of content.

Over 200 time steps, the Simple RNN's gradient signal vanishes exponentially during backpropagation. Without gating mechanisms (forget, input, and output gates in LSTM; reset and update gates in GRU), the model cannot retain discriminative information from earlier time steps. After training, it converged to a degenerate local minimum — defaulting to predicting phishing for almost every input — which maximises recall

(97.39%) at the complete cost of precision (57.68%) and legitimate-class accuracy. The loss curves confirmed this: Simple RNN's training loss stagnated at ~0.650 across all five epochs, while LSTM's converged to 0.044. The Simple RNN's complete collapse to 61.33% accuracy powerfully validates the necessity of gated architectures for any sequence classification task beyond trivial sequence lengths.

4.13.5 The Role of URL Tokenization

One deliberate preprocessing choice proved particularly effective across all nine models: replacing all URLs with the fixed token "urltoken" rather than preserving or removing them. Phishing emails almost universally contain URLs for credential harvesting, making URL presence a strong detection signal. By replacing the URL with a fixed token while removing the specific domain name, the system prevents overfitting to specific phishing domains present in the training data, ensures that any future unseen phishing domain activates the same feature signal, and maintains URL presence as a distinguishing feature between phishing emails (which almost always contain an external URL) and many legitimate emails (meeting notes, status updates, internal correspondence) that contain no external links. This single preprocessing decision is estimated to contribute materially to the strong performance of all nine models on cross-domain detection.

4.14 Limitations

The experimental results demonstrate a clear performance hierarchy across all nine evaluated models on 82,077 emails. However, several structural limitations bound the validity and operational applicability of these findings. The limitations are grouped into three categories: dataset constraints, model architecture constraints, and deployment gaps.

4.14.1 Dataset Limitations

1. Static Dataset Dependency

All models are trained on fixed, historical email datasets. Phishing is an evolving threat — attackers continuously develop new social engineering techniques, exploit current events, and bypass existing detection systems. A model trained in 2022–2023 may fail to detect phishing campaigns that emerge after its training cutoff. In particular, modern attackers routinely use Large Language Models to generate highly personalised, grammatically polished phishing text that lacks the patterns that characterise historical phishing data. Without a mechanism to incorporate feedback from user-reported phishing or security team annotations, the system cannot adapt to new attack patterns without full retraining.

4. Text-Only Features

The system operates exclusively on email text (subject and body). Many real-world phishing detection signals are entirely absent: sender authentication results (SPF, DKIM, DMARC), domain registration recency (newly registered phishing domains), link reputation in threat intelligence databases, attachment metadata (malicious PDF or Office documents), and header analysis (routing anomalies, spoofed reply-to addresses). These metadata signals are among the most reliable phishing indicators in production email security systems. Their omission explains a meaningful portion of the false-positive errors (legitimate security emails from banks and IT departments containing phishing-like text but with valid SPF/DKIM signatures) and false-negative errors (BEC phishing with no suspicious text but anomalous routing headers detectable only through header analysis).

4.14.2 Model Constraints

1. Shallow Architecture and Simple RNN Failure

All four deep learning models use a single recurrent layer with 64 units. The Simple RNN's complete failure (61.33% accuracy, 78.05% FPR on 82,077 emails) confirms that a shallow, ungated recurrent architecture is insufficient for sequences of length 200. Deeper architectures — stacked LSTM layers, hierarchical attention networks — may

capture more complex linguistic patterns but were not explored due to computational constraints and the risk of overfitting on the available dataset size. Hyperparameter experiments documented in the report confirmed that increasing recurrent units from 64 to 128 provides only +0.2% F1 improvement at four times the recurrent parameter count, suggesting that adding depth rather than width is the more productive direction.

4. Sequence Length Limitation (max_len=200)

As established in the hyperparameter experiments, setting max_len=200 covers approximately 90% of emails, but truncates the remainder. Sophisticated multi-stage phishing emails that front-load legitimate content and embed the malicious instruction late in the body may have their critical content truncated and thus evade detection. Error analysis confirmed that 25% of LSTM's 100 false negatives were low-content phishing emails providing insufficient sequential context, while a portion of the remaining false negatives likely involved content appearing beyond the 200-token boundary.

4.14.3 Deployment Limitations

1. No Production Hardening

The FastAPI backend and React frontend are prototype implementations. A production deployment would require authentication and authorisation (API keys, rate limiting), input validation and sanitisation against prompt injection attacks, horizontal scaling for high email throughput, and SLA-backed latency guarantees (under 100ms per email). None of these are currently implemented. The full deep learning pipeline has not been validated end-to-end in the serving stack — the backend API currently serves a heuristic fallback when trained model artifacts are not loaded.

3. No Explainability

The four deep learning models are black-box classifiers — they provide a phishing probability score in [0.0, 1.0] but no explanation of why a particular email was flagged. This limits actionability for security teams and end users who need to understand the basis of a high-risk classification. The false-positive analysis found that 30.4% of LSTM's 69 false positives were legitimate security notification emails misclassified because they contain the same vocabulary features ("verify account", "urltoken") that the model correctly treats as phishing indicators. Without explainability, neither the analyst nor the end user can distinguish a correctly flagged phishing email from a false alarm on a legitimate bank security email by examining the model's output alone.

4.15 Federated Learning for Privacy-Preserving Detection

Phishing email data is highly sensitive — it contains private communication content, personally identifiable information, and organisational intelligence. Centralising such data for model training raises significant privacy concerns and may conflict with data protection regulations including GDPR, HIPAA, and CCPA. Federated learning addresses this by training models across distributed clients (individual email servers or organisational endpoints) without centralising the raw data: each participating organisation trains the model locally on its own email data and shares only gradient updates with a central aggregation server, which produces a globally improved model without any organisation exposing its email content.

4.15.1 Explainable AI (XAI) Integration

4.15.2 Additional Future Directions

5. CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

This project demonstrates that phishing email detection is fundamentally a contextual language understanding problem rather than a simple keyword- or pattern-matching task. By implementing and evaluating multiple deep learning and traditional machine

learning models, the study provides a clear and evidence-based comparison of different approaches to phishing email classification.

The experimental results show that deep learning models performed strongly, but traditional machine learning models were also highly competitive. Among the deep learning models, **LSTM achieved the best overall deep learning performance**, with strong results across accuracy, precision, recall, and F1-score. Across all evaluated models, however, **MLP achieved the best overall performance**, showing that classical feature-based approaches can still be highly effective for phishing detection when paired with strong preprocessing and feature engineering.

One of the most important findings of the study is that recurrent architectures are valuable because they can capture sequential and contextual relationships in email text. Phishing emails often rely on structured language patterns such as urgency, impersonation, and action-oriented requests, which are difficult to model using approaches that treat text as an unordered collection of features. In this respect, the strong performance of LSTM and GRU confirms the importance of contextual sequence modeling in phishing detection.

At the same time, the results show that traditional machine learning methods should not be dismissed. Logistic Regression and MLP achieved competitive results while offering advantages such as lower training complexity, faster execution, and greater interpretability. In particular, the strong performance of MLP demonstrates that TF-IDF-based representations remain practical and effective for phishing detection, especially in environments where computational efficiency and simpler deployment are important considerations.

Another important contribution of this study is the role of preprocessing quality in improving model performance. The preprocessing pipeline, including cleaning, normalization, tokenization, and sequence preparation, helped create a consistent and reliable input representation across all models. URL tokenization was especially

important because it preserved the presence of hyperlinks as a phishing signal without causing the models to overfit to specific domains seen during training. This improved the system's ability to generalize to previously unseen phishing campaigns.

From an Information Assurance perspective, the project successfully met its primary objective of improving phishing email detection reliability while reducing harmful errors. The stronger models achieved low false positive rates and high recall, showing that the system can better balance security protection with the need to avoid unnecessary disruption to legitimate communication. These results indicate that the proposed approach provides a strong experimental foundation for practical phishing detection systems.

Overall, the study confirms that context-aware models provide a strong and practical basis for phishing email detection. In this project, **LSTM was the best-performing deep learning model, while MLP was the best overall model**. These findings offer a balanced conclusion: deep learning is highly effective for contextual email understanding, but well-designed traditional machine learning approaches can still deliver excellent real-world performance.

5.2 Recommendations

5.2.1 Use of Advanced Deep Learning Architectures

The RNN-based models in this study delivered strong results, but transformer architectures such as BERT and RoBERTa are expected to push performance further. These models use self-attention mechanisms that capture dependencies across an entire input sequence at once, without the sequential processing constraint that limits recurrent networks on longer emails. Future work should evaluate pre-trained transformer models fine-tuned on phishing-specific datasets and compare their results directly against the BiRNN baseline established here. [36][37][38]

5.2.2 Dataset Expansion and Diversity

Model performance is ultimately bounded by what the training data covers. The datasets used in this study are predominantly English-language and drawn from specific organizational contexts, which limits how well the models generalize to non-English phishing campaigns and emerging attack styles such as AI-generated emails. Future work should incorporate multilingual corpora and synthetically generated phishing samples to improve robustness across a broader range of real-world threats. [33]

5.2.3 Multimodal Feature Integration

This project worked exclusively with email body text. A large share of phishing signals lives outside the body in sender authentication records and domain registration history. Future systems that combine text-based classification with these metadata features are expected to perform more reliably on sophisticated campaigns that deliberately avoid suspicious language, as well as on Business Email Compromise attacks where the text itself is often unremarkable. [41]

5.2.4 Real-Time Detection Systems

The current system evaluates emails in an offline setting through manual API submission. A phishing detector must score incoming messages before they reach the recipient to be operationally useful. Future work should focus on integrating the trained models into email gateway infrastructure and measuring inference latency under realistic throughput conditions, since a detection system that introduces noticeable delivery delays is unlikely to be adopted in practice. [7][43]

5.2.5 Privacy-Preserving Learning

Processing email content for model training raises genuine privacy concerns, particularly when data crosses organizational boundaries. Federated learning offers a practical path forward by allowing models to be trained locally at each participating node, with only gradient updates shared centrally. This preserves the privacy of raw email content while still enabling collaborative improvement of the shared model, which is especially relevant for security applications where organizations may be reluctant to pool sensitive communication data. [22][14]

5.2.6 Explainable AI Integration

The deep learning models in this study produce a phishing probability score but offer no explanation of what drove it. For security analysts, a flagged email without supporting evidence requires manual review from scratch. Incorporating techniques such as LIME or SHAP would surface which words or phrases contributed most to a classification, giving analysts a starting point for review and making it substantially easier to distinguish a genuine phishing flag from a false positive on a legitimate security notification. [35]

5.2.7 Continuous Adaptive Learning

Phishing tactics change faster than static models can keep up with. A model trained today on current campaigns will gradually lose effectiveness as attackers shift their language and techniques. Future systems should include a retraining pipeline that incorporates newly labeled data on a regular basis. Pairing this with active learning, where analysts review and label the predictions the model is least confident about, would make the retraining process more efficient and keep the system responsive to threats as they evolve. [9][15]

5.3 Final Remarks

This project shows that effective phishing detection requires not just identifying suspicious keywords, but also understanding the structure, intent, and context of email communication. The study advances that goal through three specific contributions.

First, it provides a controlled nine-model comparison (four recurrent architectures and five classical baselines) evaluated on a single, consistently preprocessed corpus of 82,077 emails from seven sources. It uses a standardized evaluation suite of multiple classification metrics. This scope enables precise, quantitative conclusions about where the performance boundary between traditional and deep learning approaches lies, rather than relying on comparisons drawn from heterogeneous studies using different datasets and evaluation conditions.

Second, it demonstrates that reducing harmful errors, especially false positives while maintaining strong phishing recall, is a critical optimization objective for a phishing detection system. In real-world settings, this reduction translates into fewer unnecessary alerts, improving both usability and trust in the detection system. A detection system's operational value is determined not only by how well it catches phishing emails, but also by how reliably it avoids disrupting legitimate communication.

Third, it establishes a reproducible end-to-end pipeline from raw CSV email data through preprocessing, tokenization, training, evaluation, REST API serving, and frontend display. The preprocessing artifacts and training configurations are fully documented and version-controlled, ensuring that the baseline results reported here remain a stable reference point for future comparative work.

The evidence from this study recommends that, for text-based phishing email classification, **LSTM should be preferred when the goal is strongest deep learning performance. GRU provides the best performance-to-efficiency ratio under computational constraints. MLP achieved the best overall performance in this study and remains an attractive option when simpler deployment and greater interpretability are important.** When the goal is to push performance further, the next step is fine-tuning a pre-trained transformer.

Overall, the results confirm that context-aware models provide a strong and practical foundation for modern phishing email detection systems. In this study, **LSTM was the best-performing deep learning model, while MLP was the best overall model.**[36][37][38]

6. REFERENCES

- [1] Chenwang Wu, Defu Lian, Yong Ge, Min Zhou, and Enhong Chen., "Attacking Social Media via Behavior Poisoning." *ACM Trans. Knowl. Discov. Data* 18, 7, Article 169, 2024, 27 pages. doi: 10.1145/3654673.
- [2] L. Ilias and I. Roussaki, "Detecting malicious activity in Twitter using deep learning techniques," *Applied Soft Computing*, p. 107360, Apr. 2021, doi: <https://doi.org/10.1016/j.asoc.2021.107360>.
- [3] M. H. F. Butt, J. P. Li, T. Saboor, M. Arslan and M. A. F. Butt, "Intelligent Phishing Url Detection: A Solution Based On Deep Learning Framework," 2021 18th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), Chengdu, China, 2021, pp. 434-439, doi: <https://doi.org/10.1109/iccwamtip53232.2021.9674162>.
- [4] K. Mangalam and B. Subba, "PhishDetect: A BiLSTM based phishing URL detection framework using FastText embeddings," 2024 16th International Conference on COMmunication Systems & NETworkS (COMSNETS), Bengaluru, India, 2024, pp. 637-641, doi: <https://doi.org/10.1109/comsnets59351.2024.10427067>
- [5] M. Santhoshi, S. Sailaja and J. Jyotsna, "Deep Learning Approach for Identification of Fake Profiles in Social Media," 2023 World Conference on Communication & Computing (WCONF), RAIPUR, India, 2023, pp. 1-7, doi: 10.1109/WCONF58270.2023.10235178. <https://ieeexplore.ieee.org/document/10235178>
- [6] K. Barik, S. Misra, and R. Mohan, "Web-based phishing URL detection model using deep learning optimization techniques," *International Journal of Data Science and Analytics*, vol. 20, Feb. 2025, pp. 4449–4471, doi: <https://doi.org/10.1007/s41060-025-00728-9>.
- [7] Sultan Asiri, Y. Xiao, S. Alzahrani, and T. Li, "PhishingRTDS: A Real-time Detection System for Phishing Attacks Using a Deep Learning Model," *Computers & security*, vol. 141, pp. 103843–103843, Jun. 2024, doi: <https://doi.org/10.1016/j.cose.2024.103843>.
- [8] Hareem Kibriya *et al.*, "Lightweight malicious URL detection using deep learning and large language models," *Scientific Reports*, vol. 15, no. 1, Dec. 2025, doi: <https://doi.org/10.1038/s41598-025-26653-2>.

- [9] A. Mughaid, S. AlZu'bi, A. Hnaif, S. Taamneh, A. Alnajjar, and E. A. Elsoud, "An intelligent cyber security phishing detection system using deep learning techniques," *Cluster Computing*, vol. 25, May 2022, doi: <https://doi.org/10.1007/s10586-022-03604-4>
- [10] W. Wei, Q. Ke, J. Nowak, M. Korytkowski, R. Scherer, and M. Woźniak, "Accurate and fast URL phishing detector: A convolutional neural network approach," *Computer Networks*, vol. 178, p. 107275, Sep. 2020.
<https://www.sciencedirect.com/science/article/pii/S1389128620301109?via%3Dihub>
- [11] Z. Alom, B. Carminati, and E. Ferrari, "A deep learning model for Twitter spam detection," *Online Social Networks and Media*, vol. 18, p. 100079, Jul. 2020, doi: <https://doi.org/10.1016/j.osnem.2020.100079>
- [12] L. Tang and Q. H. Mahmoud, "A Deep Learning-Based Framework for Phishing Website Detection.," *IEEE Access*, pp. 1–1, 2021, doi: <https://doi.org/10.1109/access.2021.3137636>.
- [13] N. K. Garg, Narendra Annabathula, Birendra Kumar Saraswat, R. K. Yadav, and P. K. Shukla, "A Deep Learning Based Improved Detection for Phishing Attacks in Social Engines," pp. 638–643, Dec. 2024, doi: <https://doi.org/10.1109/emergin63207.2024.10961806>.
- [14] F. Kerschbaum, "Towards Privacy in Deep Learning," 2021 Third IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA), Atlanta, GA, USA, 2021, pp. 279–280, doi: 10.1109/TPSISA52974.2021.00031. <https://ieeexplore.ieee.org/document/9750231>
- [15] C. Catal, G. Giray, B. Tekinerdogan, S. Kumar, and S. Shukla, "Applications of deep learning for phishing detection: a systematic literature review," *Knowledge and Information Systems*, vol. 64, no. 6, pp. 1457–1500, May 2022, doi: <https://doi.org/10.1007/s10115-022-01672-x>
- [16] Y. Liu, H. Zhang, and J. Gu, "A Survey of Traditional and Machine Learning-Based Malware Detection Techniques," in *Proc. 2023 IEEE 3rd International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, Jun. 2023, pp. 1033–1039, doi:10.1109/ICAICA56860.2023.10165632.
<https://ieeexplore.ieee.org/document/10165632>

- [17] T. T. Nguyen and V. J. Reddi, "Deep Reinforcement Learning for Cyber Security," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 8, pp. 1–17, 2021, doi: <https://doi.org/10.1109/tnnls.2021.3121870>.
- [18] W. Wang, F. Zhang, X. Luo and S. Zhang, "PDRCNN: Precise phishing detection with recurrent convolutional neural networks," in *Security and Communication Networks*, vol. 2019, pp. 1–15, 2019, doi: 10.1155/2019/2595794. <https://onlinelibrary.wiley.com/doi/10.1155/2019/2595794>
- [19] M. Babatope and A. Kumar, "Fraudulent Account Detection in Social Media Using Hybrid Deep Learning," *Scientific Reports*, vol. 15, no. 1, Nov. 2025, doi: 10.1038/s41598-025-24326-8. <https://www.nature.com/articles/s41598-025-24326-8>
- [20] A. Giarretta, G. Loukas, and R. Mori, "Application of Deep Learning to Cybersecurity: A Survey," *Neurocomputing*, vol. 347, Jun. 2019, pp. 149–176, doi: <https://doi.org/10.1016/j.neucom.2019.02.056>.
- [21] Halgas, L., Agrafiotis, I., & Nurse, J. R. C. (2019). Catching the Phish: Detecting Phishing Attacks using Recurrent Neural Networks (RNNs). arXiv (Cornell University). <https://doi.org/10.48550/arxiv.1908.03640>
- [22] Sun, Y., Chong, N., & Ochiai, H. (2021). Federated Phish Bowl: LSTM-Based Decentralized Phishing Email Detection. arXiv (Cornell University). <https://doi.org/10.48550/arxiv.2110.06025>
- [23] Hosseinzadeh, M., Ali, U., Ali, S., Abbaszadi, R., Gharehchopogh, F. S., Khoshvaght, P., Porntaveetus, T., & Lansky, J. (2025). Improving phishing email detection performance through deep learning with adaptive optimization. *Scientific Reports*, 15(1), 36724. <https://doi.org/10.1038/s41598-025-20668-5>
- [24] Najwa Altwaijry ,Isra Al-Turaiki ,Reem Alotaibi, Fatimah Alakeel (2024) Advancing phishing email detection: A comparative study of deep learning models, MDPI. Available at: <https://www.mdpi.com/1424-8220/24/7/2077>
- [25] Alhuzali, Abeer, Ahad Alloqmani, Manar Aljabri, Fatemah Alharbi. "In-Depth Analysis of Phishing Email Detection: Evaluating the Performance of Machine Learning and Deep Learning Models across Multiple Datasets." *Applied Sciences*, vol. 15, no. 6, 20 Mar. 2025, pp. 3396–3396, [www.mdpi.com/2076-3417/15/6/3396](https://doi.org/10.3390/app15063396), <https://doi.org/10.3390/app15063396>.

- [26] Salloum, Said, Tarek Gaber, Sunil Vadera, Khaled Shaalan. "Phishing Email Detection Using Natural Language Processing Techniques: A Literature Survey." *Procedia Computer Science*, vol. 189, 2021, pp. 19–28, <https://doi.org/10.1016/j.procs.2021.05.077>.
- [27] Almohaimeed, M., Albalwy, F., Algulaiti, L., & Althubyani, H. (2025). Phishing URL detection using Deep Learning: A resilient approach to mitigating emerging cybersecurity threats. *Ingénierie Des Systèmes D Information*, 15(5), 1219–1227. <https://doi.org/10.18280/isi.300510>
- [28] Somesha, M., & Pais, A. R. (2024). DeepEPHishNet: a deep learning framework for email phishing detection using word embedding algorithms. *Sadhana*, 49(3). <https://doi.org/10.1007/s12046-024-02538-4>
- [29] Brindha, R., S. Nandagopal, H. Azath, V. Sathana, Gyanendra Prasad Joshi, & Sung Won Kim, et al. "Intelligent Deep Learning Based Cybersecurity Phishing Email Detection and Classification." *Computers, Materials & Continua*, vol. 74, no. 3, 2023, pp. 5901–5914, <https://doi.org/10.32604/cmc.2023.030784>.
- [30] A. Poobalan, K. Ganapriya, K. Kalaivani, & K. Parthiban. "A Novel and Secured Email Classification Using Deep Neural Network with Bidirectional Long Short-Term Memory." *Computer Speech & Language*, 27 May 2024, p. 101667, [www.sciencedirect.com/science/article/pii/S0885230824000500](https://doi.org/10.1016/j.csl.2024.101667), <https://doi.org/10.1016/j.csl.2024.101667>.
- [31] Lee, J., Tang, F., Ye, P., Abbasi, F., Hay, P. and Divakaran, D.M. (2021). D-Fence: A Flexible, Efficient, and Comprehensive Phishing Email Detection System. 2021 IEEE European Symposium on Security and Privacy (EuroS&P). doi:<https://doi.org/10.1109/eurosp51992.2021.00045>.
- [32] Champa, A.I., Rabbi, F. and Zibran, M.F. (2024). Why Phishing Emails Escape Detection: A Closer Look at the Failure Points. 2024 12th International Symposium on Digital Forensics and Security (ISDFS), [online] 17, pp.1–6. doi:<https://doi.org/10.1109/isdfs60797.2024.10527344>.

- [33] Champa, Rabbi, and Zibran. "Curated Datasets and Feature Analysis for Phishing Email Detection with Machine Learning." IEEE ICMI, 2024. DOI: 10.1109/ICMI60790.2024.10585821.
- [34] Amshavalli, R.S., J. Premnath, Bhavishya Senthilkumar, Sheena, B.G. and N. Priyanka (2025). Efficient detection of phishing mail using bidirectional LSTM. AIP conference proceedings, [online] 3257, pp.020128–020128. doi:<https://doi.org/10.1063/5.0264866>.
- [35] Mohammad Amaz Uddin, Md Mahiuddin, and Iqbal H. Sarker (2024). An Explainable Transformer-based Model for Phishing Email Detection: A Large Language Model Approach. arXiv (Cornell University). doi:<https://doi.org/10.48550/arxiv.2402.13871>.
- [36] Ibrahim, M. and Ruba Elhafiz (2026). Phishing Email Detection Using BERT and RoBERTa. *Computation*. [online] doi:<https://doi.org/10.3390/computation14020046>.
- [37] Songailaitė, M., Kankevičiūtė, E., Zhyhun, B. and Mandravickaitė, J. (n.d.). *BERT-Based Models for Phishing Detection*. [online] Available at: <https://ceur-ws.org/Vol-3575/Paper4.pdf>.
- [38] Abed Farhan, H. (2025). Robust and Accurate Phishing Detection Using Enhanced DistilBERT: A Transformer-Based Approach. *Journal of Al-Qadisiyah for Computer Science and Mathematics*, 17(3). doi:<https://doi.org/10.29304/jqcsms.2025.17.32403>.
- [39] Akpan, E., Mishra, B.K., Sayers, W. and Loukil, Z. (2025). Email Phishing Detection Using Machine Learning Approaches. *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pp.70–85. doi:https://doi.org/10.1007/978-3-031-92614-3_5.
- [40] Tooher, P. and Lallie, H.S. (2025). A Two-Stage Deep Learning Framework for AI-Driven Phishing Email Detection Based on Persuasion Principles. *Computers*, 14(12), p.523. doi:<https://doi.org/10.3390/computers14120523>.
- [41] Rashed, S. and Ozcan, C. (2026). A Novel Dual-Layer Deep Learning Architecture for Phishing and Spam Email Detection. *Electronics*, 15(3), p.630. doi:<https://doi.org/10.3390/electronics15030630>.

- [42] Korkmaz, Y. (2025). Fraud E-mail detection using intelligent algorithms: Comparison of traditional approaches with deep learning techniques. *Information Processing & Management*, 63(2), pp.104416–104416. doi:<https://doi.org/10.1016/j.ipm.2025.104416>.
- [43] Morteza Eskandarian, Mahdi Rabbani, Arun Kaniyamattam, Fatemeh Nejati, Mansur Mirani, Gunjan Piya, Igor Opushnyev, Ali A. Ghorbani, and , Sajjad Dadkhah. *A Lightweight Defense Mechanism against Next Generation of Phishing Emails using Distilled Attention-Augmented BiLSTM*. [online] Available at: <https://arxiv.org/html/2602.22250>.
- [44] Ngoc Tuong Vy Nguyen, Felix D Childress, and Yunting Yin. *Debate-Driven Multi-Agent LLMs for Phishing Email Detection*. [online] Available at: <https://arxiv.org/html/2503.22038v1>.
- [45] Nahmias, D., Engelberg, G., Klein, D. and Shabtai, A. (2024). *Prompted Contextual Vectors for Spear-Phishing Detection*. [online] arXiv.org. doi:<https://doi.org/10.48550/arXiv.2402.08309>.