

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
VALPARAÍSO - CHILE



“COMPARATIVE ANALYSIS OF DEEP LEARNING MODELS VERSUS MACHINE LEARNING ALGORITHMS FOR SPAM EMAIL DETECTION”

PAULINA VALERIA VEGA RIVERA

THESIS TO APPLY FOR THE PROFESSIONAL TITLE OF
INGENIERO CIVIL EN INFORMÁTICA

Advisor: Carlos Valle, Ph.D.
Co-Guide: Iván Pizarro
Co-Advisor: Ricardo Ñanculef, Ph.D.

July - 2025



CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título; ☐ Tesis de Postgrado;

Título del trabajo: Comparative Analysis of Deep Learning Models Versus Machine Learning Algorithms for Spam Email Detection

Nombre del candidato(a): Paulina Valeria Vega Rivera

Carrera / Grado: Ingeniería Civil Informática

Campus: Casa Central Valparaíso ; **Departamento:** Departamento de Informática

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Carlos Antonio Valle Vidal, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL

☒ El trabajo **NO contiene información que amerite confidencialidad** y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (embargo) por:

☐ 6 meses; ☐ 12 meses; ☐ 2 años; ☐ 3 años; ☐ 5 años; ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 29-07-2025

; Firma:

Estudiante o Candidato(a):

Fecha: 28-07-2025

; Firma:

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

DEDICATION

To my family, who (I hope) always believed in me, and with whom I'll celebrate.

To my partner, for always being there; and yes, we'll celebrate too.

To my friends, with whom (obviously) I'll be celebrating as well.

ACKNOWLEDGEMENTS

I would like to thank my family for their constant encouragement and understanding, and my partner for providing both intellectual and emotional support during this journey. I would also like to thank Carlos Valle and Iván Pizarro for their guidance and patience throughout this research.

RESUMEN

Resumen— La detección de correos spam es una tarea crítica en la ciberseguridad, complicada por la constante evolución de técnicas diseñadas para evadir los filtros. Este estudio compara cinco modelos de aprendizaje automático y nueve de aprendizaje profundo, evaluados con ocho métricas sobre un conjunto de 5,500 correos. Se aplicaron pruebas estadísticas a los mejores modelos para evaluar su significancia. RoBERTa obtuvo consistentemente el F1 score más alto, mientras que los modelos ajustados de GPT, entrenados con menos datos, también lograron resultados competitivos. Entre los modelos tradicionales, SVM, NB y RF fueron los más efectivos, aunque fueron superados por los cinco modelos basados en Transformers. El objetivo general es ofrecer un *benchmark* amplio entre enfoques tradicionales y modernos de detección de spam bajo condiciones prácticas.

Palabras Clave— Aprendizaje Profundo, Aprendizaje Automático, Detección de Spam, Ciberseguridad.

ABSTRACT

Abstract— Spam email detection is a critical task in cybersecurity, complicated by constantly evolving spam tactics that are designed to trick the filtering systems. This study compares five machine learning models and nine deep learning models, evaluated using eight performance metrics on a combined dataset of 5,500 email. Statistical tests were applied to the top-performing models to assess significance. Results show that RoBERTa consistently achieves the highest F1 score among all deep learning models, while the fine-tuned GPT models, considered a special case due to being trained on significantly smaller datasets, still perform competitively. Among machine traditional learning models, SVM, NB and RF achieved the highest score, however, they still performed worse than the five Transformer-based models. Overall, the study's goal is to provide a comprehensive benchmark of traditional and modern approaches to spam detection under practical constraints.

Keywords— Deep Learning, Machine Learning, Spam Detection, Cybersecurity.

GLOSSARY

AUC-PR: Area Under the Precision-Recall Curve
BAMec: Bahdanau Attention Mechanism
BERT: Bidirectional Encoder Representation from Transformers
BiGRU: Bidirectional Gated Recurrent Unit
BiLSTM: Bidirectional Long Short-Term Memory
BiRNN: Bidirectional Recurrent Neural Network
BPTT: Backpropagation Through Time
CPU: Central Processing Unit
CNN: Convolutional Neural Networks
DL: Deep Learning
Flan-T5: Fine-Tuned Language Net
FN: False Negatives
FP: False Positives
FPR: False Positive Rate
GPU: Graphics Processing Unit
GRU: Gated Recurrent Unit
GPT: Generative Pre-Trained Transformer
HTML: Hypertext Markup Language
KNN: K-Nearest Neighbor
LAMec: Luong Attention Mechanism
LR: Learning Rate
LSTM: Long Short-Term Memory
MCC: Matthews Correlation Coefficient
ML: Machine Learning
NLP: Natural Language Processing
NB: Naive Bayes
RAM: Random Access Memory
RF: Random Forest
RoBERTa: Robustly Optimized BERT Approach
RNN: Recurrent Neural Network
ROC-AUC: Area Under the Receiver Operating Characteristic Curve
SDP: Scale Dot Product Attention Mechanism
SetFit: Sentence Transformer Fine-Tuning
SMS: Short Message Services
SVM: Support Vector Machine
TN: True Negatives
TP: True Positives
TPR: True Positive Rate
XGBoost: Extreme Gradient Boosting

Contents

RESUMEN	iv
ABSTRACT	iv
GLOSSARY	v
LIST OF FIGURES	viii
LIST OF TABLES	viii
INTRODUCTION	1
CHAPTER 1: Problem Definition	2
1.1 Spam	2
1.2 State-of-the-Art	3
1.3 From Traditional to Deep Learning-Based Spam Filters	4
1.3.1 Machine Learning	5
1.3.2 Deep Learning	6
1.3.3 Deep Learning vs Machine Learning	7
1.4 Identified Problems	8
1.4.1 Techniques Used by Spammers	9
1.5 Objectives	10
1.5.1 Specific Objectives	10
CHAPTER 2: Framework	12
2.1 Metrics	12
2.1.1 Accuracy	12
2.1.2 Precision	13
2.1.3 Recall	13
2.1.4 F1 Score	13
2.1.5 MCC	14
2.1.6 ROC-AUC	14
2.1.7 AUC-PR	14
2.1.8 FPR	15
2.1.9 Primary Metric for Model Comparison	15
2.2 Understanding the Impact of Datasets on Machine and Deep Learning	15
2.2.1 Dataset Size	15
2.2.2 Dataset Noise	16
2.2.3 Dataset Imbalance	17
2.3 Cybersecurity Datasets: Characteristics and Challenges	18
CHAPTER 3: Solution Proposal	19
3.1 Selected Datasets Based on Recency and Relevance	19

3.2	Chosen Models Based on Literature Review	21
3.3	Transformer-Based Models	21
3.3.1	GPT	22
3.3.2	BERT	22
3.3.3	RoBERTa	22
3.3.4	SetFit	22
3.3.5	Flan-T5	22
3.4	Convolutional Models	23
3.4.1	CNN	23
3.5	Sequence Models	23
3.5.1	RNN	23
3.5.2	LSTM	23
3.5.3	GRU	24
3.6	Attention Mechanisms	24
3.6.1	Bahdanau Attention Mechanism	24
3.6.2	Luong Attention Mechanism	24
3.6.3	Scale-Dot Product Attention Mechanism	24
3.7	Classical Machine Learning Models	25
3.7.1	SVM	25
3.7.2	Naïve Bayes	25
3.7.3	Random Forest	25
3.7.4	KNN	25
3.7.5	XGBoost	26
CHAPTER 4: Solution Validation		27
4.1	Hardware Components and Workflow	27
4.2	Deep Learning Models by Roumeliotis, Tselikas and Nasiopoulos	27
4.2.1	GPT Model Variants and Fine-Tuning	28
4.3	Deep Learning Models by Labbone and Moran	29
4.4	Deep Learning Models by Ahmed, Akter, Rahman, et al.	30
4.5	Machine Learning Models by Li	31
4.6	Machine Learning Model by Opara, Modesti and Golightly	32
4.7	Comparative Summary	32
4.7.1	Statistical Comparison of Top-Performing Models	38
4.7.2	Controlled Comparison with GPT Models	38
CHAPTER 5: Conclusions		41
5.1	Limitations	44
5.2	Future Work	44
ANNEXES		46
REFERENCES		47

List of Figures

1	Global spam volume as percentage of total e-mail traffic from 2011 to 2022. .	2
2	Flowchart depicting the typical email classification process.	5
3	Illustration depicting the placement of deep learning.	6
4	General architecture of a shallow network and a deep neural network. . . .	6
5	Performance comparison in relation to the amount of data available.	8
6	Problem Tree.	9
7	For Transformers, performance improves smoothly as we increase the model size, dataset size, and amount of compute used for training.	16
8	Donut charts showing ham/spam and split proportions of the new dataset. Source: Made by the author.	21
9	The structure of the JSONL files used for training, validation, and testing the GPT models follows OpenAI's format. For fine-tuning GPT-4.1-Mini and GPT-4.1-Nano, the "system" field is replaced with "developer".	28
10	F1 score percentages of deep learning models on the classification task. . . .	33
11	F1 score percentages of GPT models on the classification task.	35
12	F1 score percentages of the top-performing DL and ML models.	36
13	Comparison of training dataset sizes.	42

List of Tables

1	Confusion matrix for binary classification.	12
2	Summary of public and private spam datasets, their content, and availability.	19
3	Key details of the datasets chosen.	20
4	Hyperparameters of the models.	27
5	The hyperparameters were not manually selected; instead, the models were fine-tuned using OpenAI's default configuration settings.	29
6	Hyperparameters of the models.	30
7	Hyperparameters of the models.	31
8	Comparison of the F1 scores of all models.	31
9	Performance metrics (mean $\pm$ standard deviation) across all deep learning models.	34
10	Performance metrics (mean $\pm$ standard deviation) across GPT models. . . .	34
11	Performance metrics (mean $\pm$ standard deviation) across all machine learning models.	37
12	Pairwise Wilcoxon signed-rank test results comparing model performance and their interpretation.	39
13	Comparison of the F1 scores of GPT models vs RoBERTa model trained on the same sample set.	40

INTRODUCTION

Despite advances in spam filtering technologies, spammers continue to adapt their strategies, making spam detection a persistent and evolving challenge (Wittel and Wu, 2004). These adaptations include manipulating input data to induce false negatives and crafting emails that closely mimic legitimate communication to deceive recipients (Dalvi et al., 2004). Traditionally, machine learning approaches such as Naive Bayes, Decision Trees, and Support Vector Machines were the standard for email classification (Li, 2023). However, their ability to generalize to newer types of spam, such as phishing email, image-based spam, obfuscation, and personalized language generation comes into question (Jáñez-Martino et al., 2022).

In recent years, deep learning models have gained popularity in natural language processing tasks due to their superior ability to model complex language patterns. These models are now being applied to spam detection with promising results (Apruzzese et al., 2018; Kyaw et al., 2024). Furthermore, the emergence of instruction-tuned large language models, such as GPT-4, has opened new avenues for applying generative models to text classification tasks, including spam filtering and phishing detection (Roumeliotis et al., 2024). These models not only leverage massive amounts of training data but also exhibit transfer learning capabilities, enabling them to adapt to downstream tasks with minimal fine-tuning (OpenAI, 2024).

Despite these advances, fair and reproducible comparisons between traditional and modern approaches remain difficult. Many studies rely on different datasets, often collected in varying time periods, which complicates direct comparison. Most public spam and phishing datasets date back to the early 2000s, raising concerns about the relevance of evaluation results to modern attacks (Jáñez-Martino et al., 2022; Apruzzese et al., 2018). Another prevalent dataset issue is the imbalances found in them, which affects the performance of both machine learning and deep learning models (Johnson and Khoshgoftaar, 2019; Vaswani et al., 2023). In addition, deep learning models typically require large, balanced datasets and significant computational resources, making them impractical for some real-world scenarios (Sarker, 2021).

To address these challenges, this study evaluates the performance of a range of deep learning and traditional machine learning models on a newly constructed dataset containing 5,500 emails, aggregated from three well-known public corpora. Models are assessed using eight performance metrics, with statistical tests applied to the top-performing algorithms to determine the significance of differences. Particular attention is given to training size limitations and cost considerations when interpreting the performance of GPT models.

Overall, this study benchmarks deep learning and machine learning models for spam detection under realistic constraints, highlighting trade-offs in accuracy, generalization, and resource use.

CHAPTER 1

Problem Definition

1.1 Spam

It is justifiable to assert that in the present day, most individuals use email as both a method of formal communication and as a channel for receiving news. As stated by Merriam-Webster, Spam is:

Unsolicited, usually commercial messages (such as emails, text messages, or Internet postings) sent to a large number of recipients or posted in a large number of places.
(Merriam-Webster, 2025)

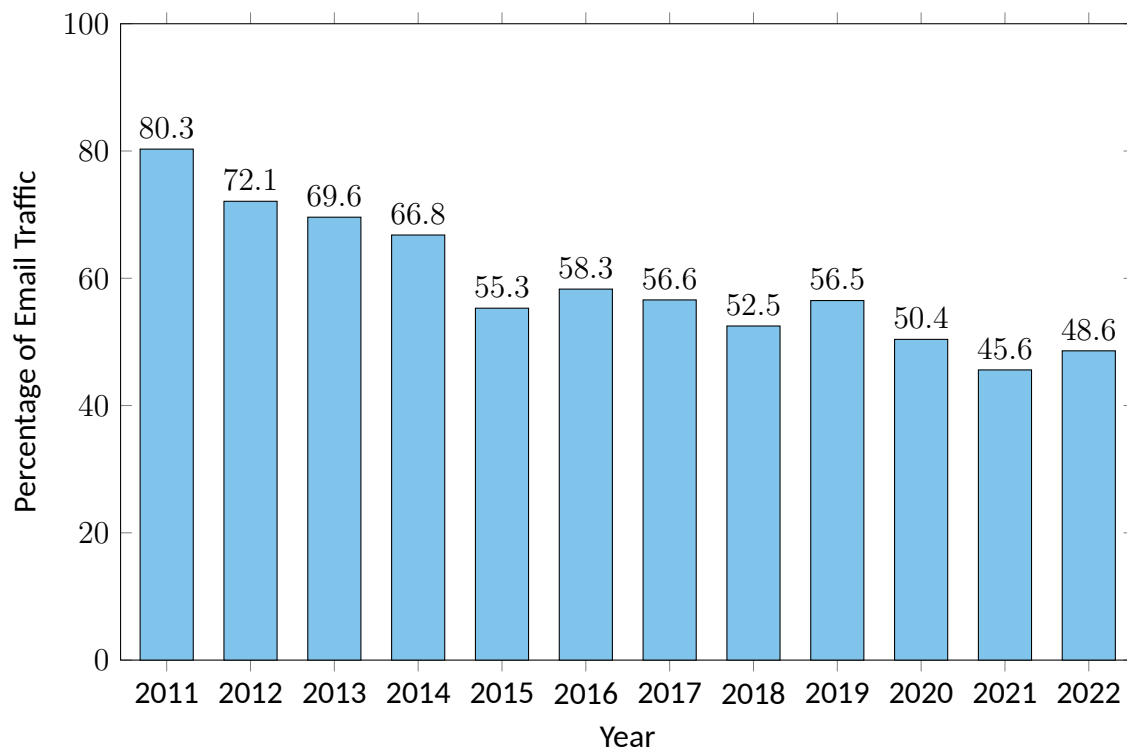


Figure 1: Global spam volume as percentage of total e-mail traffic from 2011 to 2022.
Source: Petrosyan (2022).

These emails often have the goal of obtaining sensitive personal information, defrauding individuals of their money, or deceiving them into opening attachments containing malware. As depicted in Figure 1, despite a significant decrease in the volume of spam emails, going from 80% in 2011 to 49% in 2022, the number is still alarmingly high (Petrosyan, 2022). This

percentage shows that half of the emails currently being sent on the internet are created with the explicit purpose of generating profit through dubious and, sometimes, unlawful means. The persistence of junk email in contemporary times can be attributed to a fundamental factor: the cost-effectiveness inherent in sending bulk email. Although the study alluded to in this section dates back to 1998, its findings remain relevant in the present. To this day, the practice of sending bulk email retains a low threshold of entry, requiring only a modest investment, typically in the range of a few hundred dollars to send about 1,000 email messages. For the amateurs, a personal computer, internet, and a free email client software will do. Due to its accessible and cost-effective nature, bulk mailers can recoup their expenses even when only a minuscule fraction of the messages they send out yields profitable outcomes (Cranor and LaMacchia, 1998).

Phishing is a particular type of spam that involves social engineering and disguise that may allow attackers to trick victims into revealing sensitive information. As stated by the Merriam-Webster, phishing is:

The practice of tricking Internet users (as through the use of deceptive email messages or websites) into revealing personal or confidential information which can then be used illicitly.
(Merriam-Webster, 2025)

The goal of these attacks is to deceive victims into disclosing sensitive information for identity theft or to extort money through methods such as ransomware or credit card fraud (Ferrara, 2019). In 2006, researchers from Indiana University (Jagatic et al., 2007) conducted a study to evaluate the effectiveness of phishing emails. The researchers found that when a phishing email was sent by an unknown sender impersonating the university, it had a 16% success rate in capturing user credentials. However, when the attacker impersonated a friend of the victim, the success rate increased dramatically to 75%.

1.2 State-of-the-Art

Recent advancements in deep learning have significantly influenced the development of spam email detection systems, though findings across studies remain mixed. Hossain et al. (2021) conducted a comparative study involving four traditional machine learning techniques and three deep learning models, RNN, ANN, and Gradient Descent. Their findings indicated that, for tabular datasets, the machine learning algorithms outperformed their deep learning counterparts, highlighting the importance of data structure in determining model efficacy. In contrast, AbdulNabi and Yaseen (2021) evaluated two deep learning models, BERT and BiLSTM, against traditional methods such as KNN and NB. Their results suggested that the deep learning approaches outperformed the classical algorithms in handling textual data.

Further evidence of deep learning's effectiveness is found in the work by Shajahan and Lekshmy (2023), who proposed a hierarchical attention-based hybrid deep learning model, which achieved a peak accuracy of 99.2% using the TREC 2007 dataset.

Some of the studies discussed here have also shown promising deep learning results. The *next-generation of spam filtering* was explored by Roumeliotis et al. (2024) utilizing two Kaggle datasets. Using the "NLP - Spam/Ham Email Classification" dataset from Kaggle, among the evaluated models, BERT achieved the highest F1 score of 99.01%, followed by RoBERTa with a score of 98.4%, a fine-tuned GPT-4 at 98.33%, baseline GPT-4 at 94.21%, and a CNN model at 80.42%.

Also a solution explored in this thesis, Labonne and Moran (2023) introduced the Spam-T5 study, which evaluated performance on four popular spam datasets, including both email and SMS data. Their results showed that the deep learning models, SetFit, RoBERTa, and Spam-T5 consistently outperformed the six traditional machine learning models (NB, Logistic Regression, KNN, SVM, XGBoost, and LightGBM), achieving F1 scores of 95%, 97%, and 96% respectively, on the SpamAssassin Public Corpus.

Lastly, an alternate approach not based on Transformers was proposed by Ahmed et al. (2023), in which the authors explored RNN, LSTM and GRU combined with a variety of attention mechanisms. Utilizing the Enron dataset, the RNN plus Bahdanau attention achieved an F1 score of 99.8%, while both the LSTM and GRU models, when paired with SDP self-attention, achieved an F1 score of 99.97%.

In the machine learning domain, Li (2023) evaluated several traditional machine learning algorithms, such as SVM, NB, RF and KNN. In the second experiment, utilizing the "Spam-Filter" dataset from Kaggle, these models achieved high accuracy scores of 97.54%, 97.41%, 99.3% and 87.84%, respectively. Another machine learning approach, combining XGBoost with stylometric analysis, was used by Opara et al. (2025) to detect AI-generated phishing emails. It achieved an F1 score of 96%.

1.3 From Traditional to Deep Learning-Based Spam Filters

As of today, the solution to the junk email problem is rather straightforward. Nearly all email service providers include a primary inbox, where most messages can be found, and a default "spam folder", sometimes referred to as "junk email folder", where any emails classified as such are automatically redirected to, see Figure 2. The efficacy of this solution is contingent upon the proficiency of the classification algorithm employed, which frequently takes the form of a machine learning based algorithm. Misclassification can result in users overlooking genuinely important emails that inadvertently land in their spam folder. Furthermore, they may be misled when a deceptive email resides in their primary inbox, as it escaped the spam categorization.

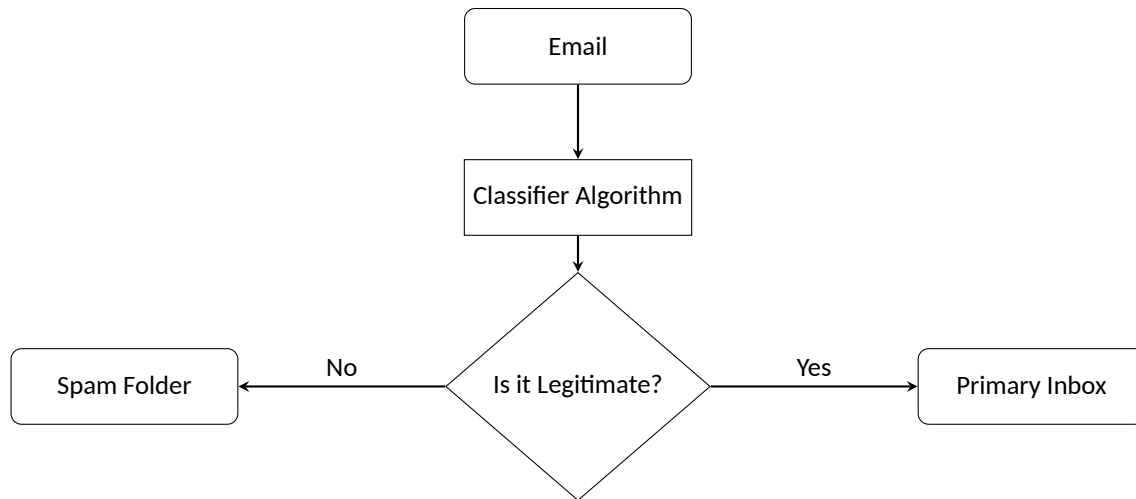


Figure 2: Flowchart depicting the typical email classification process.
Source: Made by the author.

The algorithm must exhibit a high degree of competence in discerning the nuanced cues and subtleties within each email, as evidenced in the preceding section on spam email types. This diversity encompasses a wide array of email variations, each characterized by its own unique level of intricacy and complexity. At this point, the challenge becomes apparent.

1.3.1 Machine Learning

Machine learning can be described as a set of computational techniques that utilize *experience* to enhance performance and make precise predictions. In this context, “experience” pertains to prior information available for learning, which typically takes the form of electronic data collected and made available for analysis. In summation, machine learning consists of designing efficient and accurate prediction algorithms (Mohri et al., 2018).

As mentioned previously, the task of email classification often relies on machine learning-based algorithms. For example, according to Google, TensorFlow, an open-source machine learning framework developed by them, has made it possible to block image-based messages, identify emails with concealed embedded content, and detect messages from newly created domains attempting to disguise a small volume of spam within legitimate traffic, which has allowed them to block 100 million additional spam messages every day (Kumaran, 2019). In fact, machine learning is quite popular in spam email filtering, with the paper by Sahami et al. (1998), titled “A Bayesian Approach to Filtering Junk-Email” presented at the AAAI-98 ¹ Workshop on Learning for Text Categorization is credited as one of the earliest uses of it for this purpose.

¹Fifteenth National Conference on Artificial Intelligence held July from the 26th to the 30th in 1998.

1.3.2 Deep Learning

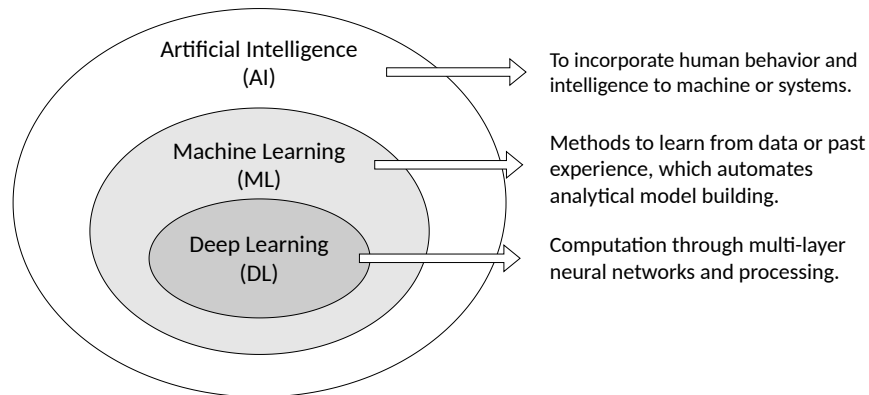


Figure 3: Illustration depicting the placement of deep learning.
Source: Adapted from Sarker (2021).

Nowadays, deep learning is considered a core technology of the Fourth Industrial Revolution. Deep learning is a branch of machine learning and artificial intelligence. Figure 3 depicts the position of deep learning, comparing it with machine learning and artificial intelligence (Sarker, 2021). Deep learning is characterized by the utilization of multiple hidden layers in the neural network architecture, hence the term deep. These layers are responsible for learning representations of data with multiple levels of abstraction. Through the technique of backpropagation, deep learning models iteratively adjust their internal parameters. These adjustments enable the computation of representations in each layer based on the information from the previous layer (LeCun et al., 2015). Figure 4 provides a visual representation highlighting the contrast between deep neural networks, which contain multiple hidden layers, and shallow networks with only one or very few hidden layers.

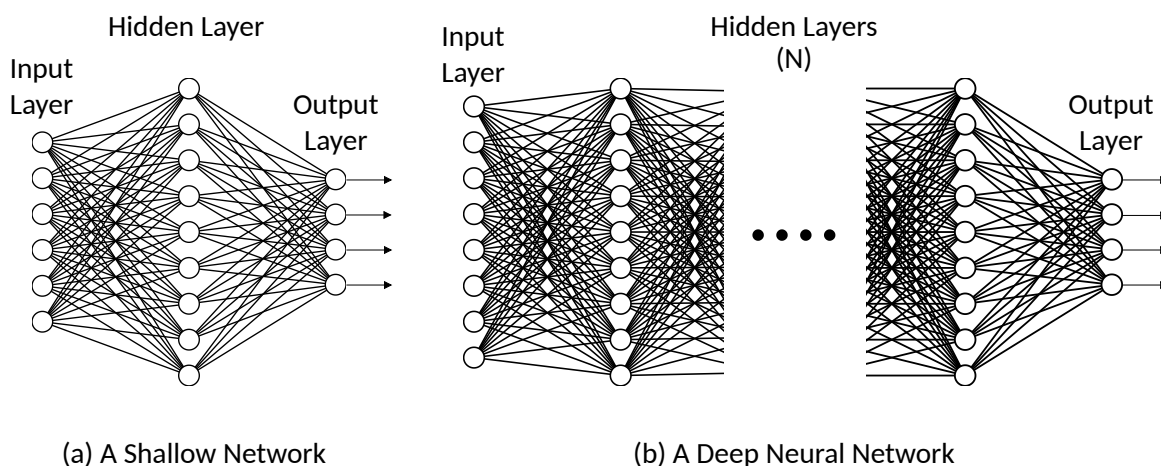


Figure 4: General architecture of a shallow network and a deep neural network.
Source: Adapted from Sarker (2021).

In regards to deep learning and spam email classification, some of the first ever usages of it for this purpose can be found in “Deep Belief Networks for Spam Filtering” by Tzortzis and Likas (2007), that provided a novel approach using Deep Belief Networks (DBNs), that are feedforward neural networks with many hidden layers. The datasets employed in the study included LingSpam, Enron Spam and SpamAssassin. The results were compared against one of the most widely accepted methods, Support Vector Machines (SVMs). The performance metrics results demonstrated that DBNs constituted a viable solution for filtering emails. It is worth noting that in the past, particularly in the 2010s, DBNs were a widely favored choice. However, they have been largely replaced by Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). Nevertheless, this newfound utilization of deep learning techniques for spam email detection highlights the potential of deep learning as a promising avenue towards achieving enhanced email classification capabilities.

1.3.3 Deep Learning vs Machine Learning

Traditional machine learning techniques were limited in their ability to process natural data in their raw form. They relied on intricate engineering and substantial domain knowledge to craft a feature extractor, converting this raw data into an appropriate internal representation or feature vector. Deep learning has achieved significant breakthroughs in addressing long-standing challenges that have resisted the best attempts of the artificial intelligence community for years. It has dramatically advanced the state-of-the-art in various domains, including speech and visual object recognition, object detection, as well as fields such as drug discovery and genomics (LeCun et al., 2015).

Although deep learning falls under the umbrella of machine learning, it exhibits notable distinctions in performance and relies on different dependencies. Some of these key properties and dependencies are (Sarker, 2021):

1. Large amounts of data, required to build a data-driven model for a particular problem domain. Deep learning techniques exhibit subpar performance when the data volume is small; conversely, they demonstrate a significant improvement over how well it does when data grows in comparison to machine learning algorithms, as shown in Figure 5.
2. High-performance machines, which are essential for deep learning algorithms due to their extensive computational requirements when training models with large datasets. These computation heavily rely on GPUs rather than CPUs. Thus, the utilization of GPU hardware is imperative for the effective implementation of deep learning techniques, specially during training.
3. Black-box perception and interpretability. In this field of study, interpretability refers to the degree of simplicity in explaining and understanding models and their functionalities. It is challenging to explain how the result of a deep learning algorithm was obtained, as the underlying process is frequently described as a “black-box”. On the other

hand, machine learning algorithms, particularly rule-based techniques, offer a transparent decision-making through explicit logic-rules, which are easily interpretable by humans.

4. Feature engineering process, which is shorter and requires less effort compared to traditional machine learning techniques. A fundamental distinction between deep learning and other machine learning algorithms lies in the attempt to extract high-level characteristics directly from data.
5. Training and execution time. Typically, training deep learning models takes a long time due to their extensive parameter count, resulting in prolonged training sessions that can extend to weeks. In contrast, machine learning algorithms generally boast significantly shorter training times, often completed within a couple of hours at most. Conversely, when it comes to testing, deep learning algorithms take extremely little time to run when compared to certain machine learning methods.

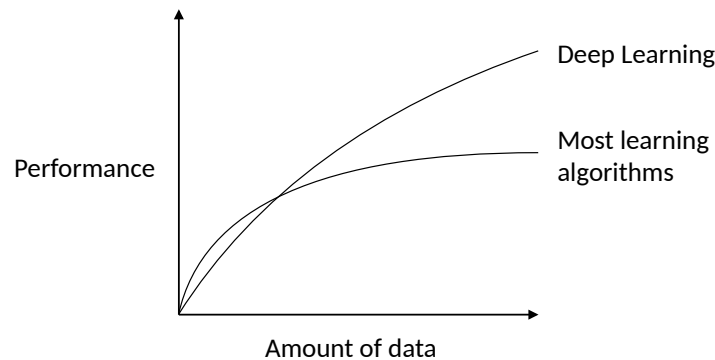


Figure 5: Performance comparison in relation to the amount of data available.
Source: Adapted from Sarker (2021).

Hence, theoretically, if there is enough data and the algorithms are executed on a sufficiently capable computer, deep learning methods are *expected* to surpass the traditional machine learning approaches in the task of classifying spam emails. The purpose of this research is to answer this assumption by analyzing and comparing the utilization of both fields in the context of spam email detection.

1.4 Identified Problems

The problems identified within the domain of spam email filtering are related to the varied, crafty and ever-evolving methods employed by scammers in their attempts to acquire sensitive information or achieve financial gains. Any misclassifications can result in the inadvertent neglect of critical emails or the unwarranted trust placed in malicious messages that find their way into the primary box. Therefore, two key issues arise; first, an individual

may be unaware of an important email incorrectly marked as spam, with consequences that vary depending on the context. Second, a person who trusts their primary inbox may fall victim to a misclassified spam email, potentially disclosing sensitive information or suffering financial losses.

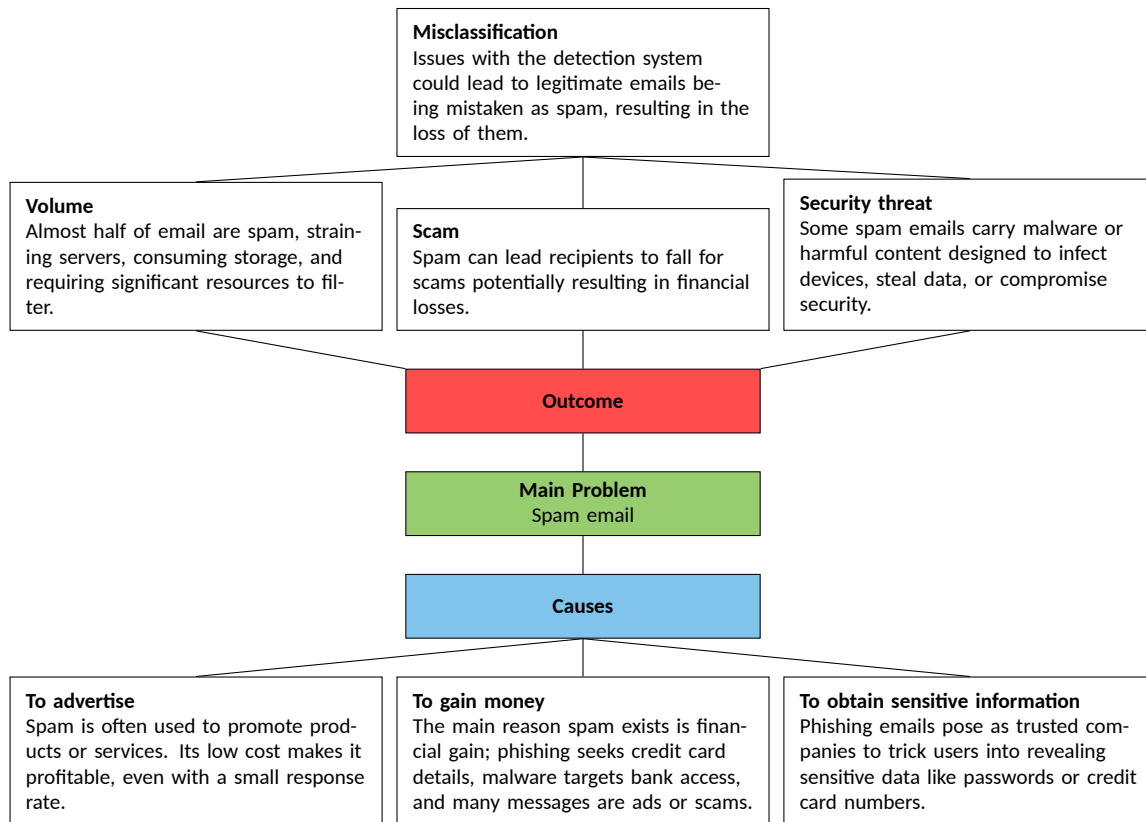


Figure 6: Problem Tree.
Source: Created by the author.

1.4.1 Techniques Used by Spammers

Spammers continuously develop sophisticated strategies to evade anti-spam filters (Wittel and Wu, 2004). They actively seek to subvert recently used classifiers by gaining information about them previously and manipulating data accordingly to induce false negatives (Dalvi et al., 2004). In addition to adapting emails to bypass detection mechanisms, spammers often attempt to cheat the receiver by imitating the look of legitimate emails, creating confusing situations like false forwards or taking advantage of spam campaigns (Oliveira et al., 2019).

1. **Poisoning text:** Some techniques involve text poisoning and word obfuscation by introducing misspellings, random words or ham-legitimate terms into spam messages

(Wang et al., 2019). For example, excluding stop words, some of the most used words in SMS ham messages are “love”, “ok”, “time”, “go” and “now”, where as common words in SMS spam messages include “call”, “free”, “txt”, “text”, “now” and “mobile”. Spammers can intentionally misspell words or inject ham words into a message to deceive the filtering model (Kuchipudi et al., 2020).

2. **Hidden text salting:** Technique used to disturb the proper behavior of textual filtering by introducing random text opportunely hidden in the email background, commonly found in phishing emails. It was started during the middle of the 2000s (Moens et al., 2010) and, while not as popular, is still being used nowadays as a strategy to evade anti-spam filters as seen in the Spam Archive dataset by Bruce Guenter (Guenter, 1998).
3. **Image-based spam email:** This technique relies on embedding the spam message within images rather than in the email body, rendering text-based processing models, such as the ones examined in this study, ineffective (Biggio et al., 2011).
4. **AI generated phishing emails:** This new technique leverages large language models to conduct personalized phishing attacks. According to Heiding et al. (2024), the AI-generated phishing emails had a click-through rates on par with human experts and were 350% better than the control group of arbitrary phishing emails. These finding suggest a likely increase in both the volume and quality of phishing emails in the coming years, making them more difficult for traditional filters to detect.

1.5 Objectives

The general objective of this research is to determine the most effective spam detection method among the options considered, providing improved classification solutions in a world inundated with unsolicited emails. The primary focus of the investigation is on junk emails and all of the various types described previously, with an exclusion of social media spam messages and phone SMS scams.

1.5.1 Specific Objectives

To achieve this aim, the specific objectives are as follows.

1. Analyze and assess a variety of deep learning methods including but not limited to Bidirectional Encoder Representation from Transformers (BERT) or Bidirectional Long Short-Term Memory (BiLSTM) with the specific goal of employing them in the task of spam email detection.
2. Evaluate the performance metrics of the models using accuracy, precision, recall, F1 score, Matthews Correlation Coefficient (MCC), Area Under the Receiver Operating

Characteristic Curve (ROC-AUC), Area Under the Precision-Recall Curve (AUC-PR), and False Positive Rate (FPR) in order to assess and compare their predictive capabilities.

3. Compare these deep learning techniques and their computed performance measurements with previously established machine learning algorithms and draw conclusions. The goal of contrasting each model is to identify the most effective solution for combating spam emails, alleviating the challenges faced by both users and service providers.

CHAPTER 2

Framework

2.1 Metrics

The subsequent discussion will address the metrics that are used to evaluate the quality of a classification model's predictions. Which parameters are most relevant depends on the specific model and task, the cost associated with different types of misclassifications, and whether the dataset is balanced or imbalanced (Google Developers, 2025a).

The confusion matrix is a tabular representation that summarizes the counts of correct and incorrect predictions made by a classification model, which provides enough information to calculate a variety of performance metrics, including precision and recall (Google Developers, 2025b).

	Actual Positive	Actual Negative
Predicted Positive	TP: True Positive (Correctly identified spam)	FP: False Positive (Non-spam misclassified as spam)
Predicted Negative	FN: False Negative (Spam misclassified as non-spam)	TN: True Negative (Correctly identified non-spam)

Table 1: Confusion matrix for binary classification.

Source: Made by the author using (Google Developers, 2025b) as reference.

2.1.1 Accuracy

Accuracy is the metric used to measure the percentage of correctly classified emails.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

Essentially, is the ratio of correct classifications to the number of total classifications. While this metric does take into account both the positive and negative results, it still is sensitive to bias (Powers, 2020). Specifically, this bias presents itself when the dataset is unbalanced, with one class containing significantly more samples than the other class (Chicco and Jurman, 2020). In this particular case, some of the datasets used are indeed considered imbalanced, which means relying on accuracy as the primary metric may lead to misleading results.

2.1.2 Precision

It is defined as the percentage of correct spam emails classified from the dataset.

$$\text{Precision} = \frac{TP}{TP + FP} . \quad (2)$$

It can also be called true positive accuracy. This parameter focuses only on the positive examples and predictions, meaning it doesn't capture any information about how well the model handles negative cases (Powers, 2020).

2.1.3 Recall

Also known as sensitivity, is the proportion of spam cases that are correctly predicted as spam. It can also be called the probability of detection (Google Developers, 2025a).

$$\text{Recall} = \frac{TP}{TP + FN} . \quad (3)$$

Similarly to precision, recall focuses on identifying positive cases correctly, without considering how well the model handles negative cases (Powers, 2020). Consequently, the model could achieve a high recall by predicting the majority of cases as positives, leading to a high number of false positives.

2.1.4 F1 Score

It's the harmonic mean of precision and recall.

$$\text{F1 Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2 \times TP}{2 \times TP + FN + FP} . \quad (4)$$

F1 is widely used in most application areas of machine learning (Chicco and Jurman, 2020). This metric balances the importance of both precision and recall, and is ideal for cases with imbalanced datasets (Google Developers, 2025a). The harmonic mean discourages a system from excessively prioritizing one metric at the expense of another (Christen et al., 2023).

It is important to acknowledge that the F1 score is not a flawless metric and has its critics, primarily due to its disregard for true negatives and its limited interpretability (Christen et al., 2023).

2.1.5 MCC

Mathews Correlation Coefficient (MCC) is a robust statistical measure for evaluating the performance of binary classification models.

$$\text{MCC} = \frac{\text{TN} \times \text{TP} - \text{FP} \times \text{FN}}{\sqrt{(\text{TN} + \text{FN})(\text{FP} + \text{TP})(\text{TN} + \text{FP})(\text{FN} + \text{TP})}}. \quad (5)$$

This metric considers all four categories of the confusion matrix: true positives, true negatives, false positives, and false negatives. By incorporating this comprehensive view, the MCC provides a balanced evaluation of a model's performance, making it particularly robust in the presence of class imbalances (Chicco and Jurman, 2020).

2.1.6 ROC-AUC

The Receiver Operating Characteristic (ROC) curve is a visual representation of a model's performance across all thresholds. It is constructed by plotting the True Positive Rate (TPR) and False Positive Rate (FPR) at every possible threshold. The Area Under the ROC Curve (ROC-AUC) quantifies the model's discriminatory power and corresponds to the probability that the model assigns a higher score to a randomly chosen positive instance than to a randomly chosen negative instance (Google Developers, 2025c).

One primary criticism of the ROC-AUC metric, particularly from the medical domain, is that it incorporates regions of the ROC space where both the TPR and FPR are insufficient, corresponding to performance levels below 50% accuracy within those intervals (Chicco and Jurman, 2020).

2.1.7 AUC-PR

Analogously to the ROC area, the Area Under the Precision-Recall Curve (AUC-PR) serves as a metric for assessing classifier performance across varying classification thresholds. This curve effectively depicts the relationship between precision and recall (Christen et al., 2023).

It offers the advantage of being particularly informative for unbalanced datasets, especially when compared to the ROC curve. Another notable benefit is its emphasis on precision, a critical metric for evaluating classifiers in the context of imbalanced datasets. Precision quantifies the proportion of correctly predicted positive instances, thereby reflecting the reliability of the classifier's positive predictions. However, there are potential challenges, especially regarding whether it shares a known limitation with the ROC area. This limitation

entails averaging of classifier performance across a distribution of classification thresholds, which may differ between classification methods (Saito and Rehmsmeier, 2015).

2.1.8 FPR

The False Positive Rate (FPR), also called fallout, is the proportion of real negatives that are incorrectly predicted to be positive (Google Developers, 2025a).

$$\text{FPR} = \frac{\text{FP}}{(\text{FP} + \text{FN})} \cdot \quad (6)$$

2.1.9 Primary Metric for Model Comparison

Drawing upon the insights from the previous subsections, the F1 score will be used as the primary metric for model comparison. Although not without limitations, it provides a clear metric that is well-suited for handling imbalanced datasets, as is the case in this study. However, all of the metrics calculated are made available to allow for comprehensive evaluation that facilitates a thorough assessment of performance differences among the models.

2.2 Understanding the Impact of Datasets on Machine and Deep Learning

A schema defines the structure of data, and a set of instances represents data that follows this structure. Typically, there is no assumed order for the instances. In most machine and deep learning applications, data is organized in a single, fixed-format table (Kohavi and Provost, 1998). In this particular instance, the datasets are extensive collections of email samples that have been pre-classified as either spam or ham, and they are used for training, testing, and validating the model. These datasets consist of the full message. Due to the nature of spam emails, they are imbalanced, with the majority of messages being non-spam.

All of the models that are run in this study use Natural Language Processing (NLP), broadly defined as the automatic manipulation of natural language, including speech and text, by software (Brownlee, 2017). In this context, it refers to the model's ability to accurately classify an email message as spam or not by *reading it* in its entirety.

2.2.1 Dataset Size

Dataset size has a significant impact in deep learning models. For example, in "Scaling Laws for Neural Language Models" by Kaplan et al. (2020) it was found that the performance of

Transformer language models is highly dependent on three factors: model size, dataset size, and the amount of compute used for training; all of which must be scaled up in tandem. Empirical performance follows a power-law relationship with each individual factor, provided it is not constrained by the other two, as shown in Figure 7. Another study on natural language processing of online reviews found that LSTM models did indeed perform better with larger datasets. While they naturally require more computational power, the authors argued that once computing power is no longer a bottleneck, LSTM models with a bigger dataset are the superior choice (Yang et al., 2020). However, these findings contrast with those of Kaplan et al. (2020), who argued that LSTM does not benefit from increased token context length in the same way Transformer models do, thereby limiting their scalability in tasks where long-range dependencies are critical.

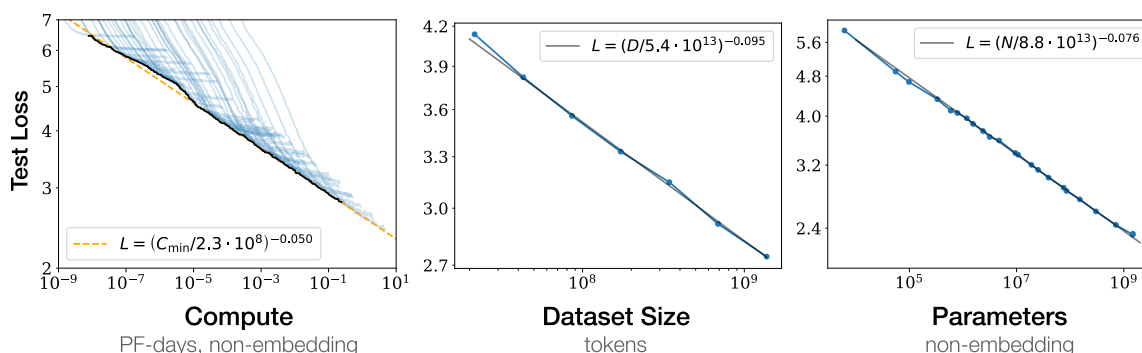


Figure 7: For Transformers, performance improves smoothly as we increase the model size, dataset size, and amount of compute used for training.

Source: Kaplan et al. (2020)

In machine learning, a paper written in 2009 suggested that the biggest successes in natural-language-related machine learning have been statistical speech recognition and statistical machine translation due to the large amount of data available for training. In fact, the authors suggested that for complex tasks, simple models trained on massive data can outperform sophisticated models on less data. It was noted how even a noisy trillion-word web corpus released by Google that came directly from unfiltered websites yielded superior results due to the sheer volume, compared to a smaller, cleaner corpus. The conclusion was to embrace the complexity of language and set aside *elegant* solutions (Halevy et al., 2009). However, this paper does not explore the relationship between the three factors mentioned in the previous paragraph.

2.2.2 Dataset Noise

Noisy data in raw text form can be considered corrupted text, misspellings or mislabeled training examples. The general consensus is that noisy data can negatively affect the performance of both machine and deep learning models.

According to Kumar et al. (2020), noisy text data could be BERT's Achilles' heel. Using the IMDB movie reviews and Stanford Sentiment Treebank (SST-2) datasets, each containing approximately 50,000 and 70,000 sentences, the authors introduced synthetic noise variants and observed a significant drop in BERT's performance. Another study comparing neural machine translation (NMT), which uses deep learning, against statistical machine translation (SMT), which relies on machine learning, found that NMT was less robust to a particular type of noise than SMT. The researchers formally identify five types of naturally occurring noise: misaligned sentences, disordered words, wrong language, short sentences, and untranslated sentences, with the latter sometimes causing neural machine translation to over-rely on copying the input (Khayrallah and Koehn, 2018).

Machine learning, conversely, was the focus of a study by Zhu and Wu (2004), examining the impact of class noise and attribute noise. After analyzing the results of 15 datasets with varying levels of noise using five classification algorithms, the expected conclusion emerged: both types of noise degrade model performance, though attribute noise is slightly less harmful. It is recommended that, even if noise handling in the test set is prohibited, cleaning attribute noise from the training set can still significantly improve overall classification accuracy.

2.2.3 Dataset Imbalance

Although already discussed throughout the paper, dataset imbalances significantly impact model performance and influence how results should be compared. Here, the bias caused by imbalances will be mitigated via using a more robust metric like F1 score. With imbalanced data, models tend to over-classify instances from the majority class due to its higher prior probability. As a result, instances from the minority group are misclassified more frequently than those from the majority group (Johnson and Khoshgoftaar, 2019). In this particular case, the models may predict spam as non-spam, since non-spam is the majority class.

It is important to note that more advanced techniques have been extensively studied in "Survey on Deep Learning with Class Imbalance" by Van Hulse et al. (2007). Data-level methods, such as over-sampling and under-sampling, include simple approaches like Random Under-Sampling (RUS), which discards random samples from the majority group, and Random Over-Sampling (ROS), which duplicates random samples from the minority group. Algorithm-level methods, which instead of modifying the distribution of training data, adjust the algorithms by adding class penalties or weights, or by shifting the decision threshold to reduce bias toward the negative class. Lastly, hybrid methods that combine data-level and algorithm-level methods (Johnson and Khoshgoftaar, 2019). These methods may differ between machine and deep learning models, but they all follow the same fundamental principles.

2.3 Cybersecurity Datasets: Characteristics and Challenges

Cybersecurity, in particular, presents unique challenges with its datasets that other fields may not encounter. In the paper “On the Effectiveness of Machine and Deep Learning for Cyber Security” by Apruzzese et al. (2018), the authors employed an assortment of machine learning and deep learning models to solve three key problems from cybersecurity: intrusion detection, malware analysis and spam detection. In the case of intrusion detection, a performance comparison between a well-known training dataset and a more recent, widely recognized dataset revealed a significantly higher F1 score and precision with the newer dataset. Furthermore, the paper discusses the difficulty of obtaining large, high-quality datasets, as most organizations are unwilling to share their internal network data. This lack of data makes periodic retraining extremely challenging or even impossible.

The authors warn against overestimating the autonomous capabilities of machine learning and conclude that continuous re-training and careful parameter tuning are essential (Apruzzese et al., 2018). A similar sentiment is expressed by Jáñez-Martino et al. (2022), hypothesizing that ignoring the fact that the spam email field is a changing environment may lead to severe degradation of generalization and performance of any model.

CHAPTER 3

Solution Proposal

3.1 Selected Datasets Based on Recency and Relevance

The three selected datasets are: The CSDMC2010 Spam Corpus², the Nazario Phishing Corpus (Nazario, 2005), and the Apache SpamAssassin Public Corpus (Apache, 2005). SpamAssassin and Nazario were chosen for their relevance, as they are widely used in spam email detection research (Kyaw et al., 2024). Most popular datasets such as Enron Spam Corpus (Metsis et al., 2006), Ling-Spam or the PU datasets (Sakkis et al., 2003) all date back to the early 2000s. Other datasets such as the TREC 2007 Public Corpus (Cormack, 2007) and CEAS 2008 (CEAS, 2008), offer similar data to that of CSDMC2010. The Nigerian Fraud/CLAIR³ (Michigan CLAIR Group, 2006) is also an outdated dataset that lacks legitimate emails, limiting its usefulness.

Dataset	Date	Total	Includes Ham?	Publicly Available?
Enron Spam	Early 2000s	31,364	Yes	Yes
Ling-Spam	Early 2000s	2,893	Yes	Yes
PU Datasets	Early 2000s	5,048	Yes	Yes
SpamAssassin	Mid 2000s	6,051	Yes	Yes
TREC 2007 Corpus	Mid 2000s	282,608	Yes	Yes
Nigerian Fraud/CLAIR	Late 1990s to 2000s	4,764	No	Yes
CEAS 2008	Mid 2000s	137,705	Yes	Yes
CSDMC2010	Late 2000s	4,110	Yes	Yes
Nazario Phishing	Mid 2000s to 2024	-	No	Yes
Guenter Spam Archive	Late 1990s to 2025	-	No	Yes
IWSPA-AP 2.0 Dataset	Late 2010s	-	Yes	No
SPEMC-15K	Early 2020s	~ 15,000	No	No

Table 2: Summary of public and private spam datasets, their content, and availability.
Source: Made by the author.

The Nazario Phishing Corpus is notable for including some of the most recent phishing emails

²Although the original link is no longer available, the dataset can still be found in a GitHub repository by Zhang (2015).

³The original link is no longer available, the dataset can still be found in some combined corpus.

among public datasets, with entries ranging from 2005 to 2024. The Guenter Spam Archive (Guenter, 1998) is another valuable dataset covering a similar time range; however, it does not appear to be widely used in recent research. Another recent dataset containing only spam is SPEMC-15K-E and SPEMC-15K-S (Jáñez-Martino et al., 2023), each containing approximately 15,000 emails in English and Spanish, respectively. Published in 2023, this dataset supports multi-class classification and, unlike the Nazario Phishing Corpus and Guenter Spam Archive, it is not publicly available. Although a newer dataset containing ham emails from this era exists, the IWSPA-AP Version 2.0 Phishing Datasets (Verma et al., 2019), it is likewise not publicly available.

	CSDMC2010	Nazario	SpamAssassin
Number of Instances	4,110	2,901	6,051
Number of Spam	1,340	2,901	1,898
Number of Ham	2,770	0	4,153

Table 3: Key details of the datasets chosen.
Source: Made by the author.

The datasets are combined into a new dataset consisting of 5,500 samples, including randomly selected emails from CSDMC2010, emails from the `easy_ham_2` and `hard_ham` subsets of SpamAssassin, and Nazario emails dated exclusively between 2015 and 2024. The final dataset is produced through a cleaning process that involves decoding Base64 content when present, removing as much HTML and noise as possible, detecting language and retaining only English emails, extracting only the email body, and removing any spam-related tags that would reveal the label. A certain level of class imbalance is maintained by including more ham emails than spam, in order to reflect real-world life distributions, as discussed in Section 1.1. See Figure 8 for a clearer visual representation of the new dataset.

This new dataset is divided into three parts:

1. The training set, `train.csv`, that is made of 2,400 ham and 2,000 spam.
2. The test set, `test.csv`, that is made of 300 ham and 250 spam.
3. The validation set, `val.csv`, that is made of 300 ham and 250 spam.

The process is repeated ten times, and the results are then used to compute the mean and standard deviation. The datasets can be found in this [GitHub repository](#).

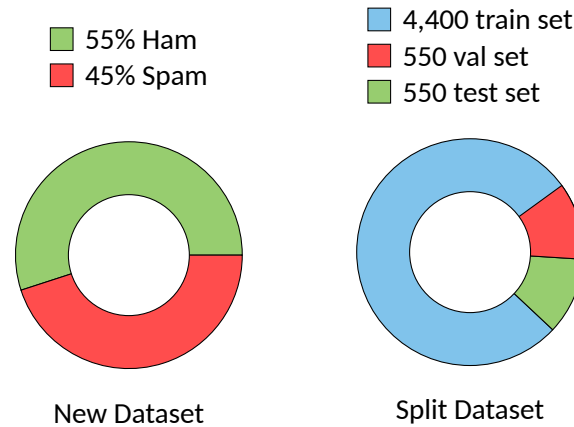


Figure 8: Donut charts showing ham/spam and split proportions of the new dataset.
Source: Made by the author.

3.2 Chosen Models Based on Literature Review

The papers were selected based on the following criteria: they must be relatively new (published between 2023 and 2025), specifically focus on text classification, not rely on external tools, and provide enough detail to support implementation or guide replication, even if not down to exact specifications. Implementation adjustments were introduced to address limitations in available computational resources. Lastly, all code will be made available in a GitHub repository.

The deep learning models selected in this study are based on the following papers: Roumeliotis et al. (2024), which includes BERT, RoBERTa, CNN, and GPT-4; Labonne and Moran (2023), which includes RoBERTa, SetFit, and Flan-T5; and Ahmed et al. (2023), which includes RNN, LSTM, and GRU. The traditional machine learning models selected are based on the following papers: Li (2023), which includes SVM, Naive Bayes, Random Forest, and KNN; and Opara et al. (2025), which includes XGBoost.

3.3 Transformer-Based Models

Transformers are a deep learning architecture based solely on attention mechanisms, replacing the recurrent layers used in encoder-decoder models with multi-headed self-attention (Vaswani et al., 2023).

3.3.1 GPT

Introduced by Radford et al. (2018), Generative Pretrained Transformer (GPT) is a family of auto-regressive language models based on the combination of two existing ideas: transformers and unsupervised pre-training with supervised fine-tuning.

3.3.2 BERT

Introduced by Devlin et al. (2019), Bidirectional Encoder Representations from Transformers (BERT) is designed to pretrain deep bidirectional representations by jointly conditioning on both left and right context across all layers, using only unlabeled text. The bidirectional nature of BERT is made possible by the Masked Language Model (MLM) objective, which alleviates the constraints of unidirectional context modeling. The other essential pretraining task in BERT is Next Sentence Prediction (NSP), which enables the model to learn inter-sentence relationships.

3.3.3 RoBERTa

Introduced by Liu et al. (2019), RoBERTa is a robustly optimized BERT approach that modifies several key hyperparameters and significantly increases the size of the training data. A significant modification was the removal of the NSP objective, coupled with the introduction of dynamic masking during pre-training.

3.3.4 SetFit

Introduced by Tunstall et al. (2022), SetFit offers an efficient and prompt-free approach to few-shot fine-tuning of sentence transformers. SetFit works by fine-tuning a pre-trained sentence transformer on a small set of text pairs in a contrastive Siamese setup, an approach that generates positive and negative pairs to effectively enlarge the training data in few-shot learning scenarios.

3.3.5 Flan-T5

Introduced by Chung et al. (2022), Flan-T5 is a fine-tuned version of T5 (Raffel et al., 2023) that relies on instruction fine-tuning, which, according to the authors, has been shown to improve model performance and generalization to unseen tasks. The model is available in various sizes, ranging from Flan-T5-Small to Flan-T5-XXL, and includes alternative architectures such as Flan-PaLM.

3.4 Convolutional Models

Convolutional networks use a combination of three key architectural principles to achieve a degree of shift, scale, and distortion invariance: local receptive fields, shared weights (or weight replication) and spatial or temporal pooling (Lecun et al., 1998).

3.4.1 CNN

Introduced by Lecun et al. (1998), Convolutional Neural Networks (CNN) are characterized by the use of convolutional networks, pooling layers (formerly known as *subsampling*), and fully connected layers.

3.5 Sequence Models

Models that use Recurrent Neural Networks (RNNs) as an architecture maintain a hidden unit whose output is fed back into the system, allowing the network to build internal representations that reflect both current and prior context (Elman, 1990).

3.5.1 RNN

The foundational works of Rumelhart et al. (1986), Rumelhart and McClelland (1987), and Williams and Zipser (1989) laid the groundwork for the development of Recurrent Neural Networks (RNNs), a class of neural networks designed for sequential data. RNNs process input sequences one element at a time, maintaining a hidden state that captures past information. Training is done via Backpropagation Through Time (BPTT), an extension of the backpropagation algorithm that propagates gradients through the unrolled sequences. However, RNNs are known to suffer from the vanishing gradient problem, making them ineffective for long sequences.

In this study, the bidirectional version of RNN is employed, BiRNN, which processes sequence data in both forward and backward directions, allowing the model to use past and future context when making predictions (Schuster and Paliwal, 1997).

3.5.2 LSTM

Introduced by Hochreiter and Schmidhuber (1997), Long Short-Term Memory (LSTM) is a type of RNN architecture that addresses the vanishing gradient problem of standard RNNs,

allowing it to learn and remember information over long sequences. It utilizes gates to regulate the flow of information into and out of the cell. The bidirectional version of LSTM, BiLSTM, is used.

3.5.3 GRU

Introduced by Cho et al. (2014), Gated Recurrent Unit (GRU) is a type of RNN architecture with fewer parameters than LSTM and features two main gates: the update gate, that decides how much of the past information needs to be passed along to the future, and the reset gate, which decides how much of the past information to forget. The bidirectional version of GRU, BiGRU, is used.

3.6 Attention Mechanisms

The sequence models in this work are used in conjunction with attention mechanisms. These mechanisms allow models to focus more intensely on a smaller number of items while ignoring others.

3.6.1 Bahdanau Attention Mechanism

Also known as additive attention, this mechanism computes alignment scores using a feed-forward neural network rather than simple dot products. At each decoding step, the current decoder hidden state is compared with each encoder hidden state to produce a score that indicates how relevant each input token is for generating the next output token (Chorowski et al., 2015).

3.6.2 Luong Attention Mechanism

Also known as multiplicative attention, this mechanism computes alignment scores using dot products, often followed by optional learned weight matrices. Unlike Bahdanau attention, Luong attention is typically applied after the decoder hidden state is computed (Luong et al., 2015).

3.6.3 Scale-Dot Product Attention Mechanism

A form of multiplicative attention that computes alignment scores using the dot product between the query and key vectors, scaled by the square root of the key dimension ($\sqrt{d_k}$). This

scaling prevents large values in the softmax input, which could otherwise lead to unstable gradients during training. Scaled dot-product attention is a core component of Transformer models.

It is also used in self-attention, where the queries, keys, and values are all derived from the same input sequence. This captures intra-sequence dependencies efficiently (Vaswani et al., 2023).

3.7 Classical Machine Learning Models

A subset of Artificial Intelligence that enables systems to learn patterns from data and make decisions (Turing, 1950). It represents a broader field that focuses on learning data, rather than relying solely on neural networks.

3.7.1 SVM

Introduced by Cortes and Vapnik (1995), Support Vector Machine (SVM) is a supervised learning model used for classification and regression tasks. It aims to find the best decision boundary, called a hyperplane, that separates data into different classes with the maximum margin.

3.7.2 Naïve Bayes

The popular paper by McCallum and Nigam (1998), describes Naïve Bayes as a probabilistic classifier based on the Bayes Theorem, with the “naïve” assumption that all features are conditionally independent given the class.

3.7.3 Random Forest

Introduced by Breiman (2001), Random Forest is an ensemble learning method that combines the predictions of many decision trees to produce more accurate and robust models. Each tree in the forest provides a classification, and the final prediction is made based on a majority vote across all trees.

3.7.4 KNN

Introduced by Cover and Hart (1967), K-Nearest Neighbors (KNN) is a supervised learning algorithm whose proximity in distance reflects similarity among data points. In classification

tasks, it predicts the label of a new instance by identifying the k nearest neighbors and assigning the majority class among them. This is done by computing the distances between the new instance and all points in the training set.

3.7.5 XGBoost

Introduced by Chen and Guestrin (2016), Extreme Gradient Boosting (XGBoost) is a modern machine learning model that provides an optimized, scalable implementation of Gradient Boosted Decision Trees (GBDTs). It builds an ensemble of trees sequentially, where each new tree corrects the errors of the previous one.

CHAPTER 4

Solution Validation

4.1 Hardware Components and Workflow

All models were run in Windows 11 using Visual Studio WSL (Ubuntu), unless stated otherwise. The system runs on a 12th Gen Intel(R) Core(TM) i5-12400F CPU (2.5 GHz, 6 cores, 12 threads), 32 GB of RAM, and an NVIDIA GeForce RTX 3060 Ti GPU. The process involves running the model on each dataset from data-1 through data-10, all of which have predefined `train.csv`, `val.csv`, and `test.csv` splits. After all 10 runs, the mean and standard deviation of the results are computed.

4.2 Deep Learning Models by Roumeliotis, Tselikas and Nasiopoulos

The models mentioned here are based entirely on the implementation presented by Roumeliotis et al. (2024). All credit for the development and design of these models goes to the original authors; for further details, please refer to their work.

The code can be found in this Github repository. This implementation should not be regarded as an exact replication.

Based on the referenced paper, the following four models are used in this work: CNN, BERT, RoBERTa, and GPT. The BERT and RoBERTa models, `bert-base-uncased` and `roberta-base`, are implemented via HuggingFace. The CNN model is made using the `tensorflow` library and includes an embedding layer with 128-dimensional vectors, a 1D convolutional layer with ReLU activation, a `GlobalMaxPooling1D` layer, a dense layer with 64 units, and a final Sigmoid activated output layer. All three models use the Adam optimizer and Binary Cross Entropy as the loss function. They are also executed on Google Colab. See Table 4 for the hyperparameters.

	Train Batch Size	Eval Batch Size	Learning Rate	Epochs
BERT	6	6	2e-5	3
RoBERTa	6	6	2e-5	3
CNN	6	6	2e-5	3

Table 4: Hyperparameters of the models.

Source: Made by the author using data provided by Roumeliotis et al. (2024).

The models achieved a variety of results, BERT and RoBERTa both performed excellently,

achieving an average F1 score of 98.44% and 98.21%, respectively, closely aligning with the authors' reported scores of 99.01% and 98.4% on the "NLP - Spam/Ham Email Classification" dataset from Kaggle. In contrast, the CNN model underperformed, with an average F1 score of 68.55% and a notably high standard deviation, compared to the originally reported 80.42%. The lowest score was 24.28% on data-7 while the highest was 87.08% on data-8. BERT and RoBERTa required around 15 minutes per dataset, while the CNN model took about one to two minutes to complete.

4.2.1 GPT Model Variants and Fine-Tuning

Instead of fine-tuning GPT-4 and comparing it to its baseline, as done in the original paper, this work uses GPT-4.1-Mini, GPT-4.1-Nano, and GPT-4o-Mini. This change was made primarily due to cost considerations. As of May 2025, the output text cost of GPT-4.1 is nearly three times higher than the combined cost of the three smaller models. The same trend applies to fine-tuning, as training with GPT-4.1 is approximately twice as expensive, and its output generation remains significantly more costly. Among the models used, GPT-4.1-Nano is the most affordable, while GPT-4.1-Mini is the priciest option, though still economical compared to GPT-4.1 (OpenAI, 2025).

```
{
  "messages": [
    {"role": "system", "content": "You are a spam filter."},
    {"role": "user", "content": "Please parse the text and classify it."},
    {"role": "assistant", "content": "{\"Spam\":0}"}
  ]
}
```

Figure 9: The structure of the JSONL files used for training, validation, and testing the GPT models follows OpenAI's format. For fine-tuning GPT-4.1-Mini and GPT-4.1-Nano, the "system" field is replaced with "developer".

Source: Made by the author.

For fine-tuning the models, new training and validation datasets were created. The training dataset consists of 24 of the shortest ham emails and 20 of the shortest spam emails from each `train.csv`, resulting in a total of 240 ham and 200 spam emails. This preserves the same class imbalance present in the training datasets used for the other models. The validation dataset was done similarly, by selecting the 12 shortest ham and 10 shortest spam emails from the first `val.csv` only. A check was performed to ensure that no emails in the new training and validation datasets are present in any of the 10 test datasets. All models were ultimately trained using 135,300 tokens. The finetuning process was around 30 minutes long.

	Epochs	Batch Size	LR	Seed
ft:gpt-4.1-mini-2025-04-14	3	1	2	1934694989
ft:gpt-4.1-nano-2025-04-14	3	1	1.8	915698810
ft:gpt-4o-mini-2024-07-18	3	1	0.1	1045943030

Table 5: The hyperparameters were not manually selected; instead, the models were fine-tuned using OpenAI's default configuration settings.

Source: Made by the author.

Each of the models took roughly the same duration to complete, averaging slightly more than five minutes. All of the finetuning models achieved better results than their baseline counterparts, with the highest F1 score being from ft:GPT-4.1-Mini with a 97.04% and a minimal standard deviation (0.059). The lowest score of all the models was GPT-4.1-Nano with an F1 score of 72.68%.

The original work by the authors not only used a different dataset but also a larger, outdated version of GPT-4, finetuned with a substantially higher number of tokens, approximately 4 million and 850,000 tokens, depending on the dataset. Specifically, with the Kaggle dataset previously mentioned, the model was fine-tuned with around 850,000 tokens and achieved an F1 score of 98.33%. This represents just over a 1% improvement compared to the results obtained in this study, highlighting the remarkable performance of the newer, more compact, and cost-efficient models. The baseline GPT-4 model achieved an F1 score of 94.21%, which is 4% higher than the baseline of the GPT-4.1-Mini.

4.3 Deep Learning Models by Labbone and Moran

The models mentioned here are based entirely on the implementation presented by Labonne and Moran (2023). All credit for the development and design of these models goes to the original authors; for further details, please refer to their work.

The code can be found in this Github repository. The code has been modified to work with the pre-existing datasets. Any modifications made to the models are detailed below.

Based on the referenced paper, the following three models are used in this work: RoBERTa, SetFit, and Flan-T5. The SetFit model is built on the sentence transformer paraphrase-minilm-l3-v2, a smaller version than the originally proposed, due to resource constraints. SetFit by default employs WordPiece tokenization and it is trained with 20 contrastive learning iterations. The Flan-T5 model used is the HuggingFace implementation `flan-t5-small`, paired with the SentencePiece tokenizer. During the tokenization process, the prefix *"classify as ham or spam"* is added. The RoBERTa model used is the HuggingFace implementation `roberta-base`, paired with byte-level Byte Pair Encoding (BPE)

tokenization. Additionally, mixed precision training with `fp16` was enabled. See Table 6 for the hyperparameters.

	Train Batch Size	Eval Batch Size	Learning Rate	Epochs
SetFit-MiniLM	16	16	2e-5	3
Flan-T5-Small	8	8	5e-5	5
RoBERTa	16	8	5e-5	10

Table 6: Hyperparameters of the models.

Source: Made by the author using data provided by Labonne and Moran (2023).

All of the models achieved excellent average F1 scores, with the results being 98.26%, 98.29% and 98.84%, respectively. These results compare favorably with those reported by the authors, 95%, 96% and 97%, using the SpamAssassin Public Corpus. Flan-T5 was the fastest, taking around 15 minutes per round, SetFit took about 1 hour per dataset, and RoBERTa, the slowest, required nearly 3 hours.

4.4 Deep Learning Models by Ahmed, Akter, Rahman, et al.

The models mentioned here are based entirely on the implementation presented by Ahmed et al. (2023). All credit for the development and design of these models goes to the original authors; for further details, please refer to their work.

The code can be found in this Github repository. This implementation should not be regarded as an exact replication.

Based on the referenced paper, the following three models are used in this work: RNN, LSTM and GRU. All of the three models are imported from the `torch` library, use the Binary Cross Entropy Loss function, use Adam as the optimizer and Sigmoid as the activation function. All models are set to `bidirectional=True`, to enable bidirectional processing. The email text sequences are truncated to a maximum length of 1,024 tokens `MAX_LEN = 1024` before being padded and fed into the model. See Table 7 for the hyperparameters.

Additionally, each of the models was tested with different attention mechanisms: the Bahdanau Attention Mechanism (BAMec), the Luong Attention Mechanism (LAMec), the Scale Dot-Product Attention Mechanism (SDP), and the SDP with Self-Attention Mechanism (SDP_self).

	Train Batch Size	Eval Batch Size	Learning Rate	Epochs
RNN	32	32	1e-3	5
LSTM	32	32	1e-3	5
GRU	32	32	1e-3	5

Table 7: Hyperparameters of the models.

Source: Made by the author using data provided by Ahmed et al. (2023).

All of the RNN models achieved good F1 scores, with the highest being the RNN combined with Bahdanau attention at 96.15%, followed closely by the RNN with SDP self-attention at 95.7%. The lowest-performing configuration was the RNN with SDP attention, with an score of 91.55%. Compared to the reported values, based on the Enron dataset, the corresponding F1 scores were 99.8%, 99.78%, 99.78%, and 99.77%.

The LSTM models achieved very good F1 scores, with the highest score being the LSTM with the Bahdanau attention at 97.04%, and a close second by the combination with SDP self-attention at 96.93%. All of the models scored above 96%. Compared to the reported values, the corresponding F1 scores were 99.73%, 99.77%, 99.76%, and 99.97%.

Lastly, just like the LSTM models all of the GRU scored above 96%, with the highest score being GRU with Luong attention at 97.02%, followed closely by the configuration with SDP at 96.92%. Compared to the reported values, the corresponding F1 scores were 99.76%, 99.77%, 99.91%, and 99.97%.

The differences between the reported values and those obtained here may be attributed to the smaller dataset used in this study, as well as the maximum input length constraint. All models except those incorporating SDP self-attention, took approximately 5 to 7 seconds to train, while the latter required between 10 to 12 seconds.

4.5 Machine Learning Models by Li

The models mentioned here are based entirely on the implementation presented by Li (2023). All credit for the development and design of these models goes to the original au-

	F1 Score
RNN-BAMec	0.9615$\pm$0.0098
RNN-LAMec	0.9284 $\pm$ 0.0214
RNN-SDP	0.9155 $\pm$ 0.0750
RNN-SDP_self	0.9570 $\pm$ 0.0170
LSTM-BAMec	0.9704$\pm$0.0087
LSTM-LAMec	0.9664 $\pm$ 0.0110
LSTM-SDP	0.9657 $\pm$ 0.0074
LSTM-SDP_self	0.9693 $\pm$ 0.0075
GRU-BAMec	0.9634 $\pm$ 0.0091
GRU-LAMec	0.9702$\pm$0.0105
GRU-SDP	0.9692 $\pm$ 0.0094
GRU-SDP_self	0.9610 $\pm$ 0.0137

Table 8: Comparison of the F1 scores of all models.

Source: Made by the author.

thor; for further details, please refer to their work.

The code can be found in this Github repository. This implementation should not be regarded as an exact replication.

Based on the referenced paper, the following five models are used in this work: SVM, Naive Bayes, Random Forest, and KNN. All models utilize the same text preprocessing pipeline; lower-casing, punctuation removal, stop-word filtering, and stemming with the `nltk` library. All of the five models used are imported from the `scikit-learn` library and vectorized using `CountVectorizer`.

Most models achieved strong F1 scores, with the exception of KNN, which underperformed relative to the others. SVM achieved the highest F1 score at 97.25%, while both Naive Bayes and Random Forest scored above 96%. The original work, using the Kaggle dataset “Spam Filter”, reported F1 scores of 99.3% for Naive Bayes, 97.5% for SVM, 97.4% for Random Forest, and 87.4% for KNN. These results were obtained using a larger test size of 40%, rather than the smaller 10% used here. This difference may partly explain why Naive Bayes performed better in the original work. All models completed training in approximately 1 to 3 seconds.

4.6 Machine Learning Model by Opara, Modesti and Golightly

The model mentioned here is based entirely on the implementation presented by Opara et al. (2025). All credit for the development and design of this model goes to the original authors; for further details, please refer to their work.

The code can be found in this Github repository. This implementation should not be regarded as an exact replication.

Based on the referenced paper, the following model is used in this work: XGBoost. The model is implemented using the `xgboost` library’s `XGBClassifier`, with a classification threshold set at 0.5. A total of 57 stylometric features are used, categorized into lexical, syntactic, punctuation, readability, word category, email-specific, complexity, and stylistic features.

The XGBoost model achieved a good F1 score of 95.79%. In comparison, the original study, using an original dataset composed of AI-generated phishing emails, reported a similar score of 96%. The model trained very fast, taking less than a second.

4.7 Comparative Summary

Table 9 presents the results of the nine models evaluated, including both versions of RoBERTa from different authors, using the eight metrics discussed in the Metrics subsection. For the RNN, LSTM, and GRU models, only the best-performing scores are reported, being RNN-

BAMec, GRU-LAMec, and RNN-BAMec. Additionally, Table 10 includes a comparison between each GPT model. The models are ordered by F1 score, from highest to lowest. Due to its size, the table is displayed in landscape format; additional graphs are provided later for clearer visualization.

As shown in Chart 10, all deep learning models, except one, achieved high F1 scores, all above 95%. The Transformer-based models reached an score of approximately 98%, while the RNN-based models ranged between 96% and 97%. In contrast, the CNN performed substantially worse, trailing the others by roughly 20%.

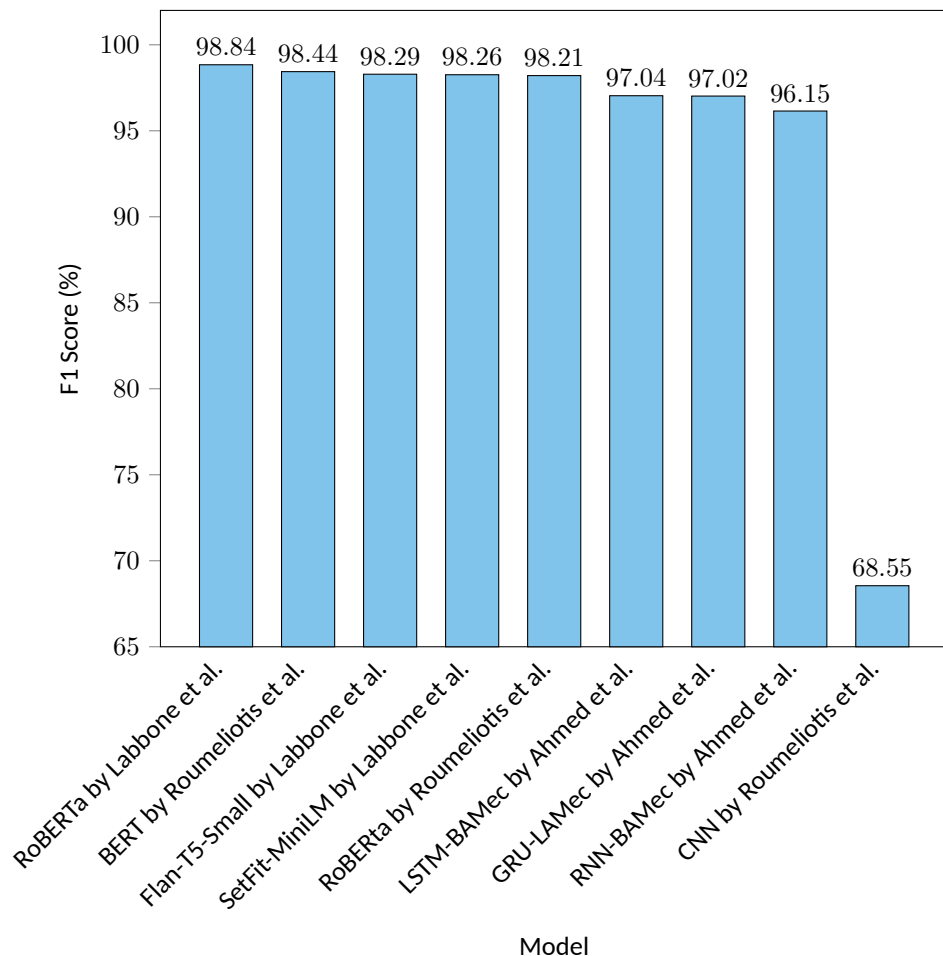


Figure 10: F1 score percentages of deep learning models on the classification task.
Source: Made by the author

As shown in Chart 11, the fine-tuned GPT models such as ft:GPT-4.1-Mini, ft:GPT-4.1-Nano, and ft:GPT-4o-Mini also perform well, all exceeding an F1 score of 95%. GPT-4.1-Mini is the final model to surpass the 90% threshold, with GPT-4o-Mini coming very close. The last model, GPT-4.1-Nano underperformed, with an F1 score of 72.68%.

	F1 Score	Accuracy	Precision	Recall	MCC	ROC-AUC	AUC-PR	FPR
RoBERTa <i>Labbone et al.</i>	0.9884$\pm$0.0062	0.9894 $\pm$ 0.0056	0.9897 $\pm$ 0.0077	0.9872 $\pm$ 0.0120	0.9788 $\pm$ 0.0112	0.9893 $\pm$ 0.0060	0.9828 $\pm$ 0.0085	0.0087 $\pm$ 0.0065
BERT <i>Roumellotis et al.</i>	0.9844 $\pm$ 0.0062	0.9858 $\pm$ 0.0057	0.9858 $\pm$ 0.0140	0.9832 $\pm$ 0.0110	0.9716 $\pm$ 0.0114	0.9856 $\pm$ 0.0056	0.9768 $\pm$ 0.0117	0.0120 $\pm$ 0.0121
Flan-T5-Small <i>Labbone et al.</i>	0.9829 $\pm$ 0.0060	0.9844 $\pm$ 0.0055	0.9772 $\pm$ 0.0092	0.9888 $\pm$ 0.0090	0.9686 $\pm$ 0.0111	0.9847 $\pm$ 0.0055	0.9713 $\pm$ 0.0097	0.0193 $\pm$ 0.0080
SetFit-MiniLM <i>Labbone et al.</i>	0.9826 $\pm$ 0.0056	0.9842 $\pm$ 0.0051	0.9821 $\pm$ 0.0081	0.9832 $\pm$ 0.0070	0.9681 $\pm$ 0.0103	0.9841 $\pm$ 0.0051	0.9732 $\pm$ 0.0091	0.0150 $\pm$ 0.0069
RoBERTa <i>Roumellotis et al.</i>	0.9821 $\pm$ 0.0114	0.9840 $\pm$ 0.0099	0.9916 $\pm$ 0.0079	0.9732 $\pm$ 0.0238	0.9681 $\pm$ 0.0196	0.9831 $\pm$ 0.0110	0.9771 $\pm$ 0.0122	0.0070 $\pm$ 0.0067
LSTM-BAMec <i>Ahmed et al.</i>	0.9704 $\pm$ 0.0087	0.9731 $\pm$ 0.0076	0.9693 $\pm$ 0.0148	0.9720 $\pm$ 0.0208	0.9461 $\pm$ 0.0152	0.9730 $\pm$ 0.0083	0.9547 $\pm$ 0.0119	0.0260 $\pm$ 0.0130
GRU-LAMec <i>Ahmed et al.</i>	0.9702 $\pm$ 0.0105	0.9729 $\pm$ 0.0096	0.9695 $\pm$ 0.0154	0.9712 $\pm$ 0.0149	0.9456 $\pm$ 0.0192	0.9728 $\pm$ 0.0096	0.9546 $\pm$ 0.0165	0.0257 $\pm$ 0.0133
RNN-BAMec <i>Ahmed et al.</i>	0.9615 $\pm$ 0.0098	0.9651 $\pm$ 0.0086	0.9637 $\pm$ 0.0184	0.9600 $\pm$ 0.0254	0.9302 $\pm$ 0.0170	0.9647 $\pm$ 0.0094	0.9430 $\pm$ 0.0134	0.0307 $\pm$ 0.0165
CNN <i>Roumellotis et al.</i>	0.6855 $\pm$ 0.2011	0.7320 $\pm$ 0.1621	0.7170 $\pm$ 0.1932	0.6632 $\pm$ 0.2175	0.4577 $\pm$ 0.3345	0.7262 $\pm$ 0.1661	0.6622 $\pm$ 0.1427	0.2106 $\pm$ 0.1343

Table 9: Performance metrics (mean $\pm$ standard deviation) across all deep learning models.
Source: Made by the author.

	F1 Score	Accuracy	Precision	Recall	MCC	ROC-AUC	AUC-PR	FPR
ft:GPT-4.1-Mini <i>Roumellotis et al.</i>	0.9704$\pm$0.0059	0.9733 $\pm$ 0.0052	0.9781 $\pm$ 0.0068	0.9628 $\pm$ 0.0102	0.9462 $\pm$ 0.0105	0.9724 $\pm$ 0.0055	0.9586 $\pm$ 0.0080	0.0180 $\pm$ 0.0057
ft:GPT-4.1-Nano <i>Roumellotis et al.</i>	0.9601 $\pm$ 0.0071	0.9645 $\pm$ 0.0060	0.9825 $\pm$ 0.0060	0.9388 $\pm$ 0.0161	0.9291 $\pm$ 0.0117	0.9624 $\pm$ 0.0068	0.9502 $\pm$ 0.0068	0.0140 $\pm$ 0.0049
ft:GPT-4o-Mini <i>Roumellotis et al.</i>	0.9575 $\pm$ 0.0065	0.9627 $\pm$ 0.0056	0.9932 $\pm$ 0.0057	0.9244 $\pm$ 0.0111	0.9263 $\pm$ 0.0109	0.9595 $\pm$ 0.0060	0.9524 $\pm$ 0.0080	0.0053 $\pm$ 0.0045
GPT-4.1-Mini <i>Roumellotis et al.</i>	0.9083 $\pm$ 0.0118	0.9200 $\pm$ 0.0098	0.9471 $\pm$ 0.0104	0.8728 $\pm$ 0.0192	0.8399 $\pm$ 0.0195	0.9161 $\pm$ 0.0105	0.8845 $\pm$ 0.0138	0.0407 $\pm$ 0.0084
GPT-4o-Mini <i>Roumellotis et al.</i>	0.8993 $\pm$ 0.0119	0.9067 $\pm$ 0.0113	0.8833 $\pm$ 0.0154	0.9160 $\pm$ 0.0122	0.8130 $\pm$ 0.0224	0.9075 $\pm$ 0.0111	0.8474 $\pm$ 0.0171	0.1010 $\pm$ 0.0146
GPT-4.1-Nano <i>Roumellotis et al.</i>	0.7268 $\pm$ 0.0275	0.7971 $\pm$ 0.0166	0.9345 $\pm$ 0.0113	0.5952 $\pm$ 0.0346	0.6154 $\pm$ 0.0307	0.7803 $\pm$ 0.0180	0.7404 $\pm$ 0.0206	0.0347 $\pm$ 0.0057

Table 10: Performance metrics (mean $\pm$ standard deviation) across GPT models.
Source: Made by the author.

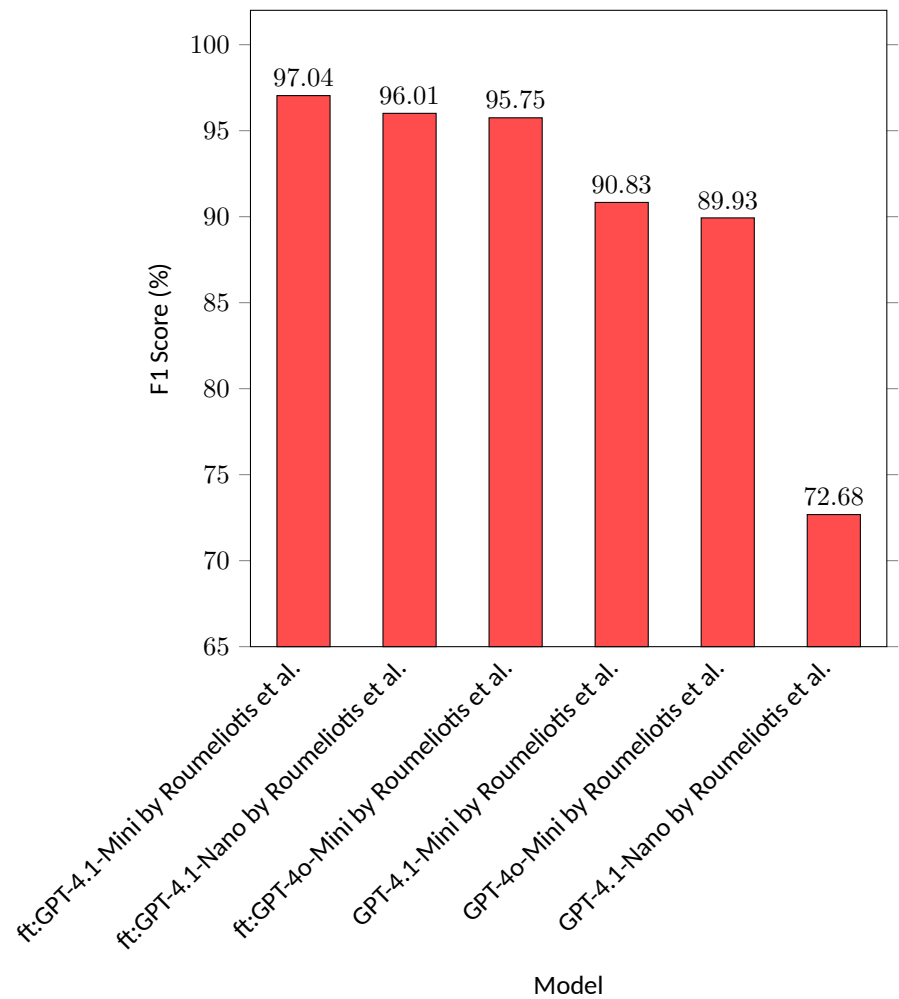


Figure 11: F1 score percentages of GPT models on the classification task.
Source: Made by the author

Table 11 presents the results of the five machine learning models evaluated. The models are ordered by F1 score, from the highest to lowest, and displayed in landscape format.

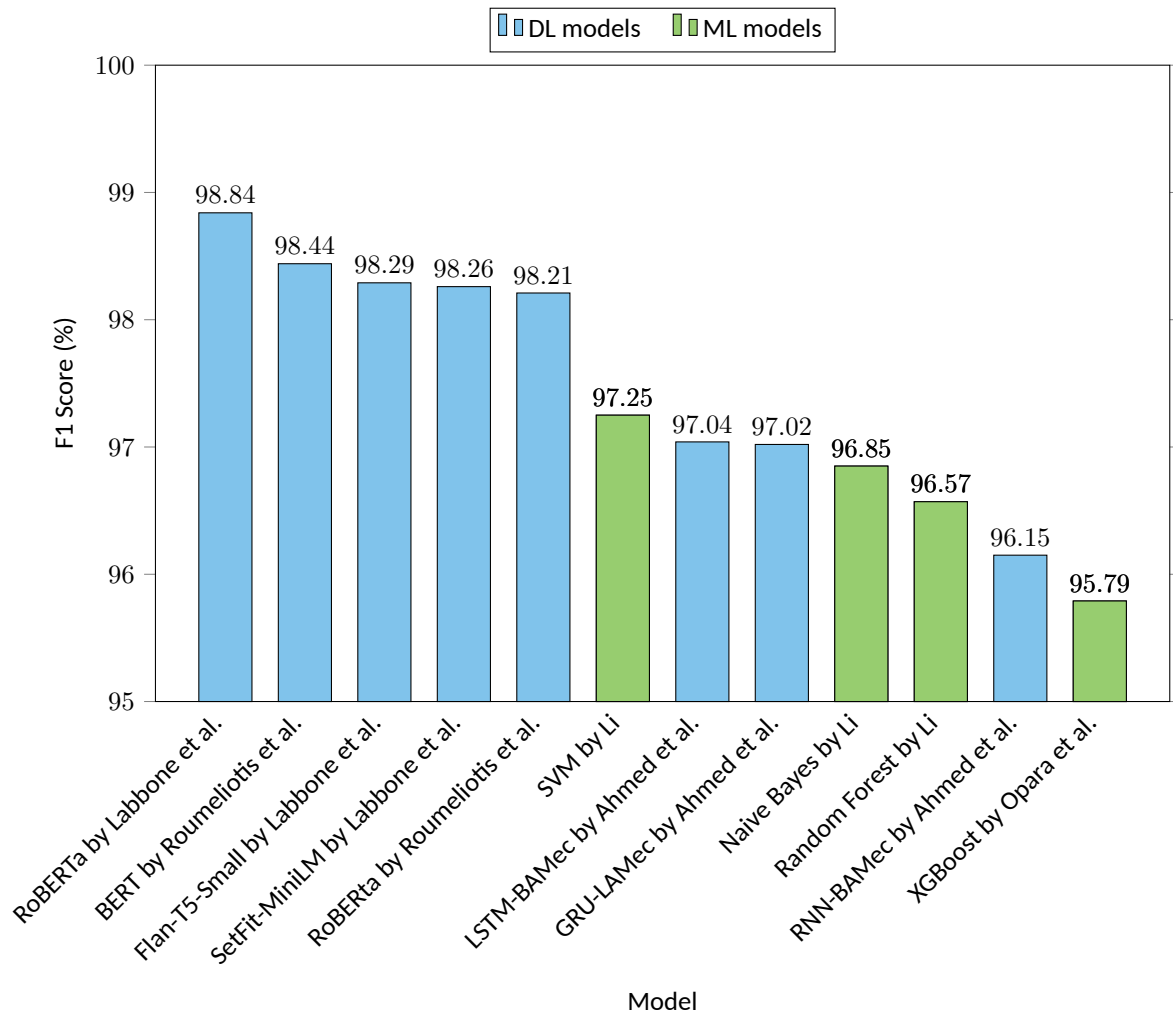


Figure 12: F1 score percentages of the top-performing DL and ML models.

Source: Made by the author

To avoid overcrowding, only models that achieved an F1 score above 95% were included in the bar chart. Refer to Figure 12 for details. Once again, the Transformer-based models were the best performing models. Among the machine learning model approaches, SVM achieved the highest performance, with a 97.25%, which gives it a gap of nearly one percent with the last Transformer-based model. However, it is important to note that the performance of the SVM model is closely followed by the RNN models, as well as by the Naive Bayes and Random Forest models.

	F1 Score	Accuracy	Precision	Recall	MCC	ROC-AUC	AUC-PR	FPR
SVM <i>Lj</i>	0.9725 ± 0.0101	0.9751 ± 0.0092	0.9759 ± 0.0134	0.9692 ± 0.0091	0.9498 ± 0.0185	0.9746 ± 0.0091	0.9599 ± 0.0163	0.0200 ± 0.0112
Naive Bayes <i>Lj</i>	0.9682 ± 0.0051	0.9711 ± 0.0046	0.9685 ± 0.0084	0.9680 ± 0.0078	0.9417 ± 0.0093	0.9940 ± 0.0022	0.9910 ± 0.0046	0.0264 ± 0.0073
Random Forest <i>Lj</i>	0.9657 ± 0.0114	0.9691 ± 0.0102	0.9748 ± 0.0126	0.9568 ± 0.0143	0.9378 ± 0.0206	0.9964 ± 0.0020	0.9956 ± 0.0026	0.0207 ± 0.0104
XGBoost <i>Opara et al.</i>	0.9579 ± 0.0114	0.9618 ± 0.0104	0.9607 ± 0.0135	0.9552 ± 0.0130	0.9230 ± 0.0209	0.9926 ± 0.0032	0.9907 ± 0.0050	0.0327 ± 0.0113
KNN <i>Lj</i>	0.8186 ± 0.0187	0.8007 ± 0.0242	0.6996 ± 0.0258	0.9868 ± 0.0068	0.6571 ± 0.0384	0.9111 ± 0.0321	0.8338 ± 0.0584	0.3543 ± 0.0419

Table 11: Performance metrics (mean $\pm$ standard deviation) across all machine learning models.
Source: Made by the author.

4.7.1 Statistical Comparison of Top-Performing Models

Given that the Transformer-based models have shown remarkably similar F1 scores, each around 98%, the **Friedman Test** was applied to determine whether the differences in performance among these algorithms are statistically significant (Friedman, 1937). The result yielded a result of 13.0722 with a p -value of 0.0109, indicating a statistically significant difference between models, as $p < 0.05$.

To find *which* models have differences among them and which one is better, the **Wilcoxon Signed-Rank Test** is performed (Wilcoxon, 1945). The results and their interpretation are presented in Table 12, where RoBERTa by Labonne et al. is referenced as “RoBERTa-AdamW”, and RoBERTa by Roumeliotis et al. is referenced as “RoBERTa-Adam”. The table shows that RoBERTa-AdamW outperforms all other models, while no significant statistical differences were found among the remaining models.

Applying the same logic to the best-performing machine learning models, the Friedman Test applied to the results of SVM, NB, and RF yields a test statistic of 1.2821, with a p -value of 0.5268, indicating no statistically significant difference among the models.

4.7.2 Controlled Comparison with GPT Models

As previously mentioned, the GPT models were trained and validated using a vastly smaller dataset, consisting of only 440 training samples and 55 validation samples. Due to this limitation, the GPT models cannot be reliably compared to the other models. To provide a fairer point of reference, the best performing model, the RoBERTa by Labbone et al., was also trained, validated, and tested in the exact same way as the models to have a better understanding of it. This comparison is shown in 13.

Notably, all fine-tuned GPT models outperformed the RoBERTa, which was trained in the same way as them, by more than 10%. Even the baseline models, GPT-4.1-Mini and GPT-4o-Mini, achieved higher F1 scores, with GPT-4.1-Mini exceeding RoBERTa’s score by over 6%. RoBERTa only outperformed the smallest baseline, GPT-4.1-Nano.

Model A	Model B	W-stat	p (2-sided)	p (A > B)	p (B > A)	Effect size r	Interpretation
RoBERTa-AdamW	SetFit-MiniLM	0.0	0.0020	0.0010	1.0000	0.8864	RoBERTa-AdamW > SetFit-MiniLM
RoBERTa-AdamW	Flan-T5-Small	5.0	0.0195	0.0098	0.9932	0.7252	RoBERTa-AdamW > Flan-T5-Small
RoBERTa-AdamW	RoBERTa-Adam	7.0	0.0663	0.0332	0.9668	0.6608	RoBERTa-AdamW > RoBERTa-Adam
RoBERTa-AdamW	BERT	5.0	0.0687	0.0344	0.9656	0.7252	RoBERTa-AdamW > BERT
SetFit-MiniLM	Flan-T5-Small	21.0	0.8587	0.5706	0.4294	0.2095	No significant difference
SetFit-MiniLM	RoBERTa-Adam	17.0	0.5147	0.7427	0.2573	0.3384	No significant difference
SetFit-MiniLM	BERT	22.0	0.6250	0.7217	0.3125	0.1773	No significant difference
Flan-T5-Small	RoBERTa-Adam	22.0	0.6250	0.7217	0.3125	0.1773	No significant difference
Flan-T5-Small	BERT	22.0	0.6250	0.7217	0.3125	0.1773	No significant difference
RoBERTa-Adam	BERT	21.0	0.8590	0.4295	0.5705	0.2095	No significant difference

Table 12: Pairwise Wilcoxon signed-rank test results comparing model performance and their interpretation.
Source: Made by the author.

	F1 Score
ft:GPT-4.1-Mini by Roumeliotis et al.	0.9704 ± 0.0059
ft:GPT-4.1-Nano by Roumeliotis et al.	0.9601 ± 0.0071
ft:GPT-4o-Mini by Roumeliotis et al.	0.9575 ± 0.0065
GPT-4.1-Mini by Roumeliotis et al.	0.9083 ± 0.0118
GPT-4o-Mini by Roumeliotis et al.	0.8993 ± 0.0119
RoBERTa (Small Sample) by Labbone et al.	0.8344 ± 0.0163
GPT-4.1-Nano by Roumeliotis et al.	0.7268 ± 0.0275

Table 13: Comparison of the F1 scores of GPT models vs RoBERTa model trained on the same sample set.

Source: Made by the author.

CHAPTER 5

Conclusions

Deep learning models demonstrated greater potential compared to their machine learning counterparts. Notably, all but one of the deep learning models scored above 95%. The one deep learning model that notably underperformed was CNN, which achieved an F1 score of only 68.55%.

Based on the results, it can be reasonably concluded that a simple CNN is a less effective option compared to traditional machine learning models, unless significant modifications or enhancements are applied. Additionally, the non-Transformer deep learning models, RNN, LSTM, and GRU, achieved solid F1 scores; however, their performance was comparable to that of the stronger traditional models, such as SVM, Naive Bayes, and Random Forest. Lastly, KNN underperformed relative to most models, but it still outperformed CNN by approximately 20%.

It is noteworthy that the deep learning models performed competitively against the machine learning models, despite the relatively small dataset, especially given the commonly held belief that deep learning models typically require more data to perform effectively (Sarker, 2021). Therefore, the strong results achieved by deep learning models on a fairly small dataset are particularly impressive. Reiterating the findings of Kaplan et al. (2020), in the case of Transformer-based architectures, performance should theoretically improve with access to larger datasets and more powerful computational resources. This suggests that these models would likely yield more precise F1 scores when trained on larger datasets.

Additionally, Kaplan et al. (2020) work highlights that Transformers benefit more from increased token context length than LSTM-based models, which further strengthens the case for their scalability and future performance, particularly in comparison to non-Transformer architectures.

These results are consistent with prior findings by Kyaw et al. (2024) in “A Systematic Review of Deep Learning Techniques for Phishing Email detection”. The authors reported that the majority of studies reviewed demonstrated that various deep learning-based techniques outperformed traditional machine learning models with promising results in phishing detection. Only three studies found that traditional machine learning models to be superior, though still highlighted notable limitations. Traditional models often struggle to generalize unseen or complex data patterns. While they are computationally inexpensive and simple to deploy, they lack the ability to learn and adapt to evolving attack patterns, an area where deep learning models show a clear advantage.

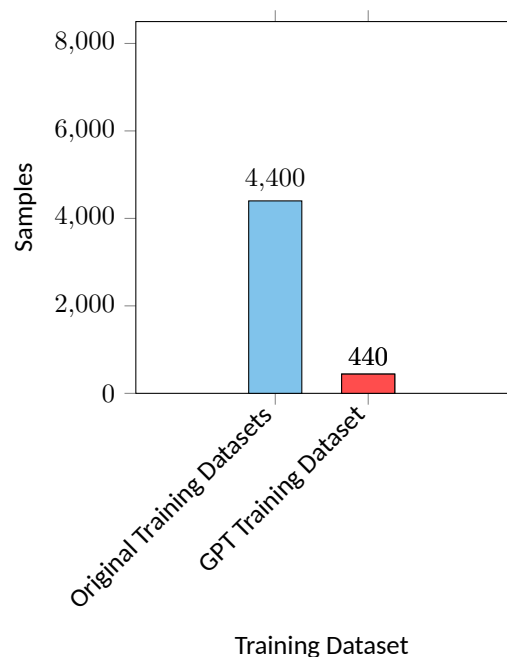


Figure 13: Comparison of training dataset sizes.
Source: Made by the author.

Additionally, while RoBERTa achieved the highest performance among the deep learning models, the GPT models yielded more intriguing results, particularly considering that, although all models were tested on the same dataset, the GPT models were trained on a dataset approximately ten times smaller due to budget constraints. In particular, the fine-tuned nano version of GPT 4.1, despite being trained on only 135,300 tokens, improved its F1 score from 72.68% to 96.01%, a substantial gain of over 24%. This made it the second-best performing GPT model, surpassed only by the fine-tuned mini version of GPT-4.1. Even then, the performance difference between the two was minimal, with only a 1% margin.

Another interesting finding when it comes to the GPT models emerges when comparing the results of this study to those originally reported by Roumeliotis et al. (2024). In their work, the fine-tuned version of GPT-4 achieved F1 scores of 99.3% and 98.33% across two separate datasets, with training comprising nearly 4 million tokens for the first dataset and approximately 850,000 tokens for the second. This study demonstrated that the newer, mini GPT 4.1 model, despite being only fine-tuned once with less than a quarter of the tokens, still provides competitive performance across 10 different test sets. The mean F1 score across these was 97.04%. This indicates that its performance was only 2% lower than achieved on the first dataset, and just over 1% lower on the second, when compared to the results reported in the original study. When RoBERTa was trained with the same dataset as the GPT models, it underperformed, achieving an F1 score of 83.4%. This further highlights how well the GPT models performed under constrained conditions and underscores their potential.

Another key observation from this study relates to the current landscape and utilization of

publicly available spam datasets. Starting with the landscape, the only two publicly available corpora that include data beyond 2010 are the Nazario Phishing Corpus and the Guenter Spam Archive; however, both include only spam emails and lack legitimate messages. It is likely that ham emails have also undergone natural changes over time. The availability of a newer, balanced dataset containing both spam and ham emails would be of significant value to the research community. However, it is important to note that creating a dataset containing both spam and ham emails is not a straightforward task and may potentially expose personal information if not properly anonymized and cleaned.

Secondly, the majority of recent studies continue to rely heavily on older datasets. All of the works discussed in the State-of-the-Art section use datasets originating from the early to mid 2000s, with the TREC 2007 Public Corpus and the Enron Spam Corpus being particularly popular. Some used Kaggle datasets that lacked details about their original sources. In the study by Kyaw et al. (2024), out of the 33 selected works, 14 used the Enron dataset, making it by far the most frequently used email corpus, followed by the Nazario Phishing Corpus with 11 mentions. Notably, the Guenter Spam Archive was not referenced in any of the selected studies.

The issues that arise with outdated datasets can be explained by the dataset shift problem discussed by Jáñez-Martino et al. (2022), which highlights that spam content evolves in unnatural ways to bypass spam filters. This dynamic nature makes it challenging to accurately assess how well a model is truly performing. It also complicates cross-study comparisons, raising the question: *how representative is a comparison between a model trained on an outdated dataset and one trained on a more recent corpus?* This concern extends beyond spam detection; similar challenges regarding dataset relevance have been identified in other areas of cybersecurity, as noted by Apruzzese et al. (2018).

Lastly, some studies did not provide F1 scores. Although this metric is becoming increasingly popular, a few works relied solely on accuracy. Reporting a comprehensive set of evaluation metrics, such as accuracy, precision, recall, and F1 score, is essential for enabling meaningful comparison between models. Including additional metrics, like those discussed in this study, further enhances the robustness of evaluation and supports more informed conclusions.

In summary, the deep learning models, particularly Transformer-based models, demonstrate better performance than the traditional machine learning models. Among them, the GPT models stand out, showing remarkable performance with significantly less training data than their counterparts. Even in their smallest configurations, these models achieved competitive F1 scores when compared to the machine learning models and to the results reported in the original studies using earlier versions of GPT.

Furthermore, the advancement of research in this field would greatly benefit from the development and adoption of newer datasets, especially those that include both spam and ham emails. However, progress does not rely solely on the creation of new corpora; it also depends on the research community's willingness to utilize the recent datasets that are already publicly available, such as the Nazario Phishing Corpus and the Guenter Spam Archive,

rather than continuing to rely on outdated, albeit popular, datasets. Equally important is the consistent inclusion of comprehensive evaluation metrics, especially the F1 score, to ensure fairness and depth in model evaluation and comparison, particularly in a field that predominantly deals with imbalanced datasets.

5.1 Limitations

One of the main limitations encountered in this study was due to computational constraints. As a result, the datasets used were limited to 5,500 samples and even then, some models had to be modified or scaled down, as their full versions exceeded the available computational resources. Access to a more powerful computing environment would enable the use of larger models, such as the original Flan-T5 and SetFit proposed, and allow the construction of larger datasets exceeding 10,000 samples.

Another limitation became evident when working with the GPT models, their usage cost significantly restricted the extent to which they could be fairly compared with other models.

Finally, the reliance on publicly available datasets necessitated the use of some outdated email samples, which may not fully reflect current spam techniques.

5.2 Future Work

Given the limitations identified, future research should be conducted using a more capable computing environment, enabling a broader evaluation of larger models and the use of substantially larger datasets. Continuing with the limitations, a larger budget would enable a more thorough evaluation of paid models and allow the incorporation of private datasets, particularly the IWSPA-AP 2.0 Dataset and the SPEMC-15K-E/SPEMC-15K-S, which could offer valuable insights, especially for assessing model performance on more diverse and up-to-date email samples. These larger datasets and more computationally intensive models would not only allow for a clearer assessment of the full potential of Transformer architectures, but also help determine whether other capable models, such as RNN, GRU, and LSTM, scale in a similar way or eventually reach a point of saturation in performance. Additionally, a more powerful computing setup would make it possible to measure training and inference times realistically, providing insights beyond metrics and helping to assess the practical efficiency of each model.

In addition to the above, further research should not only continue evaluating deep learning models and traditional machine learning models, but also explore ensemble methods. Ensemble learning models are gaining popularity, with many recent approaches combining multiple models, including hybrid configurations that integrate both machine and deep learning techniques.

Since this study focused exclusively on Natural Language Processing (NLP) and the ability of models to recognize patterns within the body of spam emails, future research should adopt a broader scope. This could include the detection of email headers, URLs, attachments, and the use of image recognition techniques such as Optical Character Recognition (OCR) to capture a wider spectrum of spam indicators. This would likely require multiple datasets for distinct purposes, a more sophisticated framework, and potentially the integration of multiple models. Also, a multilingual approach presents an intriguing direction for future work. Such an approach would not only be more practical on a global scale, but it would also provide valuable insight into which models truly excel in handling linguistic diversity.

Another interesting direction for future research is the differentiation between spam and phishing, as many studies tend to use these terms interchangeably, including this one, given that phishing is a subset of spam. While phishing emails are theoretically more challenging for filters to detect due to their targeted and deceptive nature, it would be valuable to investigate the extent of performance differences between the two categories. Such analysis could offer deeper insight into the specific threat posed by phishing emails and the effectiveness of current detection methods.

ANNEXES

For a complete understanding of spammer strategies and the dataset shift problem, please refer to the work by Jáñez-Martino et al. (2022), “A Review of Spam Email Detection: Analysis of Spammer Strategies and the Dataset Shift Problem”. The authors present experimental results demonstrating a general decline in generalization accuracy when models are applied on unseen, future data coming from datasets different from those used during training. Another valuable read is “A Systematic Review of Deep Learning Techniques for Phishing Email Detection” by Kyaw et al. (2024). Using the PRISMA approach, the authors review 33 selected papers to categorize various deep learning methods and analyze their respective strengths and limitations. Another recommended read is “On the Effectiveness of Machine and Deep Learning for Cyber Security” by Apruzzese et al. (2018), which explores the application of artificial intelligence in intrusion detection, malware analysis, and spam detection, while also cautioning against overestimating the autonomous capabilities of such algorithms. The eight metrics used in this work are based on the approach of Ghosh and Senthilrajan (2023), who reviewed different machine learning models and used these metrics to better assess their performance.

For a more comprehensive framework that captures the broader scope of spam detection, see the blog by Fernández (2024). It is worth noting, however, that the content is in Spanish.

To explore both the theoretical and experimental foundations of scalability in neural networks, refer to “Scaling Laws for Neural Language Models” by Kaplan et al. (2020), which offers an interesting insight into the performance potential of Transformer language models.

Lastly, for a deeper understanding of the history of the F-measure, including its advantages and disadvantages, refer to “A Review of the F-Measure: Its History, Properties, Criticism, and Alternatives” by Christen et al. (2023).

REFERENCES

Isra'a AbdulNabi and Qussai Yaseen. Spam email detection using deep learning techniques. *Procedia Computer Science*, 184:853–858, 2021. ISSN 1877-0509. doi: <https://doi.org/10.1016/j.procs.2021.03.107>. URL <https://www.sciencedirect.com/science/article/pii/S1877050921007493>. The 12th International Conference on Ambient Systems, Networks and Technologies (ANT) / The 4th International Conference on Emerging Data and Industry 4.0 (EDI40) / Affiliated Workshops.

Md. Tofael Ahmed, Mariam Akter, Md. Saifur Rahman, Maqsudur Rahman, Pintu Chandra Paul, Nargis Parvin, and Almas Hossain Antar. Deep neural network based spam email classification using attention mechanisms. *Journal of Intelligent Learning Systems and Applications*, 15(4):144–164, 2023. doi: 10.4236/jilsa.2023.154010.

SpamAssassin Apache. Spamassassin public corpus. <https://spamassassin.apache.org/old/publiccorpus/>, 2005. Accessed: 2025-05-09.

Giovanni Apruzzese, Michele Colajanni, Luca Ferretti, Alessandro Guido, and Mirco Marchetti. On the effectiveness of machine and deep learning for cyber security. In *2018 10th International Conference on Cyber Conflict (CyCon)*, pages 371–390, 2018. doi: 10.23919/CYCON.2018.8405026.

Battista Biggio, Giorgio Fumera, Ignazio Pillai, and Fabio Roli. A survey and experimental evaluation of image spam filtering techniques. *Pattern Recognition Letters*, 32(10):1436–1446, 2011. ISSN 0167-8655. doi: <https://doi.org/10.1016/j.patrec.2011.03.022>. URL <https://www.sciencedirect.com/science/article/pii/S0167865511000936>.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. doi: 10.1023/A:1010933404324. URL <https://link.springer.com/article/10.1023/A:1010933404324>.

J. Brownlee. *Deep Learning for Natural Language Processing: Develop Deep Learning Models for your Natural Language Problems*. Machine Learning Mastery, 2017. URL https://books.google.cl/books?id=_pmoDwAAQBAJ.

CEAS. Ceas 2008 spam track public corpus. <https://www.ceas.cc/2008/>, 2008. Accessed: 2025-05-29.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 785–794, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342322. doi: 10.1145/2939672.2939785. URL <https://doi.org/10.1145/2939672.2939785>.

Davide Chicco and Giuseppe Jurman. The advantages of the matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics*, 21(1):6, January 2020.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014. URL <https://arxiv.org/abs/1406.1078>.

Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-based models for speech recognition, 2015. URL <https://arxiv.org/abs/1506.07503>.

Peter Christen, David J. Hand, and Nishadi Kirielle. A review of the f-measure: Its history, properties, criticism, and alternatives. *ACM Comput. Surv.*, 56(3), October 2023. ISSN 0360-0300. doi: 10.1145/3606367. URL <https://doi.org/10.1145/3606367>.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models, 2022. URL <https://arxiv.org/abs/2210.11416>.

Gordon V. Cormack. Trec 2007 public spam corpus. <https://plg.uwaterloo.ca/~gvcormac/treccorpus07/>, 2007. Accessed: 2025-05-29.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995. doi: 10.1007/BF00994018. URL <https://link.springer.com/article/10.1007/BF00994018>.

T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967. doi: 10.1109/TIT.1967.1053964.

Lorrie Faith Cranor and Brian A. LaMacchia. Spam! *Commun. ACM*, 41(8):74–83, aug 1998. ISSN 0001-0782. doi: 10.1145/280324.280336. URL <https://doi.org/10.1145/280324.280336>.

Nilesh Dalvi, Pedro Domingos, Mausam, Sumit Sanghai, and Deepak Verma. Adversarial classification. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, page 99–108, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 1581138881. doi: 10.1145/1014052.1014066. URL <https://doi.org/10.1145/1014052.1014066>.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. URL <https://arxiv.org/abs/1810.04805>.

Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990. ISSN 0364-0213. doi: [https://doi.org/10.1016/0364-0213\(90\)90002-E](https://doi.org/10.1016/0364-0213(90)90002-E). URL <https://www.sciencedirect.com/science/article/pii/036402139090002E>.

Santiago Agustín Fernández. Analizando correos con inteligencia artificial. <https://blog.santiagoagustinfernandez.com/analizando-correos-con-inteligencia-artificial>, 2024. URL <https://blog.santiagoagustinfernandez.com/analizando-correos-con-inteligencia-artificial>. Publicado en el blog personal de Santiago Agustín Fernández.

Emilio Ferrara. The history of digital spam. *Communications of the ACM*, 62:82–91, 07 2019. doi: 10.1145/3299768.

Milton Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701, 1937. doi: 10.1080/01621459.1937.10503522.

Argha Ghosh and A. Senthilrajan. Comparison of machine learning techniques for spam detection. *Multimedia Tools and Applications*, 82(19):29227–29254, Aug 2023. ISSN 1573-7721. doi: 10.1007/s11042-023-14689-3. URL <https://doi.org/10.1007/s11042-023-14689-3>.

Google Developers. Accuracy, precision, and recall. <https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall>, 2025a. Accessed: 2025-06-26.

Google Developers. Confusion matrix. https://developers.google.com/machine-learning/glossary#confusion_matrix, 2025b. Accessed: 2025-06-26.

Google Developers. Classification: Roc and auc. <https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc>, 2025c. Accessed: 2025-06-26.

Bruce Guenter. Spam archive, 1998. URL <https://untroubled.org/spam/>.

Alon Halevy, Peter Norvig, and Fernando Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2):8–12, 2009. doi: 10.1109/MIS.2009.36.

Fred Heiding, Simon Lermen, Andrew Kao, Bruce Schneier, and Arun Vishwanath. Evaluating large language models' capability to launch fully automated spear phishing campaigns: Validated on human subjects, 2024. URL <https://arxiv.org/abs/2412.00586>.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 11 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL <https://doi.org/10.1162/neco.1997.9.8.1735>.

Fahima Hossain, Mohammed Uddin, and Rajib Halder. Analysis of optimized machine learning and deep learning techniques for spam detection. pages 1–7, 04 2021. doi: 10.1109/IEMTRONICS52119.2021.9422508.

Tom N. Jagatic, Nathaniel A. Johnson, Markus Jakobsson, and Filippo Menczer. Social phishing. *Commun. ACM*, 50(10):94–100, October 2007. ISSN 0001-0782. doi: 10.1145/1290958.1290968. URL <https://doi.org/10.1145/1290958.1290968>.

Justin M. Johnson and Taghi M. Khoshgoftaar. Survey on deep learning with class imbalance. *Journal of Big Data*, 6(1):27, December 2019. ISSN 2196-1115. doi: 10.1186/s40537-019-0192-5. URL <https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0192-5>.

Francisco Jáñez-Martino, Rocío Alaiz, Víctor González-Castro, Eduardo Fidalgo, and Enrique Alegre. A review of spam email detection: analysis of spammer strategies and the dataset shift problem. *Artificial Intelligence Review*, 56, 05 2022. doi: 10.1007/s10462-022-10195-4.

Francisco Jáñez-Martino, Rocío Alaiz-Rodríguez, Víctor González-Castro, Eduardo Fidalgo, and Enrique Alegre. Classifying spam emails using agglomerative hierarchical clustering and a topic-based approach. *Applied Soft Computing*, 139:110226, 2023. ISSN 1568-4946. doi: <https://doi.org/10.1016/j.asoc.2023.110226>. URL <https://www.sciencedirect.com/science/article/pii/S1568494623002442>.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *CoRR*, abs/2001.08361, 2020. URL <https://arxiv.org/abs/2001.08361>.

Huda Khayrallah and Philipp Koehn. On the impact of various types of noise on neural machine translation. In Alexandra Birch, Andrew Finch, Thang Luong, Graham Neubig, and Yusuke Oda, editors, *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 74–83. Association for Computational Linguistics, July 2018. doi: 10.18653/v1/W18-2709. URL <https://aclanthology.org/W18-2709/>.

Ron Kohavi and Foster Provost. Glossary of terms. *Machine Learning*, 30(2):271–274, 1998. doi: 10.1023/A:1017181826899. URL <https://doi.org/10.1023/A:1017181826899>.

Bhargav Kuchipudi, Ravi Teja Nannapaneni, and Qi Liao. Adversarial machine learning for spam filters. In *Proceedings of the 15th International Conference on Availability, Reliability and Security*, ARES '20, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450388337. doi: 10.1145/3407023.3407079. URL <https://doi.org/10.1145/3407023.3407079>.

Ankit Kumar, Piyush Makhija, and Anuj Gupta. Noisy text data: Achilles' heel of bert, 2020. URL <https://arxiv.org/abs/2003.12932>.

Kumaran. Spam does not bring us joy-ridding gmail of 100 million more spam messages with tensorflow. <https://workspace.google.com/blog/product-announcements/ridding-gmail-of-100-million-more-spam-messages-with-tensorflow>, 2019. Accessed: 2023-08-10.

Phyo Htet Kyaw, Jairo Gutierrez, and Akbar Ghobakhlou. A systematic review of deep learning techniques for phishing email detection. *Electronics*, 13(19), 2024. ISSN 2079-9292. doi: 10.3390/electronics13193823. URL <https://www.mdpi.com/2079-9292/13/19/3823>.

Maxime Labonne and Sean Moran. Spam-t5: Benchmarking large language models for few-shot email spam detection, 2023. URL <https://arxiv.org/abs/2304.01238>.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Yann LeCun, Y. Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–44, 05 2015. doi: 10.1038/nature14539.

Yixuan Li. Algorithm comparison on email spam filtering task. *Highlights in Science, Engineering and Technology*, 34:143–148, 02 2023. doi: 10.54097/hset.v34i.5436.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pre-training approach, 2019. URL <https://arxiv.org/abs/1907.11692>.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation, 2015. URL <https://arxiv.org/abs/1508.04025>.

Andrew McCallum and Kamal Nigam. A comparison of event models for naive bayes text classification. In *Learning for Text Categorization: Papers from the 1998 AAAI Workshop*, volume 752, pages 41–48. AAAI Press, 1998. URL <http://www.kamalnigam.com/papers/multinomial-aaaiws98.pdf>.

Merriam-Webster. Spam. <https://www.merriam-webster.com/dictionary/spam>, 2025. Accessed: 2025-05-22.

Vangelis Metsis, Ion Androutsopoulos, and Georgios Paliouras. Spam filtering with naive bayes—which naive bayes? In *CEAS*, volume 17, pages 28–69. Mountain View, CA, 2006.

Michigan CLAIR Group. Nigerian scam email corpus. <http://clair.si.umich.edu/clair/nigerian-fraud/>, 2006. dead link.

Marie-Francine Moens, Jan De Beer, Erik Boiy, and Juan Carlos Gomez. Identifying and resolving hidden text salting. *IEEE Transactions on Information Forensics and Security*, 5(4): 837–847, 2010. doi: 10.1109/TIFS.2010.2063024.

M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning, second edition*. Adaptive Computation and Machine Learning series. MIT Press, 2018. ISBN 9780262351362. URL <https://books.google.ca/books?id=dWB9DwAAQBAJ>.

Jose Nazario. Phishing email corpus. <https://monkey.org/~jose/phishing/>, 2005. Accessed: 2025-05-09.

Daniela Seabra Oliveira, Tian Lin, Harold Rocha, Donovan Ellis, Sandeep Dommaraju, Huizi Yang, Devon Weir, Sebastian Marin, and Natalie C. Ebner. Empirical analysis of weapons of influence, life domains, and demographic-targeting in modern spam: an age-comparative perspective. *Crime Science*, 8(1):3, 2019. doi: 10.1186/s40163-019-0098-8. URL <https://link.springer.com/article/10.1186/s40163-019-0098-8>.

Chidimma Opara, Paolo Modesti, and Lewis Golightly. Evaluating spam filters and stylistic detection of ai-generated phishing emails. *Expert Syst. Appl.*, 276(C), May 2025. ISSN 0957-4174. doi: 10.1016/j.eswa.2025.127044. URL <https://doi.org/10.1016/j.eswa.2025.127044>.

OpenAI. Openai platform: Model documentation. <https://platform.openai.com/docs/models>, 2024. Accessed: 2025-05-08.

OpenAI. Api pricing. <https://platform.openai.com/docs/pricing>, 2025. Accessed: 2025-05-14.

Petrosyan. Global spam volume as percentage of total e-mail traffic from 2011 to 2022. <https://www.statista.com/statistics/420400/spam-email-traffic-share-annual/#:~:text=Spam%20e,2023%20In%202022%2C%20nearly%2049,2022>. Accessed: 2023-11-24.

David M W Powers. Evaluation: from precision, recall and f-measure to ROC, informedness, markedness and correlation. *arXiv [cs.LG]*, 2020.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018. URL https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf. OpenAI.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2023. URL <https://arxiv.org/abs/1910.10683>.

Konstantinos I. Roumeliotis, Nikolaos D. Tselikas, and Dimitrios K. Nasiopoulos. Next-generation spam filtering: Comparative fine-tuning of llms, nlps, and cnn models for email spam classification. *Electronics*, 13(11), 2024. ISSN 2079-9292. doi: 10.3390/electronics13112034. URL <https://www.mdpi.com/2079-9292/13/11/2034>.

David E. Rumelhart and James L. McClelland. *Learning Internal Representations by Error Propagation*, pages 318–362. 1987.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986. doi: 10.1038/323533a0.

Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. A bayesian approach to filtering junk e-mail. *AAAI'98 Workshop on Learning for Text Categorization*, 62, 06 1998.

Takaya Saito and Marc Rehmsmeier. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PLOS ONE*, 10(3): 1–21, 03 2015. doi: 10.1371/journal.pone.0118432. URL <https://doi.org/10.1371/journal.pone.0118432>.

Georgios Sakkis, Ion Androutsopoulos, Georgios Paliouras, Vangelis Karkaletsis, Constantine D. Spyropoulos, and Panagiotis Stamatopoulos. A memory-based approach to anti-spam filtering for mailing lists. *Information Retrieval*, 6(1):49–73, 2003. doi: 10.1023/A:1022948414856. URL <https://link.springer.com/article/10.1023/A:1022948414856>.

Iqbal H. Sarker. Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions. *SN Computer Science*, 2(6):420, Aug 2021. ISSN 2661-8907. doi: 10.1007/s42979-021-00815-1. URL <https://doi.org/10.1007/s42979-021-00815-1>.

M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997. doi: 10.1109/78.650093.

Rimitha Shajahan and P. L. Lekshmy. Hybrid learning approach for e-mail spam detection and classification. In *Intelligent Cyber Physical Systems and Internet of Things (ICoICI 2022)*, volume 3 of *Engineering Cyber-Physical Systems and Critical Infrastructures*, pages 781–794. Springer, 2023. doi: 10.1007/978-3-031-18497-0_57. URL https://link.springer.com/chapter/10.1007/978-3-031-18497-0_57.

Lewis Tunstall, Nils Reimers, Unso Eun Seo Jo, Luke Bates, Daniel Korat, Moshe Wasserblat, and Oren Pereg. Efficient few-shot learning without prompts, 2022. URL <https://arxiv.org/abs/2209.11055>.

Alan M. Turing. Computing machinery and intelligence. *Mind*, 49(236):433–460, 1950. doi: 10.1093/mind/LIX.236.433. URL <https://courses.cs.umbc.edu/471/papers/turing.pdf>.

Grigorios Tzortzis and Aristidis Likas. Deep belief networks for spam filtering. In *19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007)*, volume 2, pages 306–309, 2007. doi: 10.1109/ICTAI.2007.65.

Jason Van Hulse, Taghi M. Khoshgoftaar, and Amri Napolitano. Experimental perspectives on learning from imbalanced data. *ICML '07*, page 935–942, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595937933. doi: 10.1145/1273496.1273614. URL <https://doi.org/10.1145/1273496.1273614>.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL <https://arxiv.org/abs/1706.03762>.

Rakesh M. Verma, Victor Zeng, and Houtan Faridi. Data quality for security challenges: Case studies of phishing, malware and intrusion detection datasets. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS '19*, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450367479. doi: 10.1145/3319535.3363267. URL <https://doi.org/10.1145/3319535.3363267>.

Xianmin Wang, Jing Li, Xiaohui Kuang, Yu an Tan, and Jin Li. The security of machine learning in an adversarial setting: A survey. *Journal of Parallel and Distributed Computing*, 130:12–23, 2019. ISSN 0743-7315. doi: <https://doi.org/10.1016/j.jpdc.2019.03.003>. URL <https://www.sciencedirect.com/science/article/pii/S0743731518309183>.

Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6): 80–83, 1945. doi: 10.2307/3001968. URL <https://www.jstor.org/stable/3001968>.

Ronald J. Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280, 1989. doi: 10.1162/neco.1989.1.2.270.

Gregory Wittel and Shyhtsun Wu. On attacking statistical spam filters. 01 2004.

Shudong Yang, Xueying Yu, and Ying Zhou. Lstm and gru neural network performance comparison study: Taking yelp review dataset as an example. pages 98–101, 06 2020. doi: 10.1109/IWECAI50956.2020.00027.

Renzhi Zhang. Spam-email-classifier-dataset. <https://github.com/zrz1996/Spam-Email-Classififer-DataSet>, 2015. Accessed: 2025-05-09.

Xingquan Zhu and Xindong Wu. Class noise vs. attribute noise: A quantitative study. *Artificial Intelligence Review*, 22(3):177–210, Nov 2004. ISSN 1573-7462. doi: 10.1007/s10462-004-0751-8. URL <https://doi.org/10.1007/s10462-004-0751-8>.