An Analysis of Supervised Machine Learning Algorithms for Email Classification Employing Natural Language Processing Techniques

Md. Mohidul Hasan
Computer Science(Software Engineering)
University of Hertfordshire
London , UK
shahan.hasan101294@gmail.com

Syed Mahbub Uz Zaman
Computer Science & Engineering
BRAC University
Dhaka, Bangladesh
syed.mahbub.uz.zaman@g.bracu.ac.bd

Md. Asif Talukdar
Computer Science & Engineering
BRAC University
Dhaka, Bangladesh
md.asif.talukdar@g.bracu.ac.bd

Ayesha Siddika
Computer Science & Engineering
BRAC University
Dhaka, Bangladesh
ayesha.siddika2@g.bracu.ac.bd

Md. Golam Rabiul Alam
Computer Science & Engineering
BRAC University
Dhaka, Bangladesh
rabiul.alam@bracu.ac.bd

***Abstract*—Due to the extensive use of technology in our daily lives, email has become essential for online correspondence between individuals from all walks of life. As such certain individuals have weaponized this service by bulk mailing malicious emails to recipients with the goal of retrieving some form of classified information. Thus, Email classification has become a major area of interest for researchers as it enables identification and isolation of such malicious emails. The objectives of this paper include a robust comparison of several traditional machine learning algorithms such as Multinomial Naive Bayes etc, exploring transfer learning with static (non-trainable) GLOVE (Global word vector representation) embedding, comparison of several deep learning algorithms such as ANN (Artificial Neural Network), Bi-LSTM (Bidirectional Long short-term memory) and CNN (Convolutional neural network) trained with static GLOVE and keras embedding separately. All models and classifiers were evaluated employing appropriate performance metrics such as Accuracy, Recall, Precision, F1-score and Area under curve for Receiver Operating Characteristics (AU-ROC). The analysis of the performance metrics show that XGBoost achieves the highest accuracy, AU-ROC, f1-score and recall respectively. Among Deep learning algorithms, models with keras embedding achieves the highest evaluation scores compared to models with pretrained GLOVE Embedding (static). In Conclusion ANN, CNN and Bi-LSTM with Keras Embedding out-perform all other models and classifiers. However, CNN and Bi-LSTM models with static GLOVE embedding achieve extremely high evaluation scores which shows the efficiency of transfer learning in downstream NLP tasks like email classification.**

***Index Terms*—XGBoost, Transfer Learning, Artificial Neural Network, Convolutional Neural Network & Bi-directional Long short term memory.**

I. INTRODUCTION

Electronic mails (emails) play a significant role in day-to-day communication for a wide variety of professionals and businesses alike. Approximately A total of 319 billion emails are being sent and received per day in 2021 and this number is likely to grow over 376 billion by the end of 2025 according to email statistics report 2021 by RADICATI group [1]. As such malicious actors have begun using unsolicited emails to exploit users, customers or professionals of particular businesses. Despite the use of several machine learning and deep learning technologies in spam email detection, the proportion of spam emails in total email traffic remains enormous [1], [2]. Statista states that a total of 45.1 percent of all emails exchanged in March, 2021 is identified as spam [3]. Traditionally, rules and protocol-based systems were employed to identify spam and phishing emails [4], [5]. These rule-based systems were static in nature rendering them ineffective against modern spam and phishing attempts [6]. Malicious attackers are growing more versatile in circumventing existing email filters as computational resources become more widely available. As such various machine learning based spam email detection systems have been proposed in the existing literature. Our work focuses on comparing and contrasting the efficacy's of several spam email classification algorithms, such as machine learning classifiers and deep neural networks using natural language processing algorithms. The primary contributions of this paper include:

1) A comparison of the machine learning classifiers trained on identical experimental conditions using appropriate performance metrics.
2) Exploring transfer learning in training deep learning models (GLOVE embedding based models).
3) A comparison of Deep learning models trained on identical experimental conditions using appropriate performance metrics.

This study is structured as follows. The report begins with a review of existing literature related to Spam email classification in chapter 2. Chapter 3 describes the working principle followed within this study as well as it provides a general outline to how most machine learning based classification systems are constructed. Chapter 4 provides a concise analysis of the results obtained within the study. Specifically, it addresses the research objective of the study relating to the performance metrics of the models trained. Chapter 5 provides conclusions drawn from chapter 4 and outlines future work relating to this study.

II. LITERATURE REVIEW

A. Related Work

In their paper [7], I. AbdulNabi et al. trained several Machine learning (ML) and Deep learning (DL) models such as K-NN (K-nearest neighbour), NB (Naive bayes), Bi-LSTM (Bi directional Long short-term memory) and Google BERT for email classification. These models were evaluated using Accuracy and f1-score. The results show that Bert out performed all the other models with an accuracy of 97.30% and an F1-score of 96.96%.

S. Srinivasan et al. in their paper [8], explored 3 Deep Convolutional Neural Network (DCNN) architectures as well as popular pretrained CNN architectures such as VGG29, Xception etc by means of transfer learning to classify spam images. The authors used several Image spam data-sets namely Image spam hunter data-set, an improved data-set developed by authors of [9] and Dredze ImageSpam data-set. These models were evaluated using accuracy and f1-score. The authors show that DCNN architectures obtained better performance metrics compared to the pretrained CNN models.

In their paper [10] S. Ishik et al. explored several Recurrent Neural Network architectures for email classification on Agglutinative Language like Turkish. The data-set was collected from [11]. The authors trained several ML and DL based models namely Artificial Neural Network (ANN or MLP - Multi layer perceptron), LSTM, Bi-LSTM with MI (Mutual Information) and WMI (Weighted Mutual Information) as feature selection. The authors show that some of the models have received an accuracy of 100% which could be attributed to the scarcity of available data as well as over-fitting DNN models.

Ankit Narendrakumar in his paper [12], explored the efficacy of D-CNN algorithms for email classification. The authors used Enron and spam assassin data-sets to train the DCNN models. Finally, the author proposed THEMIS, an email classification model based on a mathematical approach where by the emails are divided into several sections and complex functions are employed to extract and classify the email signature. The models were evaluated using accuracy and f1-score. The proposed THEMIS model achieved an accuracy of 99.84%.

Alia. Barushka et al. in their paper [13], reviewed a spam classification system based on DNN and word embeddings. The data-sets used by the authors include Cornell University positive hotel review spam and negative hotel review spam and TripAdvisor (Amazon Mechanical Turk). The models selected by the authors include CNN, NB, SVM, Random Forest with ngram and skip gram word representation models. These models were evaluated using accuracy, AU-ROC, FN and FP. ANN and CNN models with a combination of ngram , skip-gram word representation out performed all the other models with an accuracy of 88.38% on the Negative data-set and 89.75% on the Positive data-set.

In their paper [14] Feng Wei et al. proposed a Bi-LSTM with GLOVE word Embedding to detect twitter bots. Cresci-2017 twitter data-set was used by the authors in their work. The authors contrasted their proposed models to other models obtained from their literature review. The model evaluation metrics include Precision, Recall, Specificity, Accuracy, F-Measure and MCC. The proposed model achieved an accuracy of 96.1%, a recall score of 97.6%, precision score of 94% and a specificity score of 93.5%.

Ismaila Idris in his paper [15], proposed an ANN with a negative selection algorithm to classify spam and non-spam emails. Specifically, the author trained a genetic algorithm to separate the emails into self and non-self-units. These units are augmented into a vector with two elements which is used as the input for the Neural network. Essentially the author employed a genetic algorithm to identify key features from the emails, which are then used to train the Neural Network. The author contrasted the proposed model with an SVM classifier. The models were evaluated using train and test accuracy. The proposed model received a train accuracy score of 94.30% and a test accuracy score of 91.37%.

Sarit Chakraborty et al. in their paper [16] employed several variations of Decision tree classifier to filter spam emails from non-spam emails. The authors specifically used the NBTree Classifier, C 4.5 / J48 Decision Tree Algorithm and Logistic Model Tree Induction (LMT) classifier. These models were evaluated using accuracy. The authors show that LMT out performs all the other classifiers with an accuracy of over 85%, followed by NBTree with an accuracy of over 82% and lastly J48 with an accuracy of 78%.

Ola Amayri et al. in their paper [17] explored inductive and transductive SVM classifiers to identify spam emails. The authors have employed myriad kernels for both Inductive and Transductive SVM which include string-based kernels. The authors show that string-based kernels outperform all other kernels used in conjunction with SVM classifiers in classifying spam emails. The authors have also explored myriad feature extraction techniques for text-based data. Some notable distance-based kernels include Polynomial.TF, RBF-Gaussian.TF, RBF-Laplacian.TF etc all normalized with both L1 and L2 norm. Notable string based SVM kernels include SSK, Spectrum, mismatched etc. The authors have also explored several learning modes for SVM classifiers which include both online and Batch learning mode. The results show varying performances when kernel type, data-preprocessing techniques and feature extraction techniques are taken into account. The kernels used by the authors were predominantly

based on Term Frequency and the feature extraction algorithm was BOW (Bag of words). The data-sets used by the authors were spambase, 2006 ECML Discovery, trec05-p1 (Cormack and Lynam 2005) and trec06p.

Enaitz Ezpeleta et al. in their paper [18], analyse several Naive Bayes based spam filters. The authors propose using the average polarity score of emails as a feature for spam classification. The authors show that NB based classifiers trained with polarity score out perform traditional NB based spam filters. The NB classifiers used by the authors include – Bayesian Logistic Regression, Complement Naive Bayes, DMNBtext, Naive Bayes Multinomial Updateable, Naive Bayes, Naive Bayes Multinomial and Naive Bayes Updateable. The data-sets used by the authors include polarity data-set v2.0, Movie Reviews2 and CSDMC 2010 Spam corpus3. The results show that using polarity scoring of emails in conjunction with NB classifiers boost the accuracy of spam filters to 99.61%.

Tim Repke et al. in their paper [19], propose the use of RNN models to divide email threads into individual sections. These individual sections are further classified into 2 or 5 zones that encapsulate information relating to the individual email HEADER, BODY, GREETINGS and SIGNATURES. The proposed system is comprised 2 distinct sections where by the email is encoded using a CNN- Word Embedding model, the output of which is used as the input to the second section wherein an RNN in conjunction with CRF (Conditional Random Field) is Employed to zone the input email into multiple sections. The authors used the Enron data-set [20].

Yoon Kim in his paper [21] employ a CNN model along with a pre trained word vectors to classify sentences. The author used several benchmarks text-based classification data-sets to conduct necessary experiments. The author used a pre trained word2vector word representation model trained by Google. The CNN variations considered are CNN-rand, CNN-static, CNN-nonstatic and CNN-multichannel. The author concludes that simple CNN based architectures perform quite well in classification tasks related to Natural language using pre trained word2vector word representation models.

B. Observation

Based on the literature review conducted above several research gaps have been identified. Most research lack the use of appropriate performance metrics for model evaluation, as such it is difficult to conclude if these models are generalizing to the trained corpus or over-fitting. Although transfer learning has been widely used in Image classification tasks, its use in NLP is limited and, in most cases, unused. In NLP tasks data wrangling is an important aspect as it defines the structure of the features vectors but is missing in the existing literature. Lastly comparisons provided within the literature are performed under varying experimental conditions such as different studies perform different data pre-processing steps, employ different models using different feature extraction techniques, make use of different data-sets with different distributions. As such a statistically conclusive result cannot be inferred from such comparisons. Our study aims at providing solutions to such issues.

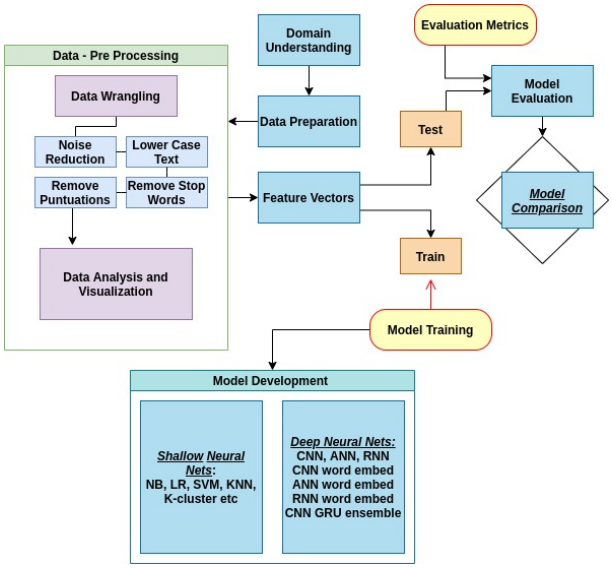

Fig. 1. Workflow diagram

III. METHODOLOGY

Supervised Classification tasks generally consist of six steps. These steps are defined as Data acquisition, Data pre-processing, Feature Extraction, Model Selection, Model Evaluation and Model Deployment. This study explores the application of NLP (Natural Language Processing) techniques to classify spam emails.
The figure 1 illustrates the steps performed for preprocessing data-set in our work.

A. Data Acquisition

The data set used in our work was acquired from the Enron data-set [20], a well-known publicly available benchmark corpus dedicated to spam email classification. Only the Kaminski folder of the data-set was used to generate the .csv file. Table I shows preliminary statistics. We divided the dataset into a training set (80 percent) and test set (20 percent). Resulting training set had 4396 email samples and the test set had 1099 email samples. Figure 2 illustrates the data imbalance issue.

B. Data Pre-processing

Data wrangling is defined as the process of augmenting raw data into optimised usable formats that enables optimal feature extraction and model training. Steps involving data wrangling are project specific which include the type of data being handled, the goal of the project etc.

Several data wrangling/ pre-processing steps were performed on raw emails towards optimising the dataset for the purpose of spam email classification. These steps include **Normalization** (removing repeating emails, stopwords, words less than 3 words and punctuations), **Tokenization** and **Lemmatization**.

TABLE I
STATISTICS ON THE DATA-SET

Number of emails in the data-set	5696.00
Number of spam emails in the data-set	1366.00
Mean length of emails	354.99
Standard deviation for length of emails	423.47
Minimum length of emails	3.00
25%, length at Lower Quarter of the Distribution	112.00
50%, median length	232.00
75%, length at Upper Quarter of the Distribution	438.0
Max length of sentence	6623.00

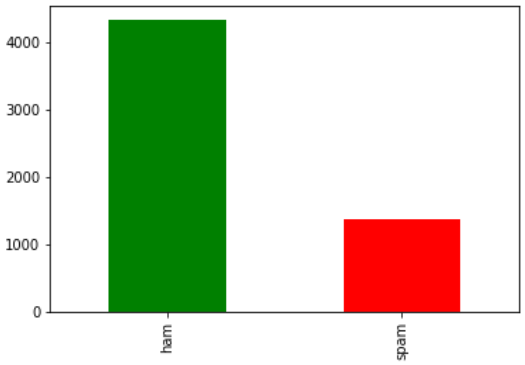

Fig. 2. Total number of ham and spam in the data-set

Stopwords are defined as words that don't add any semantic or contextual meaning to the overall message. These words are frequently used within emails or documents as such hold negligible usable information.As such these words were removed. Some examples of stop words include:
[your, she's, what, that, are, was, a, while]

For similar reasons punctuations and other linguistic symbols were also removed from the dataset. Stopwords and puntuations usually hold negligible value when it comes to classification of texts or documents but they may be very useful in predicting words, completing sentences and other similar tasks.

Usually raw texts contain empty spaces, new line characters or other document specific symbols. For the purpose of this task the raw emails were converted into a list of words where each word is referred to as a token. This process is defined as tokenization. The following example shows:
General Text = " Hello, How are you doing ? " Tokenized = [Hello, How, are, you, doing, ?]
Lastly these tokens were lemmatized. Generally words in a document are used in varying forms due to grammatical requirements. These tokens were converted back to sentences. For example :
Democracy - Democratic - Democratization

Lemmatization converts these words to their base, root or dictionary form. This allows optimal feature extraction as

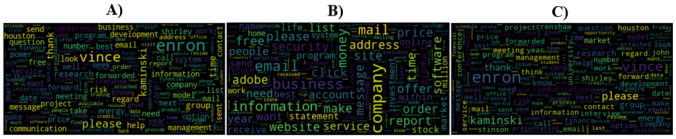

Fig. 3. A. Word Cloud on the entire database B. of only spam C. only ham emails

words used in varying contexts will have the same base form.

Three word cloud plots were generated from the augmented/pre-processed data set to encapsulate word occurrences based on the entire data set. The figure 3 plots illustrate the followings: Plot A shows the most common words used within the entire data set including both spam and ham emails. Plot B shows only the spam, on the other hand, C shows only the same ham emails.

C. Feature Extraction

Machines are unable to process natural language such as text in English. As such, Linguistic data is required to be transformed into a numeric representation which concisely encapsulates the statistical inferences (Distribution, Frequencies etc) of the data as well as contextual and semantic meaning in many cases. This numeric representation is used as features for training Machine learning models. **For the purpose of this study TF-IDF and Word Embedding has been used**.

*1) **TF-IDF (Term Frequency - Inverse Document Frequency)**:* TF-IDF was used to train traditional machine learning classifiers (Not neural network based) within our work. TF-IDF score is assigned to a word based on the frequency of the word in a document and the number of documents it exists in [22]. Generally, within linguistic data or corpus certain words are used more frequently despite retaining lower significance or relevance contrast to certain other words used rarely despite holding higher relevance to the meaning of the message. TF-IDF score is used to balance out the weights assigned to words such that frequent non relevant words hold lower values compared to infrequent highly relevant words. That is, the TF-IDF score gives more meaning to rare terms in the corpus and penalizes more commonly occurring terms.

TF-IDF has two parts namely Term Frequency (TF) and Inverse Document Frequency (IDF). TF can be defined as the probability of a certain word occurring in any document. IDF assigns more weight to rare words as frequency can lower the importance of certain words. TF-IDF is governed by the following equations:
Weight of Term, t in document/record/email

$$d = TF(t, d) * IDF(t) \tag{1}$$

Where, TF is the term frequency of term t in document d and IDF is the inverse document frequency of t.

TF(t, d) = Total number of times t occurs in d / Total number of words in d [23]

IDF(t) = log (Total number of documents/emails, N / Total number of d with t) [24]

TF-IDF scores of certain words in the first email are given in table II.

TABLE II
TF-IDF SCORES

university	texas	conference	energy	finance	february
0.188157	0.241525	0.243075	0.17721	0.200091	0.375109

*2) **Word Embedding (Keras Embedding and GLOVE Embedding using Transfer Learning)**:* Prior to training a word embedding texts, messages or emails i.e. linguistic data is required to be transformed into an indexed format. This is done by creating a unique set of words (vocabulary) from the training dataset and indexing them with real numbers. The words of the emails are then replaced with their index from the vocabulary. Keras tokenizer was used to create a vocabulary and convert the textual data/emails to numeric feature vectors (list of index). These feature vectors were padded to a max length of 2000 for keras embedding and 2400 for GLOVE embeddings. The following example illustrates:

0th training sample text:
naturally irresistible corporate...
0th training sample indexed:
[257316 192882 109423...0 0 0]

Such feature vectors are used to learn a word embedding jointly as part of a neural network. Word Embeddings are able to represent linguistics items or words in a low dimensional vector space. These numeric vectors that represent words are able to encapsulate word similarities and so are grouped together within the vector space/ word space e.g Boat - Ship.

Word embeddings are a type of word representation model that can retain semantics, contexts and meanings of words unlike TF-IDF scores which rely heavily on word frequencies. Word embeddings were heavily discussed within linguistics by Zellig Harris [25], John Firthand [26] and was known as Distributional semantics as it was based on Semantic theory of linguistics. Bengio et al. [27] was the first to learn a word Embedding as part of a Neural Language Model. Collobert [28] demonstrated the highly effective nature of pre trained word embeddings as part of a neural network when applied to downstream NLP tasks like text classification, parts of speech tagging etc. Mikolov [313, 319] was the first to develop a pretrained word embedding toolkit called **word2vec** which can be used to train vector space models for any NLP task. Pennington [29] developed GLOVE, which is an embedding based on word co-occurances.

There are primarily two ways to train word embeddings, namely: Learnable Embedding (Keras Embedding), Pre trained word embedding (GLOVE). The deep learning models (Neural Network based) within this work were trained using keras embedding and GLOVE embedding separately with varying architectures.

Keras Embedding: Keras provides an Embedding api that can be trained on any word corpus in conjunction with a neural network or standalone fashion. Keras Embedding requires a specific input and output dimensions prior to training. The input texts/words are required to be converted into a one hot encoded vector prior to training the Embedding matrix. The parameters of the Keras Embedding matrix is updated during Gradient Descent. During this work a 100-dimension word embedding was trained using Keras Embedding layer. As such each word from the corpus/emails is transformed to a 100-dimensional vector. t-Distributed Stochastic Neighbor Embedding (t-SNE) by [30] was used to transform the 100-dimensional embedding vectors to 2 dimensional for visualization. Figure 4 shows some word representations learned in conjunction with ANN, CNN and Bi-LSTM.

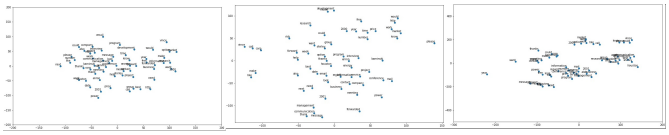

Fig. 4. A.ANN Embedding B.CNN Embedding C.RNN Embedding

GLOVE Embedding: GLOVE is a pre-trained word embedding model developed by [29]. GLOVE employs both global statistics of matrix factorization like LSA (Latent Semantic Analysis) and word2vec model. Essentially GLOVE uses word co-occurrence matrix derived using text statistics across the entire text corpus resulting in computing a better word embedding matrix. Pennington has published several GLOVE embedding matrices of varying dimensions (50,100,200,300). For this study we have employed the 100-dimensional GLOVE embedding matrix. The GLOVE Embedding layer was **static** i.e, GLOVE embedding matrix was **not fine-tuned (not updated during Gradient Descent)** towards email classification.

D. Model Selection

1) Machine Learning Classifiers: The traditional machine learning classifiers trained within this work include, *Multinomial Naive Bayes*, *Random Forest*, *Decision Tree*, *Gradient Boosting*, *XGBoost*, *Logistic Regression*,*K-nearest neighbors*, *SVM* and *SVM(RBF)*.

2) Deep Neural Network Classifiers:

a) ANN – Artificial Neural Network: An ANN is a multi-layer perceptron that uses backpropagation learning. Backpropagation involves using an objective function to fit the parameters (weights and biases) of the model, usually square error or maximum likelihood, using gradient optimization techniques. During backpropagation with ANN's, the error (the difference between the predicted outcome and the true outcome) is propagated back from the output to the parameters to adjust it in the direction of minimum error. An ANN comprises several interconnected layers of nodes: an input layer, with each input node corresponding to a feature, hidden layers and an output layer. The connections between the nodes have adjustable parameters that specify the extent to which the output of one node will be reflected in the activity of the adjacent layer nodes. These parameters, along with the connections among the nodes, determine the output of the ANN.

b) Bi-LSTM – Bi-Directional Long Short-Term Memory: RNN specializes in sequential or time-series data because of their ability to retain the memory of previous inputs, which affects the current state, meaning that the inputs and outputs are interdependent. Long Short-Term Memory, is a kind of RNN (Recurrent Neural Network). Due to its design characteristics, LSTM models are very suitable for modeling time-series data, like texts (emails). BiLSTM is a combination of forward LSTM and backward LSTM. It is often used to model contextual information in natural language processing tasks. BiLSTM's can better capture the two-way semantic dependence. For example, "This restaurant is dirty, not as good as the one next door." The "no" here is a modification of the degree of "dirty.

c) CNN – Convolutional Neural Network: CNN's employ convolution operations on the data matrix to reduce its dimensions while retaining important features of the dataset. CNN's are usually employed for image classification. However, it can also be utilized for text-based classification by taking in sequential data like text as a 1-dimensional matrix and, consequently, performing 1-dimensional convolution operation. Convolutional kernels of CNN's are able to detect patterns within textual data (Feature vector of text). These patterns are analogous to bi-grams or tri-grams in Neural Language Models. CNN models are basically Deep ANN's with Convolutional layers.

E. Model Evaluation

The classifiers and neural network models trained within this work are evaluated using the following metrics.
True Positives (TP): Total number of spam emails correctly recognized.
True Negatives (TN): Total number of benign/ham emails correctly recognized.
False Positives (FP): Total number of ham emails falsely recognized as spam emails.
False Negatives (FN): Total number of spam emails falsely recognized as ham emails.
Precision: Precision in this works context is defined as the ratio of predicted spam emails and true spam emails.

$$tp/(tp + fp) \tag{2}$$

High precision means the model predicts low false positives and high true positives.
Recall: Recall in this works context is defined as the ratio of true spam emails and predicted spam emails.

$$tp/(tp + fn) \tag{3}$$

The higher the Recall, the higher the model identifies the positive events and labels correctly. F1 score: F1 score is the weighted average of precision and recall

$$F1 = 2 * (Precision * Recall)/(Precision + Recall) \tag{4}$$

The best performing models have an F1 score close to 1.
Accuracy: Accuracy is the ratio of correctly classified emails (both spam and ham) among all emails in the test or train set.

$$(tp + tn)/(tp + fp + fn + tn) \tag{5}$$

AU-ROC: Receiver Operating Characteristics (ROC) is a probability distribution curve for both tp and tn. Area under curve (AUC) of ROC is the measure of separation that is the ability of a model to distinguish between classes correctly. A higher AU-ROC is indicative of a model's ability to classify spam emails as spam and ham emails as ham in this work. An AU-ROC score of 0.5 is suggestive of a model's inability to distinguish between spam and ham email. A score of 0 indicates that a model is misclassifying spam as ham and vice versa. AU-ROCs are able to provide actual model performance for highly imbalanced datasets.

IV. RESULT AND ANALYSIS

A. Experimental Setup:

The traditional machine learning models were trained on a laptop using a jupyter notebook environment. The configuration is given in table 3.

TABLE III
LAPTOP CONFIGURATION

System Specification	Configuration
Operating System	Ubuntu 20.04 LTS
RAM	8 gigabyte
System type	64 bit
Base processor	X11
Processor	Intel® Core™ i5-2450M CPU @ 2.50GHz × 4
python	3.8

The Deep learning models (ANN, CNN, Bi-LSTM) were trained using google colab GPU and high ram configuration. Libraries used within this work include: Pandas, Numpy, Seaborn, Matplotlib, WordCloud, Scikit-learn, Keras, nltk and Tensorflow

B. Deep Learning Model Setup:

From here on GLOVE based models will be referred to as ANN, CNN and Bi-LSTM GLOVE 1 and GLOVE 2 respectively, such as, ANN GLOVE 1, Bi-LSTM GLOVE 2 and keras embedding based models will be referred to as ANN, CNN and Bi-LSTM keras embedding respectively, such as, ANN keras embedding. The model setups are given in table 4.

C. Comparison of traditional machine learning classifiers:

Table 5 illustrates a comparison of all machine learning classifiers trained within this work. The table is sorted by Accuracy in the Descending order. The evaluation metrics are provided in percentages with the exception of training time which is in seconds. The table shows that XGBoost achieves the best scores for recall, f1 score, accuracy and AU-ROC. SVM (RBF) achieves the best precision score. Multinomial

TABLE IV
DEEP MODEL SETUP

Embedding	**Keras Embedding (Learnable)**			**GLOVE Embedding (static) Architecture 1**			**GLOVE Embedding (static) Architecture 2**		
Models	**ANN**	**RNN**	**CNN**	**ANN**	**CNN**	**RNN**	**ANN**	**CNN**	**RNN**
Learning Rate	0.001								
Optimizer	Adam								
Loss	Binary Cross Entropy								
Gradient Descent	Batch gradient Descent								
Model Structure	Input, 4396,2000 Embedding, 2000,100 Global max pool 1D Dense, 10 units Drop out, 0.1 Dense, 1 unit	Input, 4396,2000 Embedding, 2000,100 Drop out, 0.1 Bi-LSTM, 100 units Dense, 64 units Drop out, 0.2 Dense, 1 unit	Input, 4396,2000 Embedding, 2000,100 Conv1D, 20 filters kernel 3 Global max pool 1D Dense, 16 units Dense, 1 unit	Input, 4396,2000 Embedding, 2400,100 Global max pool 1D Dense, 20 units Drop out, 0.5 Dense, 10 units Drop out, 0.2 Dense, 1 unit	Input, 4396,2000 Embedding, 2400,100 Bi-LSTM, 128 units Drop out, 0.5 Bi-LSTM, 128 units Drop out, 0.5 Dense, 5 units Dense, 1 unit	Input, 4396,2000 Embedding, 2400,100 Conv1D, 20 filters kernel 3 Global max pool 1D Dense, 20 units Drop out, 0.5 Dense, 10 units Drop out, 0.2 Dense, 1 unit	Input, 4396,2000 Embedding, 2400,100 Global max pool 1D Drop out, 0.5, Flatten Dense, 1024 units Dense, 64 units Drop out, 0.5 Dense, 32 units Drop out, 0.2 Dense, 16 units Drop out, 0.2 Dense, 1 unit	Input, 4396,2000 Embedding, 2400,100 Bi-LSTM, 128 units Bi-LSTM, 128 units Drop out, 0.2 Dense, 32 units Dense, 16 units Dense, 1 unit	Input, 4396,2000 Embedding, 2000,100 Conv1D, 100 filters kernel 5 Conv1D, 100 filters kernel 5 Global max pool 1D Dense, 64 units Drop out, 0.2 Dense, 32 units Drop out, 0.2 Dense, 16 units Drop out, 0.2 Dense, 8 units Drop out, 0.2 Dense, 1 unit
Number of Epochs	4	2	4	100	4	100	25	4	9
Activities	Hidden Layers : Relu , Output Layer : Sigmoid								
Parameters (Trainable)	2817721	2990429	2823073	2241	630027	6661	171649	637505	109417

naive bayes require the least amount of training time while achieving moderately high evaluation scores. SVM (Linear) achieves the second-best evaluation scores and training time. KNN and Decision tree achieve the lowest evaluation scores. Gradient Boosting requires the most training time (24.85 seconds). All the classifiers have received evaluation scores of over 95 percentile which was expected and an improvement over [4], [6], [7], [16], [19].

[16] shows that word embedding with CNN-LSTM achieves an accuracy of 95.9 %, recall of 1.0, precision of 0.936, f1 score of 0.967 and a G-mean of 96.7 %. This paper also shows FastText email representation in conjunction with CNN-LSTM achieves the same evaluation scores as word embedding with the exception of precision which is 93.5%. [6] shows that Text CNN achieves an accuracy of 97.54 % and f1 score of 0.97. [7] shows a Bert based model (Best performing model) with accuracy of 0.9730 and f1 score of 0.9696 on the training set and that of 0.9867 and 0.9866 respectively on the holdout set. [16] shows a Logistic model tree classifier with an accuracy of 85.9%. [19] shows that their proposed model QUAGGA produces a precision, recall and accuracy score of 0.98 respectively.

The top 3 classifiers (XGBoost, SVM-Linear, SVM-RBF) within our work outperform all the models from the [4], [6], [7], [16], [19].

Heatmap of classification report and AU-ROC curve for ML-Classifiers: Figure 5 and Figure 6 show heatmaps of the classification report of all classifiers trained within this work as well as corresponding AU-ROC curves. KNN has the highest false negatives followed by random forest, decision tree and naive bayes. The other classifiers have below 1 percent false negatives with the exception of Gradient Boosting and Logistic regression. XGBoost has the lowest false negatives. All classifiers except the decision tree achieve below 1 percent false positives. Naive bayes predicted 0 false positives (lowest) followed by random forest and logistic regression at 1 respectively. Decision tree has the highest false positives (14 fp which is 1.27 % of the test set) among all the classifiers.

TABLE V
COMPARISON OF THE ML CLASSIFIERS

Model	**Precision**	**Recall**	**f1 score**	**Accuracy**	**AU-ROC**	**Train Time(s)**
XGBoost	0.9844	**0.9898**	**0.9871**	**0.9900**	**0.9898**	18.89
SVM (Linear)	**0.9868**	0.9801	0.9833	0.9873	0.9801	0.1
SVM (RBF)	0.9879	0.9789	0.9833	0.9873	0.9789	15.33
Logistic Regr.	0.9861	0.9644	0.9746	0.9809	0.9644	2.16
Gradient Boosting	0.9787	0.9620	0.9699	0.9773	0.9620	24.85
Random Forest	0.9816	0.9458	0.9620	0.9718	0.9458	7.06
Decision Tree	0.9602	0.9419	0.9506	0.9627	0.9419	2.92
KNN	0.9625	0.9350	0.9477	0.9609	0.9350	0.31
MultinomialNB	0.9352	0.7885	0.8312	0.8899	0.7885	0.02

XGBoost achieves the highest AU-ROC score followed by SVM (Linear) and SVM (RBF). KNN achieves the lowest AU-ROC score among all classifiers.

D. Deep Neural Network models

Table 6 illustrates a comparison of all Deep learning models (ANN, CNN, BI-LSTM) with keras embedding and pretrained static GLOVE embedding trained within this work. The table shows that keras Embedding based DNN models outperform static GLOVE embedding based models by a very small margin in terms of evaluation scores.

However, the GLOVE based DNN models were static in nature i.e the GLOVE embedding used in conjunction with the DNN models were not updated or fine-tuned during Gradient Descent (training). As such these models required a significantly lower number of trainable parameters compared to keras embedding based models. Despite being static in nature, GLOVE based models achieved very high evaluation scores. CNN GLOVE 1 achieved a test accuracy of 98.36%,

Fig. 5. Heatmap of classification report for ML-Classifiers:

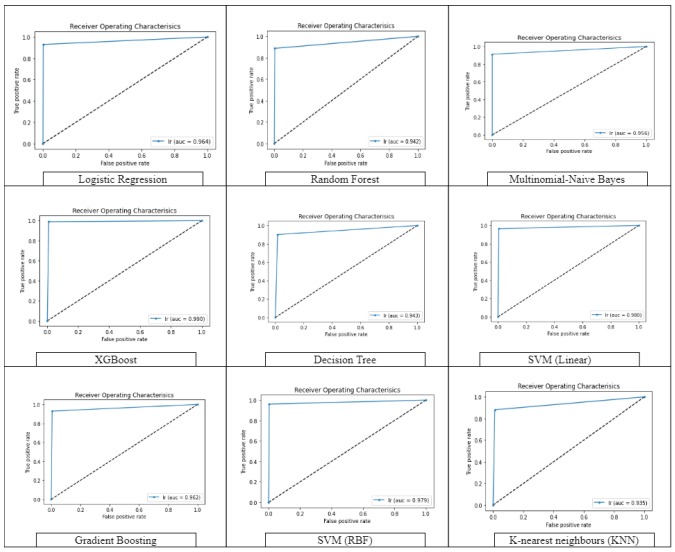

Fig. 6. AU-ROC curve for ML-Classifiers:

a recall score of 0.9788, a precision score of 0.9799, f1 score of 0.9777 and AU-ROC score of 0.9799. Bi-LSTM GLOVE 1 achieved a test accuracy of 98.18%, AU-ROC score of 0.9775, f1 score of 0.9754, precision score of 0.9775 and recall score of 0.9764.

This shows that transfer learning in downstream NLP tasks like text classification, parts of speech tagging etc can be extremely efficient in terms of both resources and time. As these pre-trained word embeddings (word2vec, GLOVE etc) encapsulate word similarities off the shelf, they can be used statically in conjunction with neural networks or classifiers and achieve extremely high evaluation scores. However, these pre-trained GLOVE based models can also be fine-tuned to a specific task which may yield better evaluation scores compared to keras embedding.

TABLE VI
OVERALL RESULTS FOR DNN

Model	Precision	Recall	f1 score	Train Accuracy	Test Accuracy	Error	Loss	AU-ROC	Train Time(s)
ANN(word embedding)	**0.9888**	**0.9894**	0.9899	0.9991	**0.9918**	0.82	0.03	**0.9888**	48.53
CNN(word embedding)	0.9865	0.9893	**0.9922**	1.0000	**0.9918**	0.82	0.04	0.9865	54.47
RNN(word embedding)	0.9789	0.9833	0.9879	0.9982	0.9873	1.27	0.05	0.9789	119.11
CNN-1(GLOVE embedding)	0.9802	0.9732	0.9668	1.0000	0.9791	2.09	1.42	0.9802	941.79
ANN-1(GLOVE embedding)	0.7289	0.7694	0.9090	0.8280	0.8571	14.29	0.27	0.7289	862.44
RNN-1(GLOVE embedding)	0.5000	0.4252	0.3699	0.7543	0.7398	26.02	0.58	0.5000	9504.23
CNN-2(GLOVE embedding)	0.9708	0.9661	0.9616	0.9934	0.9736	2.64	0.09	0.9708	154.95
RNN-2(GLOVE embedding)	0.9752	0.9621	0.9508	0.9734	0.9700	3.00	0.07	0.9752	4091.46
ANN-2(GLOVE embedding)	0.6572	0.6850	0.8870	0.8055	0.8198	18.02	0.36	0.6572	298.98

Overall, the DNN models trained in this work outperform all other models proposed within the literature review section specifically [4], [6], [7], [16], [19].

Heatmap of classification report and AU-ROC curves for all DNN models: Figure 7 shows heatmaps of classification reports for all the DNN models (both keras and GLOVE embedding based). ANN GLOVE 1 incurs the highest false positives (88) and 20 false negatives while ANN GLOVE 2 incurs the highest false negatives (165) and 5 false positives. CNN keras embedding has the lowest false positives followed by RNN keras embedding and ANN keras embedding. ANN keras embedding has the lowest false negatives followed by CNN keras embedding and RNN keras embedding.

Among DNN models ANN GLOVE 1 and ANN GLOVE 2 have the lowest evaluation metrics which was expected due to the complexity of the problem, structure of the data and the general working principle of artificial neural networks.

Figure 8 shows the AU-ROC curves for all the DNN models. ANN GLOVE 1 and ANN GLOVE 2 incurred the lowest AU-ROC scores which was unsurprising as their precision and recall was low as well as high false negatives and false positives. All other DNN models have an AU-ROC score of over 0.95 which means these models have generalized well to the imbalanced dataset and are able to distinguish spam and ham emails with moderately high accuracy.

V. CONCLUSION

The primary objective of this paper was three fold. We have provided a concise comparison of traditional machine learning algorithms (Naive Bayes, SVM etc) for email classification using Enron corpus. XGBoost achieved the highest evaluation scores among all other classifiers. We have trained 6 DNN models using pre-trained GLOVE embedding (static) and 3 DNN models using keras embedding. Then we have provided a thorough comparison of these 9 DNN models which yielded some interesting results. DNN models with keras embedding due to their large number of trainable parameters (Table 4) have outperformed all other GLOVE embedding based

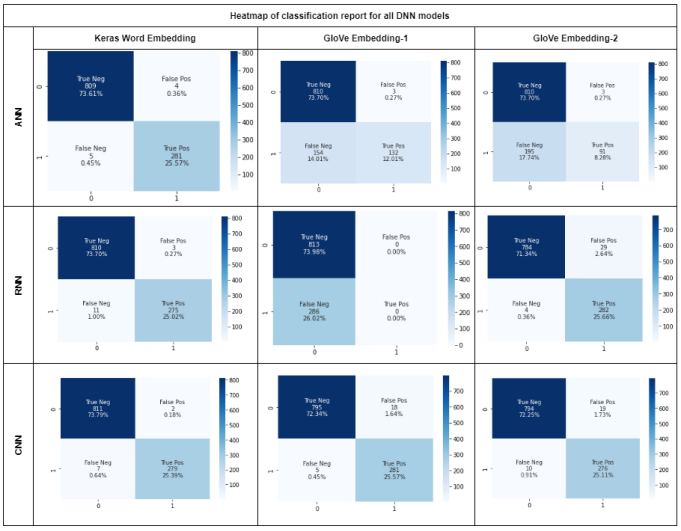

Fig. 7. Heatmap of classification report for all DNN models

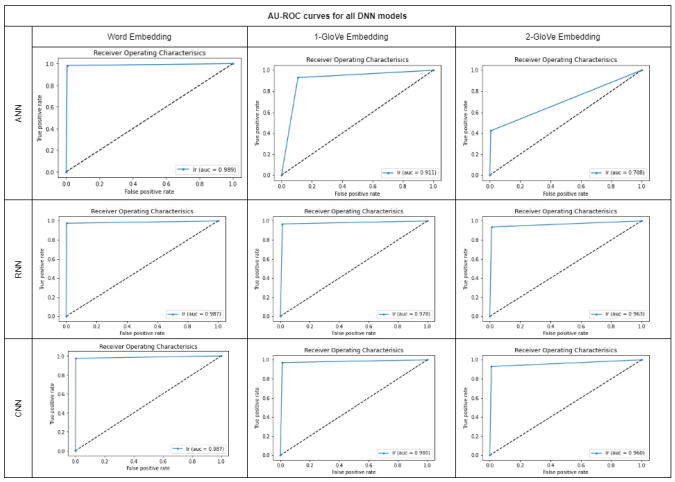

Fig. 8. AU-ROC for DNN-Classifiers

models and classifiers. We have also observed that static (non-trainable) GLOVE embedding based DNN models have achieved extremely high evaluation scores despite having extremely low trainable parameters contrast to keras embedding based models. Specifically, CNN GLOVE 1, CNN GLOVE 2, Bi-LSTM GLOVE 1 and Bi-LSTM GLOVE 2 models performed extremely efficiently which can be attributed to the CNN and Bi-LSTM layers of the models. This shows that transfer learning can be extremely useful and, in many cases, better for downstream NLP tasks like text classification etc.

All models and classifiers trained within this work yield better evaluation scores compared to existing work. This could be due to the data pre-processing techniques applied to the dataset prior to feature extraction.

Some future works include:

1) Using all Enron directory to generate a balanced data-set. The enron directory contains 0.5 million messages or emails approximately.
2) Adversarial Attacks to evaluate the robustness of the trained models.
3) Google Bert embedding layer and Bert models.
4) Explore transformer models with attention layers with the new data-set.
5) Explore Explainable AI models.
6) Fine tune the GLOVE embedding models.

REFERENCES

[1] "Email statistics report." [Online]. Available: https://www.radicati.com/wp/wp-content/uploads/2021/EmailStatisticsReport,2021-2025ExecutiveSummary.pdf

[2] J. Johnson, "Spam statistics: Spam e-mail traffic share 2019," Jul 2021. [Online]. Available: https://www.statista.com/statistics/420391/spam-email-traffic-share/

[3] P. by Statista Research Department and O. 21, "Global average daily spam volume 2021," Oct 2021. [Online]. Available: https://www.statista.com/statistics/1270424/daily-spam-volume-global/

[4] S. Srinivasan, V. Ravi, M. Alazab, S. Ketha, A.-Z. Ala'M, and S. K. Padannayil, "Spam emails detection based on distributed word embedding with deep learning," in *Machine Intelligence and Big Data Analytics for Cybersecurity Applications*. Springer, 2021, pp. 161–189.

[5] S. Nazirova, "Survey on spam filtering techniques," Aug 2011. [Online]. Available: https://www.scirp.org/journal/paperinformation.aspx?paperid=6769

[6] S. Seth and S. Biswas, "Multimodal spam classification using deep learning techniques," in *2017 13th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*. IEEE, 2017, pp. 346–349.

[7] Q. Yaseen *et al.*, "Spam email detection using deep learning techniques," *Procedia Computer Science*, vol. 184, pp. 853–858, 2021.

[8] S. Srinivasan, V. Ravi, V. Sowmya, M. Krichen, D. B. Noureddine, S. Anivilla, and K. Soman, "Deep convolutional neural network based image spam classification," in *2020 6th Conference on data science and machine learning applications (CDMA)*. IEEE, 2020, pp. 112–117.

[9] A. Chavda, K. Potika, F. D. Troia, and M. Stamp, "Support vector machines for image spam analysis," in *Proceedings of the 15th International Joint Conference on e-Business and Telecommunications - Volume 2: BASS,*, INSTICC. SciTePress, 2018, pp. 431–441.

[10] S. Isik, Z. Kurt, Y. Anagun, and K. Ozkan, "Spam e-mail classification recurrent neural networks for spam e-mail classification on an agglutinative language," *International Journal of Intelligent Systems and Applications in Engineering*, vol. 8, no. 4, pp. 221–227, 2020.

[11] L. Özgür, T. Güngör, and F. S. Gürgen, "Adaptive anti-spam filtering for agglutinative languages: a special case for turkish," *Pattern Recognit. Lett.*, vol. 25, pp. 1819–1831, 2004.

[12] A. N. Soni, "Spam-e-mail-detection-using-advanced-deep-convolution-neuralnetwork-algorithms," *JOURNAL FOR INNOVATIVE DEVELOPMENT IN PHARMACEUTICAL AND TECHNICAL SCIENCE*, vol. 2, no. 5, pp. 74–80, 2019.

[13] A. Barushka and P. Hajek, "Review spam detection using word embeddings and deep neural networks," in *IFIP International Conference on Artificial Intelligence Applications and Innovations*. Springer, 2019, pp. 340–350.

[14] F. Wei and U. T. Nguyen, "Twitter bot detection using bidirectional long short-term memory neural networks and word embeddings," in *2019 First IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*. IEEE, 2019, pp. 101–109.

[15] I. Idris, "E-mail spam classification with artificial neural network and negative selection algorithm," *International Journal of Computer Science & Communication Networks*, vol. 1, no. 3, pp. 227–231, 2011.

[16] S. Chakraborty and B. Mondal, "Spam mail filtering technique using different decision tree classifiers through data mining approach-a comparative performance analysis," *International Journal of Computer Applications*, vol. 47, no. 16, 2012.

[17] O. Amayri and N. Bouguila, "A study of spam filtering using support vector machines," *Artificial Intelligence Review*, vol. 34, no. 1, pp. 73–108, 2010.

[18] E. Ezpeleta, U. Zurutuza, and J. M. G. Hidalgo, "Does sentiment analysis help in bayesian spam filtering?" in *International Conference on Hybrid Artificial Intelligence Systems*. Springer, 2016, pp. 79–90.

[19] T. Repke and R. Krestel, "Bringing back structure to free text email conversations with recurrent neural networks," in *European Conference on Information Retrieval*. Springer, 2018, pp. 114–126.

[20] B. Klimt and Y. Yang, "The enron corpus: A new dataset for email classification research," in *European Conference on Machine Learning*. Springer, 2004, pp. 217–226.

[21] Y. Kim, "Convolutional neural networks for sentence classification," *CoRR*, vol. abs/1408.5882, 2014. [Online]. Available: http://arxiv.org/abs/1408.5882

[22] S. Robertson, "Understanding inverse document frequency: On theoretical arguments for idf," *Journal of Documentation - J DOC*, vol. 60, pp. 503–520, 10 2004.

[23] H. P. Luhn, "A statistical approach to mechanized encoding and searching of literary information," *IBM Journal of Research and Development*, vol. 1, no. 4, pp. 309–317, 1957.

[24] K. Spark Jones, "A statistical interpretation of term importance in automatic indexing," *Journal of Documentation*, vol. 28, no. 1, pp. 11–21, 1972.

[25] Z. S. Harris, "Distributional structure," *¡i¿WORD¡/i¿*, vol. 10, no. 2-3, pp. 146–162, 1954. [Online]. Available: https://doi.org/10.1080/00437956.1954.11659520

[26] J. Firth, "A synopsis of linguistic theory 1930-1955," in *Studies in Linguistic Analysis*. Philological Society, Oxford, 1957, reprinted in Palmer, F. (ed. 1968) Selected Papers of J. R. Firth, Longman, Harlow.

[27] Y. Bengio, R. Ducharme, and P. Vincent, "A neural probabilistic language model," in *Journal of Machine Learning Research*, vol. 3, 01 2000, pp. 932–938.

[28] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 160–167.

[29] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[30] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.