Machine Learning Project Report

SUBMITTED TO: **Dr. Arpan Goyal**

SUBMITTED BY: Group A5

Ayush Choudhary(2023btech020)

Divyansh Thakur(2023btech026)

Himanshu Gurjar(2023btech036)

Email Spam Detection using Hybrid Machine Learning and NN-based Deep Learning Approaches

1. Introduction

Email remains one of the most commonly used communication tools, with billions of messages exchanged daily across business, personal, and academic domains. However, the pervasive growth of unsolicited messages, commonly referred to as spam, poses a serious challenge. These messages often promote scams, malicious links, or irrelevant advertisements and significantly reduce productivity. Effective spam filtering is not just a matter of convenience but also essential for preventing fraud and securing communication systems. Traditional rule-based filtering methods, although initially effective, have become outdated with the rise of dynamically generated and cleverly crafted spam content. The introduction of machine learning (ML) and deep learning (DL) techniques has significantly enhanced the performance of spam detection systems. This paper presents a comparative study between a hybrid ML ensemble model and a deep learning model using Convolutional Neural Networks (CNN), showing the practical benefits and trade-offs of each. Applications of spam detection systems include integration into email clients, spam firewalls, anti-phishing systems, and content moderation tools across social media and communication platforms.

2. Problem Statement

The task addressed in this research is binary classification: determining whether an email is spam or ham (non-spam).

Challenges in this task include:

Data Imbalance: In most real-world datasets, non-spam messages far outweigh spam, causing classifiers to be biased.

- Linguistic Diversity: Spam messages use informal, obfuscated, or cleverly disguised text.
- Evolving Threats: Spammers frequently change patterns to evade detection.

To address these, our work proposes a hybrid machine learning ensemble combining Naive Bayes, Support Vector Machine, and Random Forest with a hard voting mechanism. Alongside, a NN-based deep learning model is constructed to exploit spatial and sequential information from tokenized messages. The novelty lies in combining comparative strategies and fine-tuning for both ML and DL architectures on the same dataset, giving insight into their efficiency and adaptability.

3. Literature Review

Numerous studies have explored spam detection using supervised learning techniques. Early work utilized Naive Bayes and Decision Trees on bag-of-words models. Androutsopoulos et al. introduced the Ling-Spam corpus and evaluated email classification using multiple Bayesian techniques. More recently, methods like SVM and Random Forest have gained popularity for their robustness. Deep learning advancements brought methods like LSTM (Long Short-Term Memory networks) and transformer-based models like BERT, which excel in contextual understanding.

Common datasets used include:
- SMS Spam Collection Dataset (used in our work)

Studies report varying results, with accuracy ranging from 93% to 98%, depending on preprocessing, feature extraction, and models used. However, many ignore ensemble strategies and focus either on ML or DL alone. This paper fills that gap by applying both approaches in a structured experiment.

Related Work on Spam Detection [4]

Several studies have explored Machine Learning (ML) and Deep Learning (DL) techniques for spam email detection:

- Srinivasan et al. [10] – Word embedding improves deep learning-based spam detection.
- Soni [11] – Proposed THEMIS (RCNN-based) model achieving 99.84% accuracy, outperforming LSTM and CNN.

- Hassanpur et al. [12] – Used word2vec & Neural Networks, achieving 96% accuracy.
- Egozi et al. [13] – Applied NLP techniques (word count, stopwords, punctuation) with SVM, detecting 80% phishing & 95% ham emails.
- Seth et al. [14] – Hybrid CNN model analyzing text & images, achieving 98.87% accuracy.
- Ezpeleta et al. [15] – Improved Bayesian classifier with sentiment analysis, boosting accuracy to 99.21%.
- Bibi et al. [16] – Comparative study on spam filtering models based on accuracy and datasets.

Result Tables [4]

Author	Classifiers Used	Best Accuracy	Dataset
Awad et al. [17]	NB, NN, SVM, KNN	99% (Naïve Bayes)	SpamAssassin (6000 instances)
Saab et al. [18]	SVM, NB, LMSVM, Decision Tree, ANN	93% (SVM)	Spambase (4597 instances)
Shajideen et al. [19]	SVM, NB, J48	94% (SVM)	3762 spam, 5172 ham emails

4. Proposed Methodology

We propose two main models:

Hybrid Machine Learning Model:
- Text Cleaning: Lowercasing, punctuation and digit removal, whitespace trimming.

- TF-IDF Vectorization: Extracted unigrams and bigrams, excluded English stopwords.
- Model Tuning: GridSearchCV was applied for optimal parameters:
  - Naive Bayes (alpha)
  - SVM (C parameter)
  - Random Forest (n_estimators, max_depth)
- Ensembling: Hard VotingClassifier aggregated predictions from the three base models.

NN Deep Learning Model:
- Label Encoding: Mapped 'spam' to 1 and 'ham' to 0.
- Tokenization and Padding: Converted text to sequences, padded to the longest length.
- Architecture:
  - Embedding Layer (128 dimensions)
  - Conv1D (128 filters, kernel size 5)
  - GlobalMaxPooling
  - Dense layers with dropout (64 and 32 units)
  - Sigmoid output layer

Training Setup:
- 30 epochs, batch size 64
- Binary cross-entropy loss with Adam optimizer
- Class weights for imbalance handling

5. Experiments

Dataset: SMS Spam Collection from UCI Repository. It includes ~5,500 text messages, each labeled as spam or ham.

Data Visualization:
- Distribution: Countplot showed a skewed ratio (Ham > Spam)
- WordClouds: Visualized high-frequency words in both classes.

ML Model Experiments:
- Conducted GridSearchCV (5-fold CV) on each classifier.
- Features: TF-IDF with bigram range and English stopword removal.
- Evaluation on hold-out test set.

CNN Model Experiments:
- Used TensorFlow and Keras libraries.
- Vocabulary size auto-generated using Tokenizer.
- Used padded sequences for fixed input.
- Applied class weighting due to label imbalance.
- Trained on Colab GPU instance.

6. Results and Discussion

Hybrid Model:
- Accuracy: 97.66%
- Precision: 1.00
- Recall (Spam): 0.83
- F1-score: 0.91
- Confusion Matrix: Few spam emails misclassified as ham
- Advantage: Fast inference, interpretable models

Overall **Accuracy:** 0.9766816143497757

Classification Report:

	precision	recall	f1-score	support
0	0.97	1.00	0.99	965
1	1.00	0.83	0.91	150
accuracy			0.98	1115
macro avg	0.99	0.91	0.95	1115
weighted avg	0.98	0.98	0.98	1115

Graphs and Visuals:

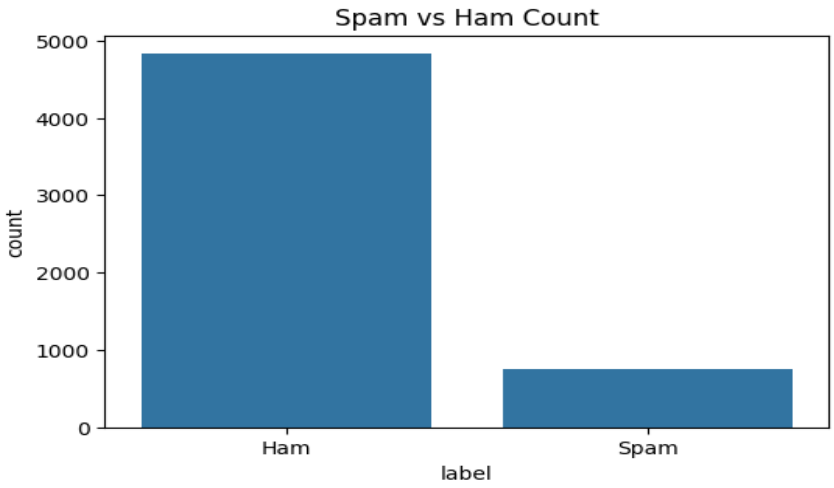

Fig1:Count Plot

Fig2: World Cloud

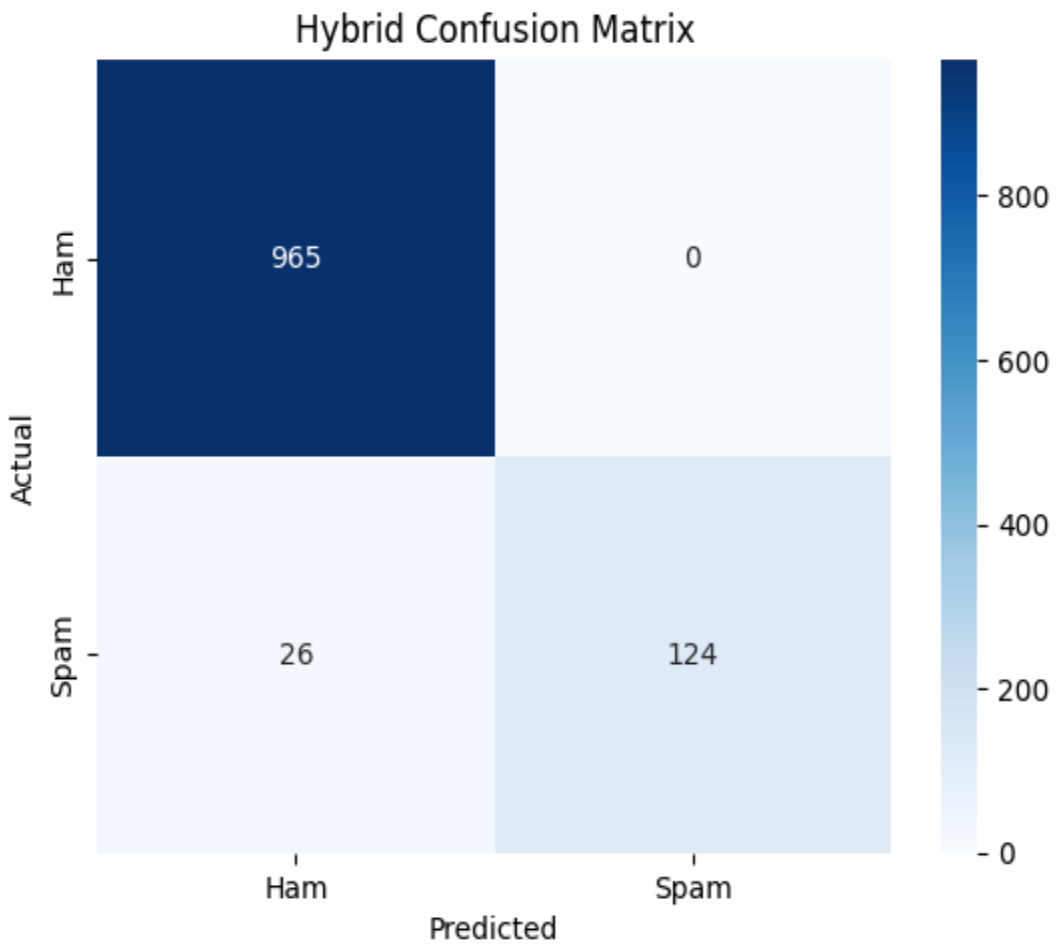

Fig3: Heatmaps of confusion matrix

CNN Model:
- Accuracy: 98.83%
- Validation loss: Low with minimal overfitting
- Advantage: Higher contextual understanding

Sample Predictions: Spam: "You've won a free ticket!" → Correctly flagged

The NN model slightly outperforms the hybrid model in terms of generalization and spam recall. However, the hybrid model is simpler and requires less computational power.

7. Conclusion and Future Scope

In conclusion, both the hybrid and CNN models provide high accuracy in spam detection. The CNN model, with its deep structure, provides a slight edge in spam recall and accuracy. The hybrid model offers speed and simplicity. Together, these results affirm the benefit of multi-approach strategies in NLP tasks.

Future Scope:

- Explore transformer-based models like BERT or RoBERTa
- Use multi-label classification (spam, phishing, scam)
- Deploy model into real-time email systems via REST APIs
- Enhance dataset diversity with multilingual and modern corpora

8. References

1. UCI SMS Spam Dataset: https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset

2. Scikit-learn documentation: https://scikit-learn.org/ Keras API reference: https://keras.io/

3. WordCloud library: https://amueller.github.io/word_cloud/

4. Spam Email Detection Using Deep Learning Techniques – ScienceDirect Research Paper

5. (PDF) Email Spam Detection Using Machine Learning Algorithms Research Paper

6. (PDF) Email based Spam Detection Research Paper

9. Appendix

Code for The Hybrid Model

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import string
import re
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import LinearSVC
from sklearn.ensemble import RandomForestClassifier, VotingClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.pipeline import Pipeline
import joblib

df = pd.read_csv('data.csv')[['v1', 'v2']]
df.columns = ['label', 'message']
df.dropna(inplace=True)
df['label'] = df['label'].map({'spam': 1, 'ham': 0})

def clean_text(text):
    text = text.lower()
    text = re.sub(r'\d+', '', text)
    text = text.translate(str.maketrans('', '', string.punctuation))
    text = text.strip()
    return text

df['cleaned'] = df['message'].apply(clean_text)


plt.figure(figsize=(6, 4))
sns.countplot(x='label', data=df)
plt.title("Spam vs Ham Count")
plt.xticks([0, 1], ['Ham', 'Spam'])
plt.show()

from wordcloud import WordCloud
```

```
spam_words = ' '.join(list(df[df['label'] == 1]['cleaned']))
ham_words = ' '.join(list(df[df['label'] == 0]['cleaned']))

plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.imshow(WordCloud(width=400, height=300).generate(spam_words))
plt.axis('off')
plt.title('Spam WordCloud')

plt.subplot(1, 2, 2)
plt.imshow(WordCloud(width=400, height=300).generate(ham_words))
plt.axis('off')
plt.title('Ham WordCloud')
plt.show()

vectorizer = TfidfVectorizer(stop_words='english', ngram_range=(1, 2))
X = vectorizer.fit_transform(df['cleaned'])
y = df['label']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
nb_params = {'nb__alpha': [0.1, 0.5, 1.0]}
nb_grid = GridSearchCV(nb_pipeline, nb_params, cv=5, scoring='accuracy')
nb_grid.fit(df['cleaned'], y)
print("Best Naive Bayes Parameters:", nb_grid.best_params_)

svm_pipeline = Pipeline([
    ('tfidf', TfidfVectorizer(stop_words='english', ngram_range=(1, 2))),
    ('svm', LinearSVC(max_iter=10000))
])
svm_params = {'svm__C': [0.1, 1, 10]}
svm_grid = GridSearchCV(svm_pipeline, svm_params, cv=5, scoring='accuracy')
svm_grid.fit(df['cleaned'], y)
print("Best SVM Parameters:", svm_grid.best_params_)

rf_pipeline = Pipeline([
    ('tfidf', TfidfVectorizer(stop_words='english', ngram_range=(1, 2))),
    ('rf', RandomForestClassifier())
])
rf_params = {'rf__n_estimators': [50, 100, 200], 'rf__max_depth': [10, 20, None]}
rf_grid = GridSearchCV(rf_pipeline, rf_params, cv=5, scoring='accuracy')
rf_grid.fit(df['cleaned'], y)
print("Best Random Forest Parameters:", rf_grid.best_params_)


nb_model = nb_grid.best_estimator_.named_steps['nb']
svm_model = svm_grid.best_estimator_.named_steps['svm']
rf_model = rf_grid.best_estimator_.named_steps['rf']

hybrid = VotingClassifier(estimators=[
    ('nb', nb_model),
    ('svm', svm_model),
    ('rf', rf_model)
], voting='hard')

hybrid.fit(X_train, y_train)
y_pred = hybrid.predict(X_test)

print("\n Accuracy:", accuracy_score(y_test, y_pred))
print("\n Classification Report:\n", classification_report(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Ham', 'Spam'], yticklabels=['Ham', 'Spam'])
plt.title("Hybrid Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

joblib.dump(hybrid, 'models/hybrid_model.pkl')
joblib.dump(vectorizer, 'models/tfidf_vectorizer.pkl')
```

Code For CNN

```
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, GlobalAveragePooling1D, Dense, Dropout, Conv1D, GlobalMaxPooling1D
import numpy as np

df = pd.read_csv("data.csv", usecols=[0, 1], names=["v1", "v2"], skiprows=1)


le = LabelEncoder()
df['label'] = le.fit_transform(df['v1'])


tokenizer = Tokenizer(oov_token="<OOV>")
tokenizer.fit_on_texts(df['v2'])
sequences = tokenizer.texts_to_sequences(df['v2'])


maxlen = max([len(x) for x in sequences])

padded = pad_sequences(sequences, padding='post', maxlen=maxlen)
X_train, X_test, y_train, y_test = train_test_split(
    padded, df['label'], test_size=0.2, random_state=42)
neg, pos = np.bincount(df['label'])
total = neg + pos
class_weight = {0: (1 / neg) * (total / 2.0), 1: (1 / pos) * (total / 2.0)}
vocab_size = len(tokenizer.word_index) + 1
model = Sequential([
    Embedding(input_dim=vocab_size, output_dim=128, input_length=maxlen),
    Conv1D(128, 5, activation='relu'),
    GlobalMaxPooling1D(),
    Dense(64, activation='relu'),
    Dropout(0.5),
    Dense(32, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

history = model.fit(X_train, y_train, epochs=30, validation_data=(X_test, y_test), batch_size=64, class_weight=class_weight)

loss, accuracy = model.evaluate(X_test, y_test)
print(f"Test Accuracy: {accuracy:.4f}")

def predict_message(msg):
    seq = tokenizer.texts_to_sequences([msg])
    pad = pad_sequences(seq, maxlen=maxlen, padding='post')
    pred = model.predict(pad)
    return "Spam" if pred[0][0] > 0.5 else "Ham"

print(predict_message("Congratulations! You've won a free ticket!"))
print(predict_message("Hey, can we meet tomorrow?"))
```