






## Article

# Enhancing Fake News Detection with Word Embedding: A Machine Learning and Deep Learning Approach

Mutaz A. B. Al-Tarawneh <sup>1,\*</sup> , Omar Al-irr <sup>1</sup> , Khaled S. Al-Maaitah <sup>2</sup> , Hassan Kanj <sup>1</sup>   
and Wael Hosny Fouad Aly <sup>1</sup> 

<sup>1</sup> College of Engineering and Technology, American University of the Middle East, Egaila 54200, Kuwait; omar.alirr@aum.edu.kw (O.A.-i.); hassan.kanj@aum.edu.kw (H.K.); wael.aly@aum.edu.kw (W.H.F.A.)

<sup>2</sup> Computer Engineering Department, Mutah University, Karak 61710, Jordan; khaled\_almaaitah@mutah.edu.jo

\* Correspondence: mutaz.al-tarawneh@aum.edu.kw

**Abstract:** The widespread dissemination of fake news on social media has necessitated the development of more sophisticated detection methods to maintain information integrity. This research systematically investigates the effectiveness of different word embedding techniques—TF-IDF, Word2Vec, and FastText—when applied to a variety of machine learning (ML) and deep learning (DL) models for fake news detection. Leveraging the TruthSeeker dataset, which includes a diverse set of labeled news articles and social media posts spanning over a decade, we evaluated the performance of classifiers such as Support Vector Machines (SVMs), Multilayer Perceptrons (MLPs), and Convolutional Neural Networks (CNNs). Our analysis demonstrates that SVMs using TF-IDF embeddings and CNNs employing TF-IDF embeddings achieve the highest overall performance in terms of accuracy, precision, recall, and F1 score. These results suggest that TF-IDF, with its capacity to highlight discriminative features in text, enhances the performance of models like SVMs, which are adept at handling sparse data representations. Additionally, CNNs benefit from TF-IDF by effectively capturing localized features and patterns within the textual data. In contrast, while Word2Vec and FastText embeddings capture semantic and syntactic nuances, they introduce complexities that may not always benefit traditional ML models like MLPs or SVMs, which could explain their relatively lower performance in some cases. This study emphasizes the importance of selecting appropriate embedding techniques based on the model architecture to maximize fake news detection performance. Future research should consider integrating contextual embeddings and exploring hybrid model architectures to further enhance detection capabilities. These findings contribute to the ongoing development of advanced computational tools for combating misinformation.

**Keywords:** natural language processing; machine learning; deep learning; fake news detection; word embedding



**Citation:** Al-Tarawneh, M.A.B.; Al-irr, O.; Al-Maaitah, K.S.; Kanj, H.; Aly, W.H.F. Enhancing Fake News Detection with Word Embedding: A Machine Learning and Deep Learning Approach. *Computers* **2024**, *13*, 239. <https://doi.org/10.3390/computers13090239>

Academic Editor: Mariofanna Milanova

Received: 25 July 2024

Revised: 30 August 2024

Accepted: 1 September 2024

Published: 19 September 2024



**Copyright:** © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

The digital age has led to an unprecedented surge in the availability and dissemination of information. While this has democratized access to knowledge, it has also accelerated the spread of misinformation and fake news. Fake news, defined as false or misleading information presented as factual news, has become a pervasive issue, influencing public opinion, swaying political outcomes, and even inciting social unrest. The challenge of distinguishing authentic news from fake news is compounded by the vast volume and rapid propagation of information across social media and other online platforms [1].

The impact of fake news extends beyond digital platforms, influencing real-world events and eroding trust in traditional media sources. For instance, during the 2016 U.S. presidential election, a surge in fake news stories arguably influenced voter behavior and election outcomes [2]. Similarly, the 2019 general election in India saw misinformation campaigns on social media attempting to sway voters by spreading false narratives [3].

The COVID-19 pandemic also witnessed a significant spread of false information regarding the virus, treatments, and vaccines, posing serious public health risks [4]. These examples underscore the urgent need for effective mechanisms to detect and combat fake news.

Natural language processing (NLP) has emerged as a crucial tool in combating fake news, with word embedding techniques playing a pivotal role. Word embeddings represent words as vectors in a continuous vector space, capturing semantic relationships such that words with similar meanings have similar vector representations. Techniques like Term Frequency-Inverse Document Frequency (TF-IDF), Word2Vec, and FastText have revolutionized text processing by providing dense vector representations that enhance the ability of algorithms to understand and analyze textual data. These embeddings are critical for improving feature extraction, allowing models to detect nuanced patterns and relationships in data [5].

This study advances the field by systematically evaluating the impact of multiple word embedding techniques on the performance of a broad range of machine learning (ML) and deep learning (DL) models for fake news detection. Unlike previous studies that have typically focused on a single embedding technique or a limited model range, our work comprehensively analyzes three word embedding techniques—TF-IDF, Word2Vec, and FastText—and their effect on various ML and DL models. This analysis is conducted using the TruthSeeker dataset [6], which encompasses a diverse array of news articles, tweets, and social media posts collected for over more than a decade.

The research investigates individual classifiers, including Logistic Regression, Naive Bayes, K-Nearest Neighbors (KNNs), Support Vector Machines (SVMs), multilayer perceptrons (MLPs), Decision Trees, and Random Forests, as well as Convolutional Neural Networks (CNNs) for deep learning. By leveraging these embeddings, this study enhances the feature extraction process, enabling models to capture nuanced patterns in the data.

The main contributions of this work are as follows:

- **Comprehensive Evaluation:** This study evaluates the effectiveness of various machine learning and deep learning models for fake news detection using a comprehensive set of performance metrics, including accuracy, precision, recall, and F1 score.
- **Novel Analysis of Word Embedding Impact:** Unlike previous studies, this work systematically analyzes the impact of different word embedding techniques (TF-IDF, Word2Vec, and FastText) on a wide range of machine learning and deep learning models.
- **Utilization of a Rich Dataset:** The TruthSeeker dataset, comprising diverse and temporally extensive data, is utilized to ensure robust training and testing of models.
- **Broad Comparative Analysis:** The performance of individual classifiers and CNN architectures is compared to identify optimal strategies for fake news detection.

The remainder of this paper is organized as follows: Section 2 summarizes related research efforts, and Section 3 presents the utilized research methodology, including data collection and preprocessing, word embedding techniques, machine learning and deep learning implementation, and performance evaluation. Section 4 presents the obtained evaluation results. Section 5 highlights and discusses the main observations drawn from the evaluation results. Finally, Section 6 concludes and summarizes this work.

## 2. Related Work

In recent years, the detection of fake news on social media platforms has garnered significant attention from researchers. Various approaches have been proposed to tackle this problem, utilizing different machine learning and natural language processing (NLP) techniques. The primary focus of these studies has been on improving the accuracy and robustness of fake news detection systems through innovative methodologies and comprehensive datasets.

The study in [7] employed stochastic neural networks to extract sentiment from Twitter data, integrating natural language processing (NLP) techniques with neural networks (NNs). This approach demonstrated the effectiveness of combining sentiment analysis with

deep learning for predicting the trends and sentiments in social media posts. However, while this method effectively captures sentiment-based features, it lacks the ability to handle nuanced linguistic cues that are crucial for distinguishing fake news from genuine news. This limitation is addressed in our approach by incorporating multiple word embeddings, such as TF-IDF, Word2Vec, and FastText, which provide a more nuanced representation of textual data.

Similarly, the work in [8] developed a content classification and sentiment analysis model using COVID-Twitter-BERT, achieving high classification accuracy across multiple themes. It has highlighted the potential of transformer-based models in capturing the nuances of social media discussions during the COVID-19 pandemic. While transformer-based models like COVID-Twitter-BERT are effective in capturing context and nuances, they often require large amounts of labeled data and computational resources. Our approach addresses this limitation by leveraging less resource-intensive models, such as Logistic Regression and Naive Bayes, alongside deep learning models to achieve high accuracy with reduced computational cost.

Additionally, the framework introduced in [9] utilized Twitter data to generate automated reports on the Russia–Ukraine conflict. This framework leveraged advanced NLP algorithms, including language translation, sentiment analysis, and topic modeling, to provide comprehensive insights into the cyber activities during the conflict. While this method provides a broad overview of misinformation through multi-faceted NLP techniques, it lacks specificity in fake news detection. In contrast, our approach focuses explicitly on fake news detection using a refined combination of text representation techniques, which enhances detection specificity and accuracy.

Moreover, the research in [10] explored sentiment classification using various machine learning algorithms to determine the polarity of textual content. It was demonstrated that the Passive Aggressive (PA) algorithm, combined with a unigram model, outperformed other algorithms in sentiment analysis tasks across multiple datasets. Despite its effectiveness in sentiment classification, this approach does not adequately address the complexity of fake news, which often involves the subtle manipulation of information. Our proposed model overcomes this by employing a combination of word embeddings and machine learning models that can capture more sophisticated patterns indicative of fake news.

Furthermore, the application of NLP methods to anomaly detection in log files was investigated in [11], using algorithms such as AdaBoost, Gaussian Naive Bayes, and random forest. The study emphasized the importance of selecting appropriate classifiers and feature extraction techniques for effective anomaly detection. While effective in anomaly detection, these methods are not specifically tailored to the linguistic subtleties of fake news. Our approach incorporates more tailored NLP techniques and embeddings that better capture the deceptive language used in fake news.

The investigation in [12] focused on detecting fake news sources and content on online social networks. It reviewed various text feature extraction techniques and classification algorithms, finding that the Support Vector Machine (SVM) linear classifier using TF-IDF features achieved the highest accuracy in fake news detection. However, SVM models with TF-IDF features may not capture deeper semantic relationships in the text. Our approach uses a combination of embeddings (TF-IDF, Word2Vec, and FastText) and evaluates their impact across a variety of machine learning and deep learning models, providing a more comprehensive detection framework.

The research in [13] focused on the classification of fake news by examining different textual properties that distinguish fake from real news. Natural language processing techniques were employed to preprocess the data, subsequently training various machine learning classifiers with all possible combinations of the extracted properties. The results indicated that the Naive Bayes algorithm achieved the best performance, demonstrating the effectiveness of thorough preprocessing and feature combination in enhancing model accuracy. While thorough preprocessing and feature combination improved model performance, this approach does not explore the use of more complex embeddings or deep

learning models. Our study extends this by systematically analyzing multiple embedding techniques and their influence on different models, providing insights into optimal strategies for fake news detection.

Graphical methods have also been employed in fake news detection. The study in [14] presented the Factual News Graph (FANG), a learning framework and graphical social context representation aimed at identifying false news. Unlike traditional contextual models, FANG emphasizes representation learning, proving to be efficient at inference time and scalable during training. However, graphical models like FANG primarily focus on social context and may not fully capture textual subtleties. Our model leverages various text embeddings to enhance textual analysis, complementing graphical approaches.

The work in [15] utilized a graph-based scan technique to identify rumors at the network level, focusing on local anomalies. The rumor detection system detects irregularities in user behavior, post content, and hashtag usage, offering a robust method for identifying rumors based on social media data streams. While graph-based approaches are effective for network-level analysis, they may not be sufficient for content-based fake news detection. Our approach enhances content analysis using a blend of embeddings and models, which is crucial for identifying fake news in text.

The research in [16] employed the XGBoost model to determine the significance of various variables in detecting fake news, identifying key factors. Representative machine learning classification models such as SVM, RF, LR, CART, and NNET were then utilized to compare their performance in fake news detection. Although effective, XGBoost and similar models may not fully leverage the textual data's depth and semantic richness. Our approach addresses this by incorporating deep learning models and word embeddings that provide a more nuanced understanding of text, enhancing the detection accuracy.

Recent advancements in fake news detection have also explored the role of Large Language Models (LLMs) and multimodal approaches. The study in [17] reveals that while LLMs such as GPT-3.5 can provide valuable multi-perspective rationales, they often underperform compared to fine-tuned Small Language Models (SLMs) like BERT. This suggests a hybrid approach leveraging both LLMs and SLMs to enhance detection capabilities. Additionally, the study in [18] focuses on the robustness of detection models against style-based attacks enabled by LLMs. The proposed SheepDog detector is designed to be resilient to stylistic mimicry, prioritizing content over style to maintain detection accuracy even in the face of sophisticated adversarial examples.

Moreover, the TELLER framework proposed in [19] introduces an explainable and generalizable approach to fake news detection, incorporating logic-based reasoning and cross-domain adaptability. TELLER's integration of logical reasoning improves model interpretability, making it more robust across different datasets without extensive retraining. This aligns with the increasing need for detection systems that not only perform well but are also adaptable and transparent in diverse real-world applications.

The inclusion of multimodal approaches, such as the framework proposed in [20], further broadens the scope of fake news detection. By integrating text, images, and videos, this model utilizes logic reasoning to provide a comprehensive analysis of misinformation, enhancing detection accuracy in a multimedia-rich environment.

Existing methods in fake news detection, while offering valuable insights, often lack a comprehensive approach that integrates multiple text representation techniques and evaluates their performance across a broad range of models. Our proposed approach aims to fill this gap by systematically analyzing the impact of various word embeddings—TF-IDF, Word2Vec, and FastText—on a diverse set of machine learning and deep learning models. This allows us to identify the most effective strategies for fake news detection, addressing the limitations observed in prior research. Our method not only enhances detection accuracy and robustness but also offers a more scalable and generalizable solution for fake news detection across different contexts.

In summary, our research builds on the existing literature by integrating advanced word embedding techniques with a comprehensive evaluation of both machine learning

and deep learning models, providing a more robust framework for fake news detection. This approach represents a significant step forward in improving the accuracy and efficiency of fake news detection systems.

### 3. Materials and Methods

This section highlights the stages followed in this research, including dataset description, dataset preprocessing, word embedding techniques, and machine learning and deep learning implementation for fake news detection along with performance evaluation. The main steps conducted to evaluate the performance of various machine learning and deep learning techniques are illustrated in Algorithms 1 and 2, respectively.

Algorithms 1 and 2 were designed to improve the detection of fake news by addressing specific challenges and incorporating innovative techniques. Algorithm 1 enhances fake news detection through advanced word embedding techniques such as TF-IDF and Word2Vec, which capture semantic relationships and contextual nuances in text, crucial for distinguishing between genuine and misleading information. This approach improves the precision of fake news detection by reducing false positives and negatives, making it effective in scenarios where subtle linguistic cues are important. Additionally, Algorithm 1 provides flexibility to adapt to new patterns in fake news content, addressing the challenge of varied linguistic expressions and evolving misinformation tactics. Algorithm 2 is designed to optimize model performance by leveraging deep learning techniques, particularly Convolutional Neural Networks (CNNs), which are capable of capturing complex patterns in textual data. This approach allows the model to effectively identify features that are not linearly separable and to learn from intricate patterns that traditional machine learning models might miss. The innovation in these algorithms lies in their ability to enhance fake news detection: Algorithm 1 excels at leveraging semantic and contextual information, while Algorithm 2 is effective in recognizing complex patterns and relationships in the data. These contributions demonstrate the potential for using advanced embedding techniques and deep learning models to improve the accuracy and robustness of fake news detection systems. In conclusion, Algorithms 1 and 2 address key challenges in fake news detection by incorporating sophisticated embeddings and deep learning architectures, providing significant improvements in model performance. Future research could further explore these approaches to enhance the adaptability and efficacy of fake news detection methods.

---

#### Algorithm 1 Tweet Pre-processing and Machine Learning Model Evaluation

---

```

1: Input: Raw tweet text  $T$ , labels  $y$ 
2: Output: Evaluation results  $R$ 
3: Step 1: Text Cleaning
4: for each tweet  $t \in T$  do
5:   Remove URLs, special characters, numerical values, emoticons, emojis, mentions,
   and hashtags
6:   Remove non-alphanumeric characters
7:   Remove extra spaces and trim the text
8:   Convert text to lowercase
9: end for
10: Step 2: Tokenization
11: for each cleaned tweet  $t \in T$  do
12:   Split  $t$  into individual words (tokens)
13: end for
14: Step 3: Stop Word Removal
15: for each token  $w \in T$  do
16:   if  $w$  is a stop word then
17:     Remove  $w$  from  $T$ 
18:   end if
19: end for

```

---

**Algorithm 1** *Cont.*


---

```

20: Step 4: Train-Test Split
21: Split the dataset into training set  $T_{train}$ ,  $y_{train}$  and testing set  $T_{test}$ ,  $y_{test}$ 
22: Step 5: TF-IDF Vectorization
23: Vectorize  $T_{train}$  and  $T_{test}$  using TF-IDF
24: Step 6: Word2Vec Embedding
25: Train Word2Vec model on  $T_{train}$ 
26: Vectorize  $T_{train}$  and  $T_{test}$  using Word2Vec
27: Step 7: FastText Embedding
28: Train FastText model on  $T_{train}$ 
29: Vectorize  $T_{train}$  and  $T_{test}$  using FastText
30: Step 8: Handle NaNs in Embeddings
31: for each vector  $v \in T_{train}, T_{test}$  do
32:   if any value in  $v$  is NaN then
33:     Replace  $v$  with a zero vector
34:   end if
35: end for
36: Step 9: Define Models
37: Define the list of models: Logistic Regression, Naive Bayes, K-Nearest Neighbors,
  Support Vector Machine, Multilayer Perceptron, Decision Tree, Random Forest
38: Step 10: Evaluate and Store Results
39: for each embedding technique  $e \in \{TF - IDF, Word2Vec, FastText\}$  do
40:   for each model  $m$  in models do
41:     Train  $m$  on  $T_{train}$  with embedding  $e$ 
42:     Predict  $y_{pred}$  on  $T_{test}$  using  $m$ 
43:     Calculate accuracy, precision, recall, and F1 score
44:     Store results in  $R$ 
45:   end for
46: end for
47: Return Evaluation results  $R$ 

```

---

**Algorithm 2** Tweet Pre-processing and CNN Model Evaluation

---

```

1: Input: Raw tweet text  $T$ , labels  $y$ 
2: Output: Evaluation results  $R$ 
3: Step 1: Text Cleaning
4: for each tweet  $t \in T$  do
5:   Remove URLs, special characters, numerical values, emoticons, emojis, mentions,
   and hashtags
6:   Remove non-alphanumeric characters
7:   Remove extra spaces and trim the text
8:   Convert text to lowercase
9: end for
10: Step 2: Tokenization
11: for each cleaned tweet  $t \in T$  do
12:   Split  $t$  into individual words (tokens)
13: end for
14: Step 3: Stop Word Removal
15: for each token  $w \in T$  do
16:   if  $w$  is a stop word then
17:     Remove  $w$  from  $T$ 
18:   end if
19: end for

```

---



**Algorithm 2** *Cont.*


---

```

20: Step 4: Train-Test Split
21: Split the dataset into training set  $T_{train}$ ,  $y_{train}$  and testing set  $T_{test}$ ,  $y_{test}$ 
22: Step 5: TF-IDF Vectorization
23: Vectorize  $T_{train}$  and  $T_{test}$  using TF-IDF
24: Step 6: Word2Vec Embedding
25: Train Word2Vec model on  $T_{train}$ 
26: Vectorize  $T_{train}$  and  $T_{test}$  using Word2Vec
27: Step 7: FastText Embedding
28: Train FastText model on  $T_{train}$ 
29: Vectorize  $T_{train}$  and  $T_{test}$  using FastText
30: Step 8: Handle NaNs in Embeddings
31: for each vector  $v \in T_{train}, T_{test}$  do
32:   if any value in  $v$  is NaN then
33:     Replace  $v$  with a zero vector
34:   end if
35: end for
36: Step 9: Define CNN Model Architectures
37: for each CNN model  $m \in \{model\_1, model\_2, model\_3\}$  do
38:   Define the architecture of  $m$ 
39: end for
40: Step 10: Reshape Data for CNN Models
41: Reshape  $T_{train}$  and  $T_{test}$  to fit CNN input requirements
42: Step 11: Train and Evaluate Models
43: for each embedding technique  $e \in \{TF - IDF, Word2Vec, FastText\}$  do
44:   for each CNN model  $m \in \{model\_1, model\_2, model\_3\}$  do
45:     Train  $m$  on  $T_{train}$  with embedding  $e$ 
46:     Predict  $y_{pred}$  on  $T_{test}$  using  $m$ 
47:     Calculate accuracy, precision, recall, and F1 score
48:     Store results in  $R$ 
49:   end for
50: end for
51: Return Evaluation results  $R$ 

```

---

**3.1. Dataset Description**

This research utilizes the TruthSeeker dataset, a groundbreaking resource designed to tackle the challenges of misinformation and fake news on social media. Covering a period from 2009 to 2022, the dataset documents the evolution of online discourse, capturing major global events and the associated rise in misinformation. The TruthSeeker dataset includes over 180,000 tweets, labeled through a meticulous multi-phase process to ensure high accuracy and reliability [6].

Initially, labels were assigned using reputable sources like Politifact and a panel of expert reviewers. This was followed by crowd-sourcing on Amazon Mechanical Turk (MTurk), where each tweet received annotations from three distinct workers. To enhance label robustness, an active learning verification process was employed, involving 456 unique MTurk Master Turkers. This process provided multi-dimensional validation, supporting both binary (real/fake) and multiclass (five-label and three-label) classification schemes. Such rigorous labeling methodologies ensure the dataset supports both basic and nuanced analyses of tweet authenticity.

Beyond simple labels, the TruthSeeker dataset offers extensive auxiliary information that enhances its analytical potential. Each tweet is accompanied by scores reflecting bot likelihood, user credibility, and user influence, providing a comprehensive view of the tweet's origin and potential impact. These scores are essential for understanding the dynamics of automated misinformation, content source credibility, and the mechanisms through which influential users propagate false information.

Additionally, the dataset is enriched with detailed metadata, including user profiles, tweet timestamps, and various engagement metrics such as likes, retweets, and replies. This contextual information is invaluable for studying the spread and virality of misinformation, enabling researchers to trace the lifecycle of fake news from inception to dissemination across networks.

Publicly accessible through the Canadian Institute for Cybersecurity (CIC), the TruthSeeker dataset is an open resource for the global research community. Its extensive validation and rich metadata make it a crucial tool for advancing fake news detection research. Researchers can use this dataset to develop and test new machine learning algorithms, study behavioral patterns associated with misinformation, and propose interventions to mitigate its spread.

In summary, the TruthSeeker dataset is a comprehensive, meticulously validated, and richly annotated resource that significantly contributes to understanding and combating misinformation on social media. Its detailed and multi-dimensional nature provides a robust foundation for a wide range of research endeavors aimed at addressing one of the most pressing issues of the digital age.

### 3.2. Dataset Pre-Processing

The pre-processing of tweets in this work involves several crucial steps to ensure that the text data are clean and suitable for analysis. These steps are described as follows:

- **Text Cleaning:** The initial step in the pre-processing pipeline is text cleaning. This involves removing any URLs, special characters, numerical values, emoticons, emojis, mentions, and hashtags from the raw tweet text. The purpose of this step is to eliminate non-alphanumeric characters and other elements that do not contribute to the semantic content of the tweets. Additionally, extra spaces are removed, and the text is trimmed to ensure uniformity. Finally, the entire text is converted to lowercase to standardize the input and treat words with different letter cases as the same word.
- **Tokenization:** Following text cleaning, the next step is tokenization. Tokenization involves splitting the cleaned text into individual words, also known as tokens. This step is essential for breaking down the text into smaller, processable units that can be analyzed independently. Each token represents a single word, which allows for more granular manipulation and analysis of the text data.
- **Stop Word Removal:** The final step in the pre-processing pipeline is stop word removal. Stop words are common words such as “the”, “and”, “in”, etc., that appear frequently in the text but do not carry significant meaning or contribute much to the context. Removing these stop words helps to reduce noise and focus on the more informative parts of the text. A predefined list of stop words is used for this purpose, and any token matching a stop word is removed from the text.

These pre-processing steps ensure that the tweet text is clean, tokenized, and stripped of irrelevant components, making it suitable for further analysis and model training.

### 3.3. Word Embedding Techniques

In the context of text classification and natural language processing, word embedding plays a crucial role in transforming textual data into numerical vectors that machine and deep learning models can interpret. This section discusses three widely used word embedding techniques: Term Frequency-Inverse Document Frequency (TF-IDF), Word2Vec, and FastText, which are used in this work.

#### 3.3.1. Term Frequency-Inverse Document Frequency (TF-IDF)

The Term Frequency-Inverse Document Frequency (TF-IDF) is a statistical measure used to evaluate the importance of a word in a document relative to a collection of documents, or corpus [21]. It combines two metrics: Term Frequency (TF), which measures the frequency of a word in a document, and Inverse Document Frequency (IDF), which



measures how unique or rare a word is across the corpus. Mathematically, TF-IDF is defined as follows:

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \times \text{IDF}(t, D) \quad (1)$$

where

$$\text{TF}(t, d) = \frac{f(t, d)}{\sum_{t' \in d} f(t', d)} \quad (2)$$

$$\text{IDF}(t, D) = \log \frac{|D|}{|\{d \in D : t \in d\}|} \quad (3)$$

Here,  $f(t, d)$  represents the frequency of term  $t$  in document  $d$ ,  $|D|$  is the total number of documents, and  $|\{d \in D : t \in d\}|$  is the number of documents containing term  $t$ . TF-IDF assigns higher scores to terms that are frequent in a document but rare across the corpus, thus highlighting terms that are more relevant to the specific document.

### 3.3.2. Word2Vec

Word2Vec is a neural network-based embedding technique that learns vector representations of words by training on a large corpus of text [22]. It generates dense word vectors that capture semantic relationships between words. Word2Vec employs two primary models: Continuous Bag of Words (CBOW) and Skip-gram. The CBOW model predicts a target word based on its context words, while the Skip-gram model predicts context words given a target word. The objective functions for these models are as follows:

$$J_{\text{CBOW}} = -\log p(w_t | w_{t-m}, \dots, w_{t-1}, w_{t+1}, \dots, w_{t+m}) \quad (4)$$

$$J_{\text{Skip-gram}} = -\log p(w_{t-m}, \dots, w_{t-1}, w_{t+1}, \dots, w_{t+m} | w_t) \quad (5)$$

where  $w_t$  represents the target word, and  $w_{t-m}, \dots, w_{t-1}, w_{t+1}, \dots, w_{t+m}$  represent the context words. The resulting word vectors are dense and capture semantic similarities, such that similar words have similar vectors.

### 3.3.3. FastText

FastText, an extension of Word2Vec developed by Facebook's AI Research (FAIR) lab, improves word embedding by considering subword information [23]. Instead of learning word vectors directly, FastText represents each word as bag of character n-grams. This approach allows FastText to generate embeddings for out-of-vocabulary words and capture morphological variations in words. The loss function used in FastText is similar to that of Word2Vec but incorporates subword information:

$$J_{\text{FastText}} = - \sum_{w_i \in \text{context}} \log \sigma(v_{w_i}^T v_{w_t}) \quad (6)$$

where  $v_{w_i}$  represents the embedding vector of context words and  $v_{w_t}$  represents the embedding vector of the target word. By leveraging subword information, FastText achieves improved performance on tasks involving rare and morphologically rich languages.

These embedding techniques were selected because they offer a balance between simplicity, interpretability, and performance, which is essential for fake news detection tasks. TF-IDF is a statistical measure that identifies important words in a document relative to the entire corpus, making it highly effective for traditional machine learning models. Word2Vec and FastText are neural network-based embeddings that capture semantic relationships between words, enabling both machine learning and deep learning models to better understand context and subtle nuances in language. These characteristics make them particularly suitable for detecting fake news, where understanding context and semantic relationships is crucial.

### 3.4. Machine Learning Techniques

In the realm of machine learning, various algorithms offer distinct approaches to classification and prediction tasks. Among these, Naive Bayes (NB) leverages probabilistic methods to classify data based on Bayes' theorem, while Logistic Regression (LR) provides a robust linear model for binary outcomes. K-Nearest Neighbors (KNN) utilizes distance metrics to classify instances based on their proximity to labeled data. Support Vector Machines (SVMs) excel in finding optimal hyperplanes that separate different classes. Multi-Layer Perceptron (MLP), a type of neural network, captures complex patterns through multiple layers of interconnected neurons. Decision Trees (DTs) model decisions as a series of hierarchical choices, and Random Forest (RF) aggregates multiple Decision Trees to improve classification accuracy and robustness. Each of these techniques brings unique strengths to the task of data analysis and classification. This section briefly describes each of these techniques, which are utilized in this work for fake news detection.

- *Naive Bayes (NB)*: Naive Bayes classifiers, based on Bayes' theorem, are extensively used for text classification tasks. This probabilistic algorithm calculates the likelihood of a class label based on feature probabilities. The assumption of feature independence given the class label simplifies computation and allows for efficient training and prediction. Despite their practical effectiveness in tasks like spam filtering and sentiment analysis, Naive Bayes classifiers can be sensitive to feature correlations and non-Gaussian class distributions [24,25].
- *Logistic Regression (LR)*: Logistic Regression is a widely used statistical method for binary classification problems. It models the probability of a binary outcome based on one or more predictor variables by applying a logistic function to a linear combination of the input features. The output probabilities are used to assign class labels, typically using a threshold of 0.5. Logistic Regression is prized for its simplicity, interpretability, and efficiency, making it suitable for large datasets. It provides insights into the relationships between features and the target variable through coefficients that indicate the strength and direction of these relationships. However, Logistic Regression assumes linear relationships between the features and the log-odds of the target variable, and might not capture more complex patterns in the data [26].
- *K-Nearest Neighbors (KNNs)*: KNN is a non-parametric algorithm that classifies by determining the majority class among the  $K$  nearest neighbors of a data point. Distance metrics like Euclidean or Manhattan distance are used to identify these neighbors, with the majority vote determining the class label. While KNN is simple to implement and effective for multi-class problems, it can become computationally expensive with large datasets and is sensitive to the choice of  $K$  and distance metric. Additionally, KNN is affected by the curse of dimensionality, where performance degrades as the number of features increases [27].
- *Support Vector Machines (SVMs)*: SVMs are a versatile classification tool applicable to both binary and one-class problems. They aim to identify the optimal hyperplane that separates different classes while maximizing the margin between them. The choice of kernel functions, such as linear, polynomial, or radial basis function (RBF), enables the modeling of both linear and non-linear relationships. The regularization parameter  $C$  adjusts the balance between margin width and classification error, influencing the model's complexity and performance [28].
- *Multilayer Perceptron (MLP)*: MLP is a feedforward neural network consisting of multiple layers of neurons. MLPs learn complex, non-linear relationships through the forward propagation of inputs and backpropagation for error correction. Activation functions like ReLU or sigmoid introduce non-linearity, enabling the network to model intricate patterns. MLPs are effective for high-dimensional data but can be computationally demanding and prone to overfitting if not properly regularized [29,30].
- *Decision Trees (DTs)*: Decision Trees are a well-known machine learning algorithm used for classification tasks. They construct a model in a tree structure, with internal nodes

representing feature tests, branches indicating test outcomes, and leaf nodes denoting class labels. The model is developed through recursive partitioning of the feature space based on the attribute providing the highest information gain. Decision Trees are appreciated for their interpretability and visualization, offering insights into the decision-making process. They can handle both numerical and categorical features and are robust against outliers. However, they are prone to overfitting, especially when the tree is too deep or the dataset is small [31].

- *Random Forest (RF)*: Random Forest is an ensemble learning method that aggregates multiple Decision Trees to improve classification accuracy. Each tree is trained on a random subset of features and data, with predictions aggregated through majority voting. This approach reduces overfitting by combining diverse trees and provides feature importance scores, aiding in understanding feature contributions to predictions. Random Forest is effective for high-dimensional data, missing values, and outliers, though it can be computationally intensive [32,33].

### 3.5. Deep Learning and Convolutional Neural Networks (CNNs)

Deep learning represents a significant advancement in machine learning, characterized by the use of neural networks with multiple layers to model complex patterns in data. Unlike traditional machine learning methods that rely on manually engineered features, deep learning approaches are capable of automatically learning hierarchical feature representations from raw data. This process involves a deep architecture composed of several layers of interconnected neurons, where each layer progressively extracts more abstract features from the input data [34].

Convolutional Neural Networks (CNNs) are a specialized type of deep learning architecture designed to process data with grid-like structures, such as images and text. CNNs utilize convolutional operations to automatically detect and learn local patterns within the input data. This is achieved through the application of convolutional filters, or kernels, which slide over the input to extract relevant features. In the case of text classification, CNNs process sequences of word embeddings, which are dense vector representations of words capturing semantic meaning. Embeddings such as those produced by Word2Vec or TF-IDF transform words into continuous vectors, allowing CNNs to effectively handle textual data [35].

Within a CNN architecture, convolutional layers apply multiple filters to the word embeddings to identify local patterns and features in the text. These filters detect various n-grams or word combinations that are indicative of specific contexts or meanings. After the convolutional layers, pooling layers are employed to downsample the feature maps, reducing their dimensionality while retaining the most important features. Max pooling, a common technique, selects the maximum value from each region of the feature map, thus capturing the most significant features and contributing to the model's ability to generalize across different text variations.

The output of the pooling layers is flattened and passed through fully connected layers, which integrate the learned features into a single, comprehensive representation. These fully connected layers then perform the classification task, mapping the high-level features to the target categories. By leveraging the hierarchical feature learning capability of CNNs, this approach enables the effective extraction and classification of complex patterns in text data.

Overall, CNNs are well-suited for text classification tasks due to their ability to learn and capture intricate patterns through their layered architecture, making them a powerful tool for analyzing and interpreting textual information.

In this work, three distinct Convolutional Neural Network (CNN) architectures have been evaluated to explore the interplay between network design and word embedding techniques for fake news detection, as shown in Table 1. The models assessed include variations in convolutional layers, pooling strategies, and dropout mechanisms. CNN Model 1 features a straightforward architecture with a single convolutional layer followed

by max pooling, designed to capture basic patterns in the text. CNN Model 2 incorporates additional complexity with batch normalization and global max pooling, aiming to improve feature extraction and model generalization. CNN Model 3 utilizes multiple convolutional and pooling layers to enhance the ability to capture hierarchical patterns in the data. Each model was tested with TF-IDF, Word2Vec, and FastText embeddings to determine which combination yields the highest performance in terms of accuracy, precision, recall, and F1 score. The results provide insights into how different CNN configurations interact with varying embeddings to optimize fake news detection.

**Table 1.** Summary of Convolutional Neural Network (CNN) models.

Model	Layer Type	Filters	Kernel Size	Activation	Other Details
CNN Model 1	Conv1D	128	5	ReLU	MaxPooling1D (pool size = 2)
	Flatten	N/A	N/A	N/A	Dense (64 units, ReLU)
	Dropout	N/A	N/A	N/A	Dropout (0.5)
	Dense (Output)	N/A	N/A	Sigmoid	
CNN Model 2	Conv1D	64	5	ReLU	BatchNormalization
	MaxPooling1D	N/A	2	N/A	Conv1D (128 filters, 5 kernel size)
	GlobalMaxPooling1D	N/A	N/A	N/A	Dense (64 units, ReLU)
	Dropout	N/A	N/A	N/A	Dropout (0.5)
	Dense (Output)	N/A	N/A	Sigmoid	
CNN Model 3	Conv1D	32	3	ReLU	MaxPooling1D (pool size = 2)
	Conv1D	64	3	ReLU	MaxPooling1D (pool size = 2)
	Flatten	N/A	N/A	N/A	Dense (128 units, ReLU)
	Dropout	N/A	N/A	N/A	Dropout (0.5)
	Dense (Output)	N/A	N/A	Sigmoid	

### 3.6. Performance Measures

Evaluating the efficacy of machine and deep learning models in the context of fake news detection involves a comprehensive assessment using several performance metrics. These metrics provide critical insights into the models' capabilities in accurately identifying false and genuine news articles, ensuring that the detection system is both effective and reliable.

- Accuracy quantifies the overall effectiveness of a model by measuring the proportion of correctly classified instances, including both true positives (TPs) and true negatives (TNs), relative to the total number of instances. It indicates the model's general performance but may not fully capture its effectiveness in detecting fake news if class distributions are imbalanced.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (7)$$

- Precision assesses the model's ability to correctly identify fake news articles, minimizing the rate of false positives (FPs). It is particularly important in scenarios where false alarms are costly, as it measures the ratio of true positives to the total number of instances flagged as fake news.

$$\text{Precision} = \frac{TP}{TP + FP} \quad (8)$$

- Recall evaluates the model's capability to detect all actual fake news instances. It calculates the proportion of true positives among the total number of actual fake news articles, including those that were missed (false negatives, FNs). High recall ensures that the majority of fake news articles are identified, which is crucial for minimizing the risk of overlooked false information.

$$\text{Recall} = \frac{TP}{TP + FN} \quad (9)$$

- F1 Score provides a balanced measure of a model's precision and recall by computing their harmonic mean. This metric is especially useful in fake news detection to balance the trade-off between precision and recall, ensuring that neither false positives nor false negatives dominate the evaluation.

$$F1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (10)$$

where:

- *TP* denotes the number of true positives, representing correctly identified fake news articles.
- *TN* denotes the number of true negatives, indicating correctly classified genuine news articles.
- *FP* denotes the number of false positives, representing genuine articles incorrectly classified as fake news.
- *FN* denotes the number of false negatives, indicating fake news articles that were incorrectly classified as genuine.

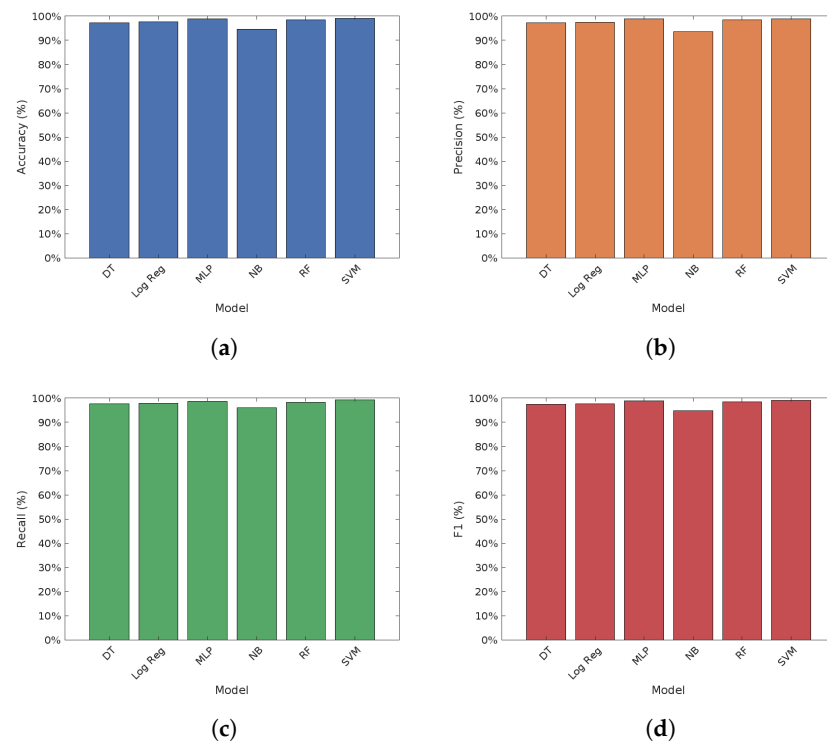
#### 4. Results

This section provides the evaluation results for both machine learning and deep learning models, under each of the considered word embedding techniques. All machine learning models were implemented using the Scikit-learn Python library, with default parameter values for each algorithm. The dataset was split into a training set and a testing set, with 70% of the data allocated for training and 30% for testing. This division ensures a balanced evaluation of model performance. All experiments were conducted using the High-Performance Computing (HPC) facility at the American University of the Middle East (AUM). The HPC system is equipped with Intel Xeon E5-2698 v3 processors running at 2.3 GHz, which provided the necessary computational power to efficiently execute our models and handle the extensive dataset.

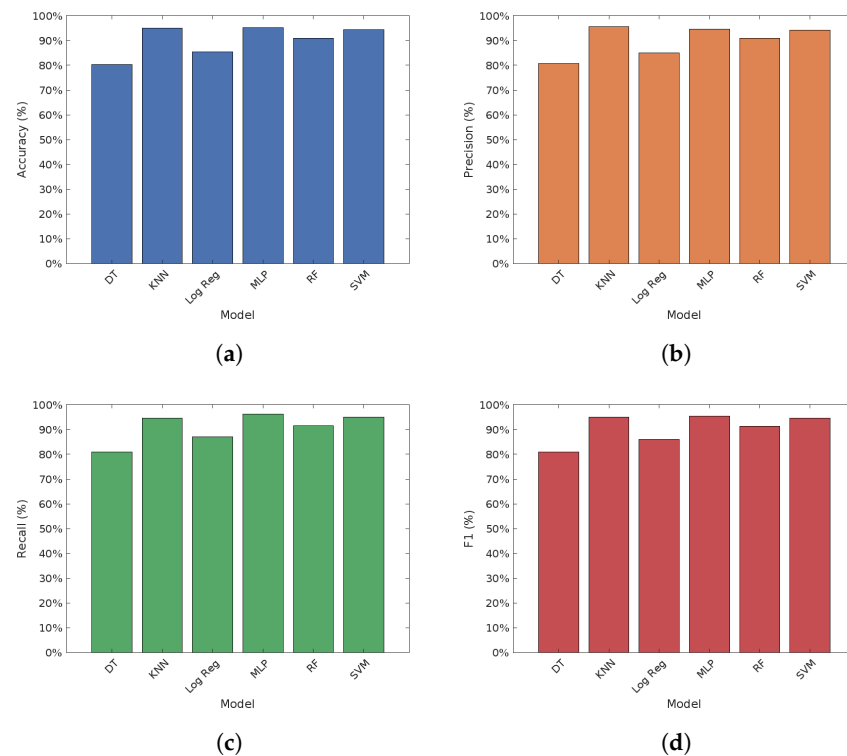
##### 4.1. Machine Learning Results

Figure 1 illustrates the performance of various machine learning algorithms on fake news detection, under the TF-IDF word embedding technique. The performance of various machine learning algorithms on fake news detection using the TF-IDF word embedding technique showcases notable differences. The Support Vector Machine (SVM) achieved the highest accuracy at 99.03%, along with the highest precision, recall, and F1 score, indicating its superior performance among the models tested. The Multilayer Perceptron (MLP) also performed well, with an accuracy of 98.77% and high precision and recall values. Logistic Regression showed strong performance with an accuracy of 97.58% and balanced precision and recall. Random Forest demonstrated a robust performance with an accuracy of 98.39%, while Decision Tree achieved an accuracy of 97.30%. Naive Bayes had the lowest accuracy at 94.55%, but still maintained reasonable precision and recall values. Overall, SVM and MLP were the top performers, followed by Logistic Regression and Random Forest, with Naive Bayes lagging behind in terms of accuracy.

On the other hand, Figure 2 depicts the performance of various machine learning algorithms on fake news detection using the Word2Vec word embedding technique. The results indicate significant variations in performance among the different algorithms.



**Figure 1.** Machine learning model performance—TF-IDF. (a) Accuracy; (b) precision; (c) recall; (d) F-1 score.



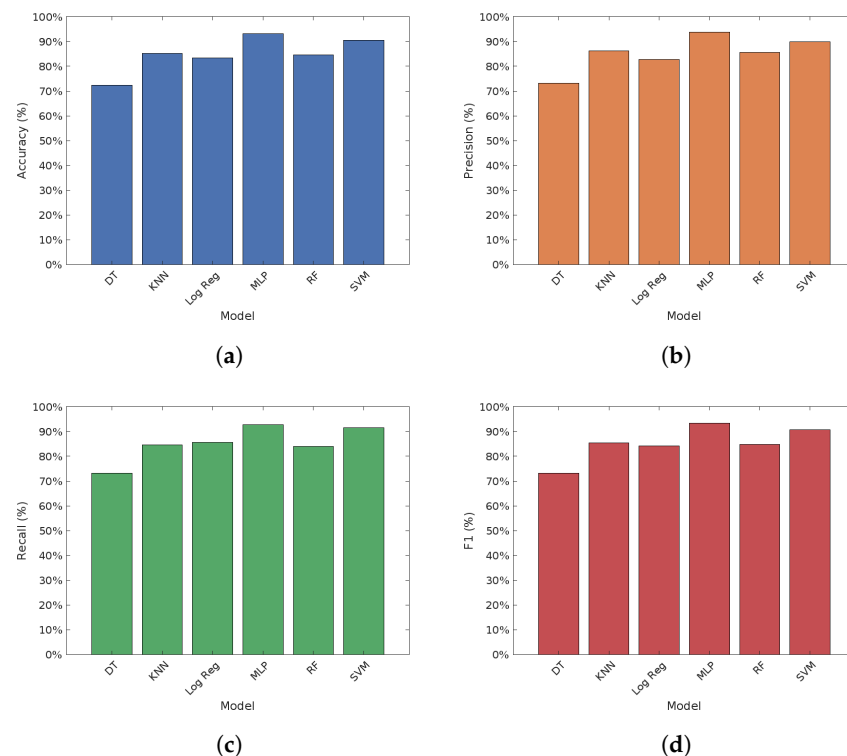
**Figure 2.** Machine learning model performance—Word2Vec. (a) Accuracy; (b) precision; (c) recall; (d) F-1 score.

The Multilayer Perceptron (MLP) achieved the highest accuracy of 95.24%, accompanied by the highest recall and F1 score, demonstrating its strong capability in detecting fake news. The K-Nearest Neighbors (KNNs) also performed impressively, with an accuracy



of 94.98% and high precision and F1 score. Random Forest showed solid results with an accuracy of 91.01%, while Support Vector Machine (SVM) achieved an accuracy of 94.47%. Logistic Regression had a lower accuracy of 85.42% but still maintained reasonable performance metrics. Decision Tree had the lowest accuracy at 80.30%, with comparatively lower precision, recall, and F1 score. Overall, MLP and KNN were the top performers under the Word2Vec embedding, followed by SVM and Random Forest, with Logistic Regression and Decision Tree lagging behind in terms of accuracy.

The performance of various machine learning algorithms on fake news detection using the FastText word embedding technique presents another set of observations, as shown in Figure 3. The Multilayer Perceptron (MLP) achieved the highest accuracy at 93.21%, along with the highest precision and F1 score, indicating its strong capability in detecting fake news. The Support Vector Machine (SVM) performed well with an accuracy of 90.41%, while K-Nearest Neighbors (KNN) showed an accuracy of 85.10% with reasonable precision and recall values. Logistic Regression had an accuracy of 83.44%, showing balanced performance metrics. Random Forest demonstrated an accuracy of 84.53%, while Decision Tree had the lowest accuracy at 72.42%. Overall, MLP and SVM were the top performers under the FastText embedding, followed by KNN and Random Forest, with Logistic Regression and Decision Tree lagging behind in terms of accuracy.



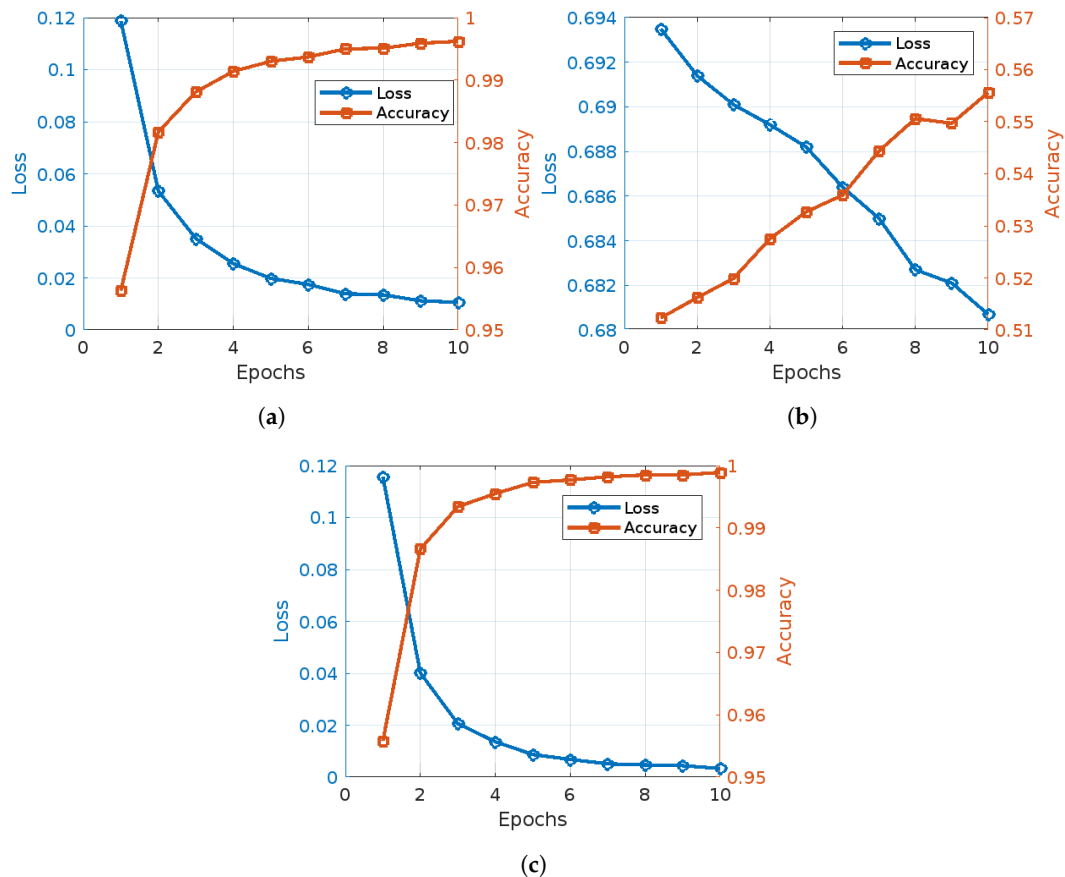
**Figure 3.** Machine learning model performance—FastText. (a) Accuracy; (b) precision; (c) recall; (d) F-1 score.

## 4.2. Deep Learning Results

### 4.2.1. Learning Curves Comparison

The training of three different CNN models with TF-IDF, Word2Vec, and FastText embedding techniques for fake news detection demonstrates various trends and observations in accuracy and training loss over the epochs, as shown in Figures 4–6.

Figure 4 illustrates the learning curves of the three CNN models under the TF-IDF word embedding technique; the main observations drawn from this figure can be summarized as follows:



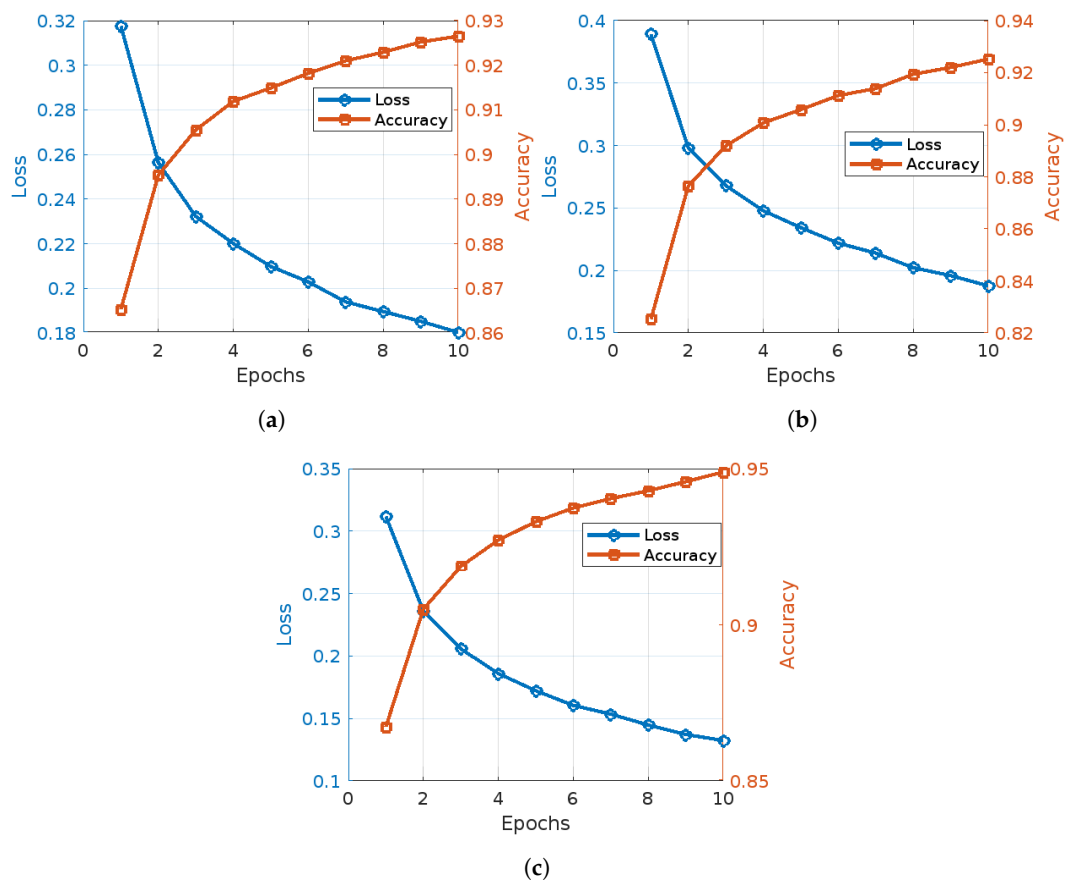
**Figure 4.** CNN model learning curves under TF-IDF. (a) CNN-1; (b) CNN-2; (c) CNN-3.

- CNN Model 1** (Figure 4a): The training of CNN Model 1 with TF-IDF embedding showed both a consistent and significant improvement in accuracy and reduction in loss over the epochs. Initially, the model began with an accuracy of 95.63% and a loss of 0.1187. As the training progressed to the second epoch, there was a notable improvement, with the accuracy increasing to 98.17% and the loss decreasing to 0.0534. This positive trend continued in the third epoch, where accuracy reached 98.81% and the loss further dropped to 0.0351. By the fourth epoch, the model's accuracy had improved to 99.14%, accompanied by a reduction in loss to 0.0255. Moving into the fifth epoch, the accuracy climbed to 99.30%, and the loss fell to 0.0198. In the sixth epoch, there was a slight increase in accuracy to 99.37%, with the loss further reducing to 0.0175. The seventh epoch saw the model achieving an accuracy of 99.50%, while the loss was further reduced to 0.0139. During the eighth epoch, the accuracy was 99.51%, with a minimal loss of 0.0135. By the ninth epoch, accuracy had increased to 99.59%, and the loss had decreased to 0.0112. Finally, in the tenth epoch, the model achieved its highest accuracy of 99.62%, with a loss of 0.0106.
- CNN Model 2** (Figure 4b): The training of CNN Model 2 with TF-IDF embeddings began with a significantly lower accuracy of 51.23% and a higher loss of 0.6935. This initial low performance can be attributed to the sparse nature of TF-IDF embeddings, which may not effectively capture the contextual relationships between words that are crucial for Convolutional Neural Networks. In the second epoch, the model showed a slight improvement with an accuracy of 51.62% and a reduction in loss to 0.6914. By the third epoch, the accuracy had increased to 51.99% and the loss had decreased to 0.6901. The fourth epoch observed a more substantial increase in accuracy to 52.75% and a decrease in loss to 0.6892. The fifth epoch saw the accuracy rise to 53.27% and the loss reduce to 0.6882. In the sixth epoch, accuracy improved to 53.59% with a

corresponding loss decrease to 0.6864. The seventh epoch achieved an accuracy of 54.44% and a loss of 0.6850. By the eighth epoch, the accuracy was 55.06%, with the loss decreasing to 0.6827. The ninth epoch saw a slight decrease in accuracy to 54.97%, but a reduction in loss to 0.6821. Finally, in the tenth epoch, the model attained its highest accuracy of 55.56%, with a loss of 0.6807. The gradual improvement in accuracy suggests that CNN Model 2 slowly adapted to the sparse feature space of the TF-IDF embeddings, learning to identify relevant patterns over time despite the initial challenges.

- CNN Model 3 (Figure 4c):** The training of CNN Model 3 with TF-IDF embedding displayed a strong performance throughout, starting with an initial accuracy of 95.58% and a loss of 0.1157. In the second epoch, the model showed a significant improvement, with accuracy increasing to 98.66% and the loss reducing to 0.0401. The third epoch saw a further increase in accuracy to 99.34% and a substantial drop in loss to 0.0206. By the fourth epoch, the accuracy had improved to 99.55% and the loss had decreased to 0.0136. The fifth epoch saw the accuracy rise to 99.73% with a loss of 0.0087. In the sixth epoch, the accuracy further improved to 99.77%, with a corresponding loss decrease to 0.0068. The seventh epoch achieved an accuracy of 99.82% and a loss of 0.0051. By the eighth epoch, the accuracy was 99.85%, with the loss decreasing to 0.0047. The ninth epoch saw a minimal increase in accuracy to 99.85%, with a slight reduction in loss to 0.0044. Finally, in the tenth epoch, the model achieved its highest accuracy of 99.89%, with a loss of 0.0033.

The training of three different CNN models with Word2Vec embedding for fake news detection exhibits distinct patterns and trends in accuracy and training loss over the epochs, as shown in Figure 5:



**Figure 5.** CNN models learning under Word2Vec. (a) CNN-1; (b) CNN-2; (c) CNN-3.

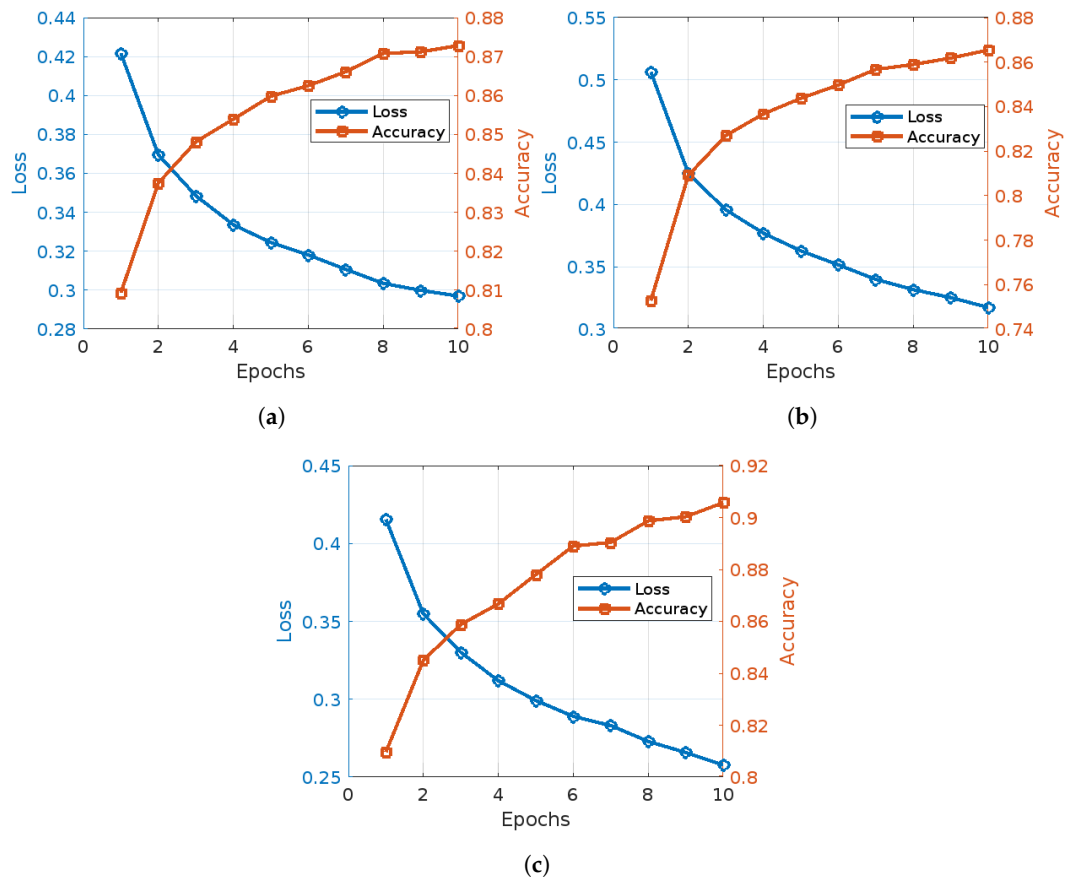
- **CNN Model 1** (Figure 5a): The training of CNN Model 1 with Word2Vec embedding displayed a steady improvement in both accuracy and reduction in loss over the epochs. Initially, the model started with an accuracy of 86.53% and a loss of 0.3176. By the second epoch, accuracy increased to 89.54% and the loss decreased to 0.2563. This positive trend continued, with the third epoch showing an accuracy of 90.55% and a loss of 0.2322. In the fourth epoch, the accuracy rose to 91.19% and the loss dropped to 0.2201. The fifth epoch saw further improvement, with accuracy reaching 91.49% and the loss falling to 0.2098. In the sixth epoch, accuracy increased to 91.82% and the loss decreased to 0.2029. The seventh epoch saw an accuracy of 92.10% and a loss of 0.1939. By the eighth epoch, the accuracy was 92.29% with a loss of 0.1896. The ninth epoch showed an accuracy of 92.52% and a loss of 0.1852. Finally, in the tenth epoch, the model achieved an accuracy of 92.65% with a loss of 0.1803.
- **CNN Model 2** (Figure 5b): The training of CNN Model 2 with Word2Vec embedding showed a different pattern, beginning with an accuracy of 82.53% and a loss of 0.3892. In the second epoch, the model's accuracy increased to 87.65%, while the loss decreased to 0.2982. The third epoch demonstrated an accuracy of 89.19% and a loss of 0.2680. By the fourth epoch, the accuracy had improved to 90.07% and the loss had dropped to 0.2478. The fifth epoch saw the accuracy rise to 90.58% and the loss decrease to 0.2343. In the sixth epoch, accuracy reached 91.12% with a loss of 0.2219. The seventh epoch showed an accuracy of 91.39% and a loss of 0.2139. By the eighth epoch, the accuracy improved to 91.94% and the loss reduced to 0.2022. The ninth epoch saw the accuracy increase to 92.20% and the loss drop to 0.1959. Finally, in the tenth epoch, the model achieved its highest accuracy of 92.52%, with a loss of 0.1877.
- **CNN Model 3** (Figure 5c): The training of CNN Model 3 with Word2Vec embedding displayed a noticeable performance throughout. The model began with an accuracy of 86.73% and a loss of 0.3117. By the second epoch, the accuracy improved to 90.50% and the loss decreased to 0.2361. The third epoch showed an accuracy of 91.87% and a loss of 0.2057. In the fourth epoch, the accuracy reached 92.71% and the loss dropped to 0.1860. The fifth epoch saw an accuracy of 93.30%, with a loss of 0.1722. In the sixth epoch, the accuracy improved to 93.74% and the loss decreased to 0.1605. The seventh epoch achieved an accuracy of 94.04%, with a loss of 0.1535. By the eighth epoch, the accuracy was 94.29%, with a loss of 0.1448. The ninth epoch saw an accuracy of 94.58% and a loss of 0.1372. Finally, in the tenth epoch, the model achieved its highest accuracy of 94.89%, with a loss of 0.1324.

The training of three different CNN models with FastText embedding for fake news detection demonstrates distinct patterns and trends in accuracy and training loss over the epochs, as shown in Figure 6:

- **CNN Model 1** (Figure 6a): The training of CNN Model 1 with FastText embedding displayed a steady improvement in both accuracy and reduction in loss over the epochs. Initially, the model started with an accuracy of 80.93% and a loss of 0.4217. By the second epoch, accuracy increased to 83.75% and the loss decreased to 0.3695. This positive trend continued, with the third epoch showing an accuracy of 84.81% and a loss of 0.3485. In the fourth epoch, the accuracy rose to 85.39% and the loss dropped to 0.3336. The fifth epoch saw further improvement, with accuracy reaching 85.98% and the loss falling to 0.3245. In the sixth epoch, accuracy increased to 86.25% and the loss decreased to 0.3181. The seventh epoch saw an accuracy of 86.61% and a loss of 0.3107. By the eighth epoch, the accuracy was 87.08% with a loss of 0.3035. The ninth epoch showed an accuracy of 87.12% and a loss of 0.2999. Finally, in the tenth epoch, the model achieved an accuracy of 87.28% with a loss of 0.2970.
- **CNN Model 2** (Figure 6b): The training of CNN Model 2 with FastText embeddings showed a different pattern, beginning with an accuracy of 75.28% and a loss of 0.5063. In the second epoch, the model's accuracy increased to 80.91% while the loss decreased to 0.4249. The third epoch demonstrated an accuracy of 82.70% and a loss of 0.3957. By the fourth epoch, accuracy had improved to 83.67% and the loss had dropped to

0.3770. The fifth epoch saw the accuracy rise to 84.37% and the loss decrease to 0.3628. In the sixth epoch, accuracy reached 84.97%, with a loss of 0.3516. The seventh epoch showed an accuracy of 85.67% and a loss of 0.3399. By the eighth epoch, the accuracy improved to 85.89% and the loss reduced to 0.3317. The ninth epoch saw the accuracy increase to 86.18% and the loss drop to 0.3253. Finally, in the tenth epoch, the model achieved its highest accuracy of 86.54%, with a loss of 0.3174.

- CNN Model 3 (Figure 6c):** The training of CNN Model 3 with FastText embedding displayed a relatively moderate performance throughout. The model began with an accuracy of 80.96% and a loss of 0.4155. By the second epoch, accuracy improved to 84.50% and the loss decreased to 0.3548. The third epoch showed an accuracy of 85.88% and a loss of 0.3301. In the fourth epoch, accuracy reached 86.67% and the loss dropped to 0.3119. The fifth epoch saw an accuracy of 87.30%, with a loss of 0.2992. In the sixth epoch, accuracy improved to 87.69% and the loss decreased to 0.2889. The seventh epoch achieved an accuracy of 88.25%, with a loss of 0.2802. By the eighth epoch, the accuracy was 88.47%, with a loss of 0.2725. The ninth epoch saw an accuracy of 88.72% and a loss of 0.2674. Finally, in the tenth epoch, the model achieved its highest accuracy of 88.99%, with a loss of 0.2618.



**Figure 6.** CNN model learning curves under FastText. (a) CNN-1; (b) CNN-2; (c) CNN-3.

#### 4.2.2. Performance Evaluation Results

Table 2 provides a comprehensive overview of the testing performance of the three CNN models under three different word embedding techniques: TF-IDF, Word2Vec, and FastText. The performance metrics—accuracy, precision, recall, and F1 score—highlight distinct differences and trends in the models' effectiveness for text classification tasks.

**Table 2.** Performance metrics for CNN models under different word embedding techniques.

Embedding	CNN Model	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)
TF-IDF	CNN Model 1	98.77	98.65	98.97	98.81
	CNN Model 2	56.15	55.69	70.93	62.39
	CNN Model 3	98.99	98.56	99.48	99.02
Word2Vec	CNN Model 1	94.25	93.69	95.20	94.44
	CNN Model 2	90.73	87.13	96.13	91.41
	CNN Model 3	94.92	94.64	95.50	95.07
FastText	CNN Model 1	89.32	88.34	91.21	89.75
	CNN Model 2	85.26	88.49	81.90	85.07
	CNN Model 3	89.55	88.60	88.72	89.96

The table shows that for the TF-IDF embedding, CNN Model 1 and CNN Model 3 demonstrate strong performance, achieving accuracy rates of 98.77% and 98.99%, respectively, indicating their robustness in handling this embedding. In contrast, CNN Model 2 performs poorly, with an accuracy of only 56.15%. This trend is consistent across other metrics, with CNN Model 1 and Model 3 showing high precision (98.65% and 98.56%), recall (98.97% and 99.48%), and F1 scores (98.81% and 99.02%), while CNN Model 2 lags significantly behind in all metrics.

When using the Word2Vec embedding, CNN Model 3 achieves the highest accuracy at 94.92%, closely followed by CNN Model 1 at 94.25%, while CNN Model 2 has a slightly lower accuracy of 90.73%. The precision for CNN Model 3 is also the highest at 94.64%, followed by CNN Model 1 at 93.69%, and CNN Model 2 at 87.13%. The recall values show a slight shift, with CNN Model 2 reaching the highest recall at 96.13%, while CNN Model 3 and CNN Model 1 follow closely at 95.50% and 95.20%, respectively. The F1 scores mirror this pattern, with CNN Model 3 leading at 95.07%, CNN Model 1 at 94.44%, and CNN Model 2 at 91.41%.

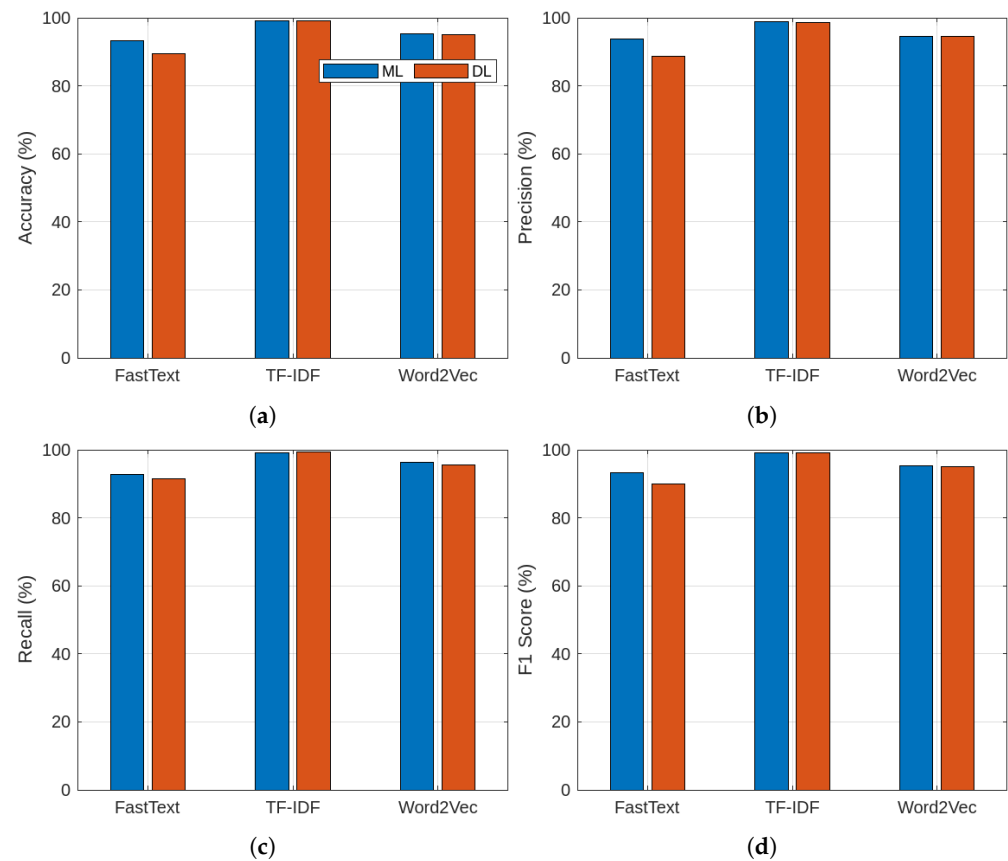
For the FastText embedding, the accuracy values are lower compared to TF-IDF and Word2Vec, with CNN Model 3 and CNN Model 1 achieving 89.55% and 89.32%, respectively, and CNN Model 2 at 85.26%. Precision is highest for CNN Model 2 at 88.49%, followed by CNN Model 3 at 88.60%, and CNN Model 1 at 88.34%. The recall metric shows CNN Model 1 achieving the highest value at 91.21%, followed by CNN Model 3 at 88.72%, and CNN Model 2 at 81.90%. In terms of F1 score, CNN Model 3 slightly outperforms the others at 89.96%, with CNN Model 1 at 89.75% and CNN Model 2 at 85.07%.

Overall, the results in Table 2 indicate that CNN Model 3 generally provides the best performance across most embedding techniques and metrics, particularly excelling with the TF-IDF embedding. CNN Model 2's performance is notably lower across all metrics, especially with the TF-IDF embedding, while CNN Model 1 consistently shows competitive performance close to that of CNN Model 3.

#### 4.3. Machine Learning vs. Deep Learning Comparison

Figure 7a–d present a detailed visual analysis of the top-performing machine learning (ML) model, Support Vector Machine (SVM), and deep learning (DL) model, CNN-3 (Convolutional Neural Network, Model 3), across various word embedding techniques, including TF-IDF, Word2Vec, and FastText.





**Figure 7.** Machine learning vs. deep learning performance comparison. (a) Accuracy; (b) precision; (c) recall; (d) F-1 score.

Figure 7a illustrates the accuracy metrics for the best ML and DL models under each embedding method. The results demonstrate that the SVM model frequently achieves superior or comparable accuracy to the CNN-3 model, with particularly notable performance gains observed for the TF-IDF embedding.

Figure 7b provides a comparative analysis of the precision scores for the models. The data reveal that the SVM model often outperforms or closely matches the precision levels achieved by the CNN-3 model.

Figure 7c,d depict the recall and F1 scores, respectively. These figures further indicate that the SVM model consistently exhibits higher or similar performance to the CNN-3 model across all embedding techniques.

These visualizations enhance the understanding of the comparative effectiveness of ML and DL approaches for fake news detection, highlighting the strengths of traditional machine learning models under specific embedding configurations.

## 5. Discussion

The comparative analysis of machine learning algorithms on fake news detection using three different word embedding techniques—TF-IDF, Word2Vec, and FastText—reveals several important insights. Overall, the Multilayer Perceptron (MLP) and Support Vector Machine (SVM) consistently emerged as the top performers across all embedding techniques. Specifically, MLP achieved the highest accuracy and performance metrics with Word2Vec at 95.24% and FastText at 93.21%, while SVM excelled with TF-IDF at 99.03%.

On the other hand, the TF-IDF embedding technique generally resulted in higher accuracy and performance metrics across most models compared to Word2Vec and FastText. This indicates that TF-IDF is more effective in capturing the relevant features for fake news detection, thus enhancing model performance. Conversely, Decision Tree consistently

exhibited the lowest performance across all three embedding techniques, with the lowest accuracy observed with FastText at 72.42%.

Furthermore, Logistic Regression and Random Forest demonstrated balanced but comparatively lower performance metrics than MLP and SVM, showing variability in accuracy depending on the embedding technique used. In particular, Logistic Regression achieved 97.58% accuracy with TF-IDF but showed lower performance with Word2Vec and FastText.

In summary, the choice of word embedding technique significantly impacts the performance of machine learning models in fake news detection. While TF-IDF generally provides the highest performance across most models, MLP and SVM consistently perform well regardless of the embedding technique employed. These findings underscore the importance of selecting appropriate embedding techniques to optimize the efficacy of machine learning algorithms in detecting fake news.

In evaluating the performance of CNN models with various word embeddings, CNN Model 3 with TF-IDF embeddings stands out as the top performer across all metrics. This combination achieved the highest accuracy (98.99%), precision (98.56%), recall (99.48%), and F1 score (99.02%), showcasing its exceptional classification capabilities. Following closely is CNN Model 1 with TF-IDF embeddings, which also delivered outstanding results with an accuracy of 98.77%, precision of 98.65%, recall of 98.97%, and F1 score of 98.81%. This makes CNN Model 1 with TF-IDF embeddings the second best performer overall. For stability, CNN Model 3 with Word2Vec embedding is notable for its consistent high performance, achieving an accuracy of 94.92%, precision of 94.64%, recall of 95.50%, and F1 score of 95.07%. Thus, CNN Model 3 with TF-IDF embedding is the preferred choice for achieving both the highest performance and stability, with CNN Model 1 with TF-IDF embedding being a close and highly competitive option.

The observed performance variations between different machine learning and deep learning models across various word embedding techniques can be attributed to several factors. For instance, the superior performance of the SVM model with the TF-IDF embedding can be linked to TF-IDF's effectiveness in capturing the frequency and importance of words, which aligns well with SVM's ability to handle high-dimensional feature spaces. This combination allows SVM to leverage TF-IDF's feature-rich vectors to make more accurate classifications.

On the other hand, deep learning models, such as CNN-3, showed notable performance with TF-IDF and Word2Vec embeddings. The CNN-3 model's ability to capture complex patterns in text data is enhanced by Word2Vec embeddings, which effectively capture semantic relationships and context between words. This combination helps CNNs build more robust feature maps that contribute to better performance metrics, especially in tasks requiring understanding of contextual nuances.

Moreover, the relatively lower performance of some models, like Decision Trees with FastText embeddings, could be due to FastText's subword information capturing ability, which might introduce noise that affects models that do not inherently perform feature selection, such as Decision Trees. Therefore, the interplay between model architecture and embedding technique is crucial, as it dictates how well the model can learn and generalize from the provided features. Future research could further explore this by analyzing the impact of more advanced embeddings like BERT or GloVe to identify whether similar patterns hold or if new dynamics emerge.

## 6. Conclusions

In this study, the performance of various word embedding techniques—TF-IDF, Word2Vec, and FastText—was systematically evaluated on both machine learning and deep learning models for fake news detection. The key findings indicate that TF-IDF and Word2Vec embeddings generally provide better performance across different models, particularly in distinguishing between genuine and misleading information. TF-IDF excels due to its ability to capture the importance of words in context, which is crucial for the

nuanced language often found in fake news. Word2Vec, on the other hand, captures semantic relationships between words, enhancing the model's ability to understand context and detect subtle patterns of misinformation.

The comparison between machine learning and deep learning models reveals that while deep learning models, such as Convolutional Neural Networks (CNNs), offer superior performance with more complex embeddings like Word2Vec, traditional machine learning models, such as Support Vector Machines (SVMs), perform comparably well with simpler embeddings like TF-IDF. This suggests that the choice of embedding technique should be aligned with the model type to optimize detection accuracy. Deep learning models are particularly effective in scenarios where capturing non-linear relationships and contextual information is crucial, whereas machine learning models are advantageous for tasks requiring less computational power and when simpler text representations are sufficient.

The implications of this research for social media platforms and information authenticity are significant. By demonstrating the effectiveness of specific embedding techniques and model combinations, this study provides a foundation for developing more robust fake news detection systems that can be implemented by social media platforms to enhance content verification processes. Improved detection models contribute to maintaining information authenticity, reducing the spread of misinformation, and promoting a more informed and trustworthy digital environment. These findings underscore the importance of continued innovation in natural language processing and machine learning to address the evolving challenges of fake news and misinformation.

**Author Contributions:** Conceptualization, M.A.B.A.-T., O.A.-i. and K.S.A.-M.; methodology, M.A.B.A.-T. and K.S.A.-M.; software, M.A.B.A.-T. and K.S.A.-M.; validation, M.A.B.A.-T., O.A.-i., K.S.A.-M. and W.H.F.A.; formal analysis, M.A.B.A.-T. and K.S.A.-M.; investigation, M.A.B.A.-T. and K.S.A.-M.; resources, M.A.B.A.-T. and K.S.A.-M.; data curation, M.A.B.A.-T. and K.S.A.-M.; writing—original draft preparation, M.A.B.A.-T.; writing—review and editing, M.A.B.A.-T., O.A.-i., K.S.A.-M., H.K. and W.H.F.A.; visualization, M.A.B.A.-T.; supervision, M.A.B.A.-T., K.S.A.-M. and H.K.; project administration, M.A.B.A.-T. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The data supporting the findings of this study are available within the article. The TruthSeeker dataset, which was utilized in this study, is publicly accessible through the Canadian Institute for Cybersecurity (CIC) at <https://www.unb.ca/cic/datasets/truthseeker-2023.html> (accessed on 1 July 2024). This dataset includes extensive metadata and auxiliary information used for analysis. Additional data and materials related to this study are available from the corresponding author upon reasonable request. No new data were created specifically for this study.

**Acknowledgments:** Machine and deep learning training and evaluation have been performed using the Phoenix High Performance Computing facility at the American University of the Middle East, Kuwait.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

Conv1D	Convolutional 1D
ReLU	Rectified Linear Unit
CNN	Convolutional Neural Network
TF-IDF	Term Frequency-Inverse Document Frequency

MLP	Multilayer Perceptron
SVM	Support Vector Machine
KNN	K-Nearest Neighbors
RF	Random Forest
DT	Decision Tree
NB	Naive Bayes
LR	Logistic Regression

## References

- Olan, F.; Jayawickrama, U.; Arakpogun, E.O.; Suklan, J.; Liu, S. Fake News on Social Media: The Impact on Society. *Inf. Syst. Front.* **2024**, *26*, 443–458. [\[CrossRef\]](#) [\[PubMed\]](#)
- Allcott, H.; Gentzkow, M. Social Media and Fake News in the 2016 Election. *J. Econ. Perspect.* **2017**, *31*, 211–236. [\[CrossRef\]](#)
- Gupta, A.; Roy, A. Manipulation of Social Media during the 2019 Indian General Elections. *Asian J. Commun.* **2019**, *29*, 537–556.
- Cinelli, M.; Galeazzi, A. The Covid-19 Social Media Infodemic. *Sci. Rep.* **2020**, *10*, 16598. [\[CrossRef\]](#) [\[PubMed\]](#)
- Meesad, P. Thai Fake News Detection Based on Information Retrieval, Natural Language Processing and Machine Learning. *Comput. Sci.* **2021**, *2*, 425. [\[CrossRef\]](#)
- Dadkhah, S.; Zhang, X.; Weismann, A.G.; Firouzi, A.; Ghorbani, A.A. The Largest Social Media Ground-truth Dataset for Real/fake Content: Truthseeker. *IEEE Trans. Comput. Soc. Syst.* **2024**, *11*, 3376–3390. [\[CrossRef\]](#)
- di Tollo, G.; Andria, J.; Filogrosso, G. The Predictive Power of Social Media Sentiment: Evidence from Cryptocurrencies and Stock Markets Using NLP and Stochastic ANNs. *Mathematics* **2023**, *11*, 3441. [\[CrossRef\]](#)
- Xie, T.; Ge, Y.; Xu, Q.; Chen, S. Public Awareness and Sentiment Analysis of COVID-Related Discussions Using BERT-Based Infoveillance. *AI* **2023**, *4*, 333–347. [\[CrossRef\]](#)
- Sufi, F. Social Media Analytics on Russia–Ukraine Cyber War with Natural Language Processing: Perspectives and Challenges. *Information* **2023**, *14*, 485. [\[CrossRef\]](#)
- Gamal, D.; Alfonse, M.; M. El-Horbaty, E.S.; M. Salem, A.B. Analysis of Machine Learning Algorithms for Opinion Mining in Different Domains. *Mach. Learn. Knowl. Extr.* **2019**, *1*, 224–234. [\[CrossRef\]](#)
- Ryciak, P.; Wasielewska, K.; Janicki, A. Anomaly Detection in Log Files Using Selected Natural Language Processing Methods. *Appl. Sci.* **2022**, *12*, 5089. [\[CrossRef\]](#)
- Hisham, M.; Hasan, R.; Hussain, S. An Innovative Approach for Fake News Detection Using Machine Learning. *Sir Syed Univ. Res. J. Eng. Technol.* **2023**, *13*, 115–124. [\[CrossRef\]](#)
- Khanam, Z.; Alwasel, B.; Sirafi, H.; Rashid, M. Fake News Detection Using Machine Learning Approaches. *Iop Conf. Ser. Mater. Sci. Eng.* **2021**, *1099*, 012040. [\[CrossRef\]](#)
- Nguyen, V.H.; Sugiyama, K.; Nakov, P.; Kan, M.Y. Leveraging Social Context for Fake News Detection Using Graph Representation. In Proceedings of the 29th Acm International Conference on Information & Knowledge Management, Virtual, 19–23 October 2020; pp. 1165–1174.
- Tam, N.T.; Weidlich, M.; Zheng, B.; Yin, H.; Hung, N.Q.V.; Stantic, B. From Anomaly Detection to Rumor Detection Using Data Streams of Social Platforms. *Proc. Vldb Endow.* **2019**, *12*, 1016–1029. [\[CrossRef\]](#)
- Park, M.; Chai, S. Constructing a User-centered Fake News Detection Model by Using Classification Algorithms in Machine Learning Techniques. *IEEE Access* **2023**, *11*, 71517–71527. [\[CrossRef\]](#)
- Hu, B.; Sheng, Q.; Cao, J.; Shi, Y.; Li, Y.; Wang, D.; Qi, P. Bad actor, good advisor: Exploring the role of large language models in fake news detection. In Proceedings of the AAAI Conference on Artificial Intelligence, Vancouver, BC, Canada, 20–27 February 2024; Volume 38, pp. 22105–22113.
- Wu, J.; Guo, J.; Hooi, B. Fake News in Sheep’s Clothing: Robust Fake News Detection Against LLM-Empowered Style Attacks. In Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Barcelona, Spain, 25–29 August 2024; pp. 3367–3378.
- Liu, H.; Wang, W.; Li, H.; Li, H. Teller: A trustworthy Framework for Explainable, Generalizable and Controllable Fake News Detection. *arXiv* **2024**, arXiv:2402.07776.
- Liu, H.; Wang, W.; Li, H. Interpretable multimodal misinformation detection with logic reasoning. *arXiv* **2023**, arXiv:2305.05964.
- Dai, S.; Li, K.; Luo, Z.; Zhao, P.; Hong, B.; Zhu, A.; Liu, J. Ai-based Nlp Section Discusses the Application and Effect of Bag-of-words Models and Tf-idf in Nlp Tasks. *J. Artif. Intell. Gen. Sci.* **2024**, *5*, 13–21. [\[CrossRef\]](#)
- Johnson, S.J.; Murty, M.R.; Navakanth, I. A Detailed Review on Word Embedding Techniques with Emphasis on Word2vec. *Multimed. Tools Appl.* **2024**, *83*, 37979–38007. [\[CrossRef\]](#)
- Umer, M.; Imtiaz, Z.; Ahmad, M.; Nappi, M.; Medaglia, C.; Choi, G.S.; Mehmood, A. Impact of Convolutional Neural Network and Fasttext Embedding on Text Classification. *Multimed. Tools Appl.* **2023**, *82*, 5569–5585. [\[CrossRef\]](#)
- Dharta, F.Y.; Januar Mahardhani, A.; Rachmawati Yahya, S.; Dirsia, A.; Usulu, M. Application of Naive Bayes Classifier Method to Analyze Social Media User Sentiment Towards the Presidential Election Phase. *J. Inf. Dan Teknol.* **2024**, *6*, 176–181. [\[CrossRef\]](#)
- Al-Tarawneh, M.; Muheilan, M.; Al Tarawneh, Z. Hand Movement-Based Diabetes Detection Using Machine Learning Techniques. *Int. J. Eng. Appl.* **2021**, *9*, 234–242. [\[CrossRef\]](#)

26. Leukel, J.; Özbek, G.; Sugumaran, V. Application of Logistic Regression to Explain Internet Use among Older Adults: A Review of the Empirical Literature. *Univers. Access Inf. Soc.* **2024**, *23*, 621–635. [[CrossRef](#)]
27. Zhang, J.; Li, Y.; Shen, F.; He, Y.; Tan, H.; He, Y. Hierarchical text classification with multi-label contrastive learning and KNN. *Neurocomputing* **2024**, *577*, 127323. [[CrossRef](#)]
28. El-Rashidy, M.A.; Mohamed, R.G.; El-Fishawy, N.A.; Shouman, M.A. An Effective Text Plagiarism Detection System Based on Feature Selection and Svm Techniques. *Multimed. Tools Appl.* **2024**, *83*, 2609–2646. [[CrossRef](#)]
29. Rashedi, K.A.; Ismail, M.T.; Al Wadi, S.; Serroukh, A.; Alshammari, T.S.; Jaber, J.J. Multi-Layer Perceptron-Based Classification with Application to Outlier Detection in Saudi Arabia Stock Returns. *J. Risk Financ. Manag.* **2024**, *17*, 69. [[CrossRef](#)]
30. Al-Tarawneh, M.A.B.; Al-Tarawneh, S.A.; Al-Maaitah, K.S. Predicting Processor Performance Using Machine Learning Techniques: A Study on SPEC CPU2017 Benchmark Suite. *Int. J. Eng. Trends Technol.* **2021**, *69*, 108–117. [[CrossRef](#)]
31. Reusens, M.; Stevens, A.; Tonglet, J.; De Smedt, J.; Verbeke, W.; vanden Broucke, S.; Baesens, B. Evaluating text classification: A benchmark study. *Expert Syst. Appl.* **2024**, *254*, 124302. [[CrossRef](#)]
32. Kanahuati-Ceballos, M.; Valdivia, L.J. Detection of Depressive Comments on Social Media Using Rnn, Lstm, and Random Forest: Comparison and Optimization. *Soc. Netw. Anal. Min.* **2024**, *14*, 44. [[CrossRef](#)]
33. Tong-Ern, T.; Su-Cheng, H.; Kok-Why, N.; Al-Tarawneh, M.; Gee-Kok, T. Performance Evaluation on Resolution Time Prediction Using Machine Learning Techniques. *JOIV: Int. J. Inf. Visual.* **2024**, *8*, 583–591. [[CrossRef](#)]
34. Taye, M.M. Understanding of Machine Learning with Deep Learning: Architectures, Workflow, Applications and Future Directions. *Computers* **2023**, *12*, 91. [[CrossRef](#)]
35. Li, Z.; Liu, F.; Yang, W.; Peng, S.; Zhou, J. A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *33*, 6999–7019. [[CrossRef](#)] [[PubMed](#)]

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.