On the Effect of Intelligent Neurally Processed Features on COVID–19 Fake News Detection Performance

Selva Birunda S[1], Kanniga Devi R[2], Suresh Kumar N[3], Muthukannan M[4]

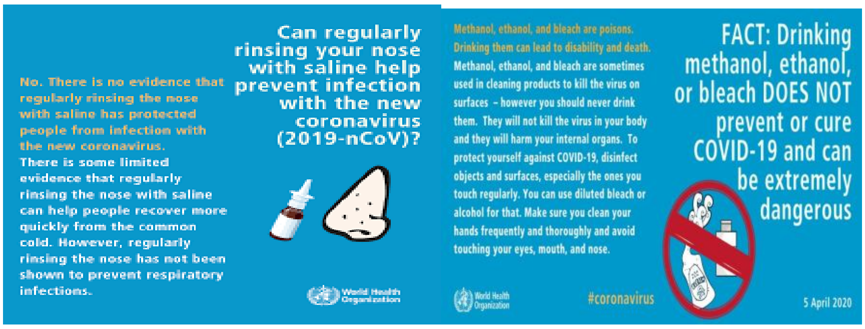
[1]*Department of Computer Science and Engineering, Kalasalingam Academy of Research and Education, Srivilliputhur, India, sbirunda89@gmail.com*
[2]*School of Computer Science and Engineering, Vellore Institute of Technology Chennai, Chennai, India*
[3]*Department of Computer Science and Engineering, Kalasalingam Academy of Research and Education, Srivilliputhur, India*
[4]*Department of Civil Engineering, KCG College of Technology, Chennai, India*

Online social networks are flooded with lot of user-generated information; fake news offenders use these online social network platforms to spread COVID fake news. This propagation of fake news results in a low level of content truthfulness, distrust in online social networks, panic and fear, which makes people take erroneous decisions. Hence, accurate classification of fake news against real news is mandatory. Therefore, in this study, a novel Neurally Augmented model is proposed to classify fake news accurately based on the content of the tweet. The proposed model creates a novel Deep Neural Network to automatically extract the neutrally processed features for downstream processing in a classic Machine Learning model. A neurally processed feature from 1D ConvNet is used to augment classic Machine Learning model in an attempt to improve the classification and discriminative capability of the classic Machine Learning model. Experimentation is done on a COVID-19 Twitter dataset curated by the authors. The proposed methodology provides a highly accurate fake news classification of 97.25%, which is 12% superior to the classic Machine Learning and Deep Learning models without neural augmentation. The proposed methodology is further evaluated on the ISOT, Indian Fake News Dataset, LIAR and Constraint Shared task COVID-19 Fake News benchmark datasets to determine its robustness, and it achieves significant accuracy of 93.75%, 89.17%, 83.68%, and 91%, respectively. The proposed model is tested on the proposed dataset and achieved a 5% increase in accuracy over the benchmark datasets.

Keywords: COVID-19, Fake News, Twitter, Infodemic, Deep Neural Network, Neural Augmentation, Machine Learning.

1 Introduction

Over the past few years, the rapid and vast adoption of online social networks has played a crucial role in connecting people, consuming news, and sharing it across the internet in large volumes [1]. People often fail to intensely evaluate the news shared on online social networks and confirm their beliefs. Consequently, this impact leads to the problem of the propagation of inaccurate information. This widespread tremendous fake news originated in the 2016 United States presidential election [2]. Fake news is intentionally written to convey false information, making it difficult to understand and decipher facts from fiction. Fake news exists on almost every Online Social Networks platform, including Facebook, Twitter, Instagram and WhatsApp [3]. The majority of fake news and rumours spread differently than genuine news via Twitter [4]. Twitter's public tweets were mined for unstructured data that takes various forms to propagate text, image, audio, and video [5, 6]. Recently, during the COVID-19 pandemic, coronavirus fake news threatened the public. Some fake information about COVID-19 treatments, medicines and cures on Twitter platform include 'Alcohol is a cure for COVID -19', 'Steaming in a bowl with towel can recover COVID-19', 'Eating garlic help prevent COVID -19 infection [7]. Some of the tweets claimed as fake cures by World Health Organization are depicted in Figure 1.


Can regularly rinsing your nose with saline help prevent infection with the new coronavirus (2019-nCoV)?

No. There is no evidence that regularly rinsing the nose with saline has protected people from infection with the new coronavirus.
There is some limited evidence that regularly rinsing the nose with saline can help people recover more quickly from the common cold. However, regularly rinsing the nose has not been shown to prevent respiratory infections.

World Health Organization

Methanol, ethanol, and bleach are poisons. Drinking them can lead to disability and death.
Methanol, ethanol, and bleach are sometimes used in cleaning products to kill the virus on surfaces – however you should never drink them. They will not kill the virus in your body and they will harm your internal organs. To protect yourself against COVID-19, disinfect objects and surfaces, especially the ones you touch regularly. You can use diluted bleach or alcohol for that. Make sure you clean your hands frequently and thoroughly and avoid touching your eyes, mouth, and nose.

FACT: Drinking methanol, ethanol, or bleach DOES NOT prevent or cure COVID-19 and can be extremely dangerous

World Health Organization

#coronavirus

5 April 2020

Figure 1: Tweets claimed as fake cure by World Health Organization.

Fake news on Twitter is a far bigger concern since it leads people to adapt drastic measures if they believe the information is real. This becomes the motivation for this work to classify real information against the fake news about COVID-19 based on the tweet content. Some of the conventional techniques utilized for solving the issue of fake news were network analysis, linguistic cue methods, knowledge-based approach, topic agnostic approach, Feature extraction, Machine Learning approach, Deep Learning approach, and hybrid approach, [37].

Social network behaviour and linked data were implemented as part of the network analysis method [36]. Deep syntax, sentiment analysis, data representation, and semantic analysis were all implemented as part of the linguistic cue method [36]. Crowdsourcing-oriented fact-checking, computational-oriented fact-checking, and expert-oriented fact-checking were implemented as part of knowledge-based approach [37]. Topic diagnostic approach identified fake news by utilizing web mark-up capabilities and linguistic features [37]. Feature extraction was an essential part of Machine Learning to process the text as it focused on creating features and data/word representations (vectors) from the raw data, which is helpful for training the Machine Learning model. The vector representation connected the human level of knowledge to the machine's understanding. Large datasets would require tremendous computing resources for processing. Hence, feature engineering could reduce the compiler's processing time and maximise the efficiency rate in recognizing the word's value

[8]. From [49], it is stated that feature engineering technique relies heavily on data domain expertise. However, [10] did not consider domain knowledge-associated features. Some of the feature extraction strategies that have been utilized to extract valuable features from textual information in classifying were Bag of Words, Term frequency – Inverted Document Frequency (TF-IDF), N-grams, Doc2Vec, Word2Vec, Latent Semantic Analysis, Glove and One-dimensional Convolutional Neural Network (1D-CNN) [10 – 21], [31]. The research work in [22] proposed an approach that automatically identifies applicable features related to fake news without any prior understanding of the domain by Long Short-Term Memory (LSTM) and CNN. As Deep Learning models allowed for automatic feature extraction, dependencies between words in bogus messages could be automatically recognized without having to explicitly define them in the networks [23].

Machine Learning techniques applied for the classification of fake news in online social networks are Naïve Bayes (NB), Support Vector Machines (SVM), Logistic Regression, Decision Trees, K-Nearest Neighbour (KNN), Random Forest (RF), XGBoost (XGB) and Neural Networks [10-17], [25,26]. Of which, SVM and Logistic Regression outperformed other algorithms for the characterization approach along with the TF-IDF technique [10, 13]. Four Machine Learning algorithms were employed for COVID-19 fake news classification: SVM with Linear kernel, Gradient Boosting (GB), Logistic Regression, and Decision Tree. Among these, SVM outperformed other algorithms with a better F1 score. The predictions are balanced among the two labels as they train and evaluate the model on the balanced dataset [7]. The authors of [35] suggested users use Natural Language Processing (NLP) to communicate with machines in human language. NLP model, namely, Passive Aggressive Classifier, was built for classifying the tweets with TF-IDF technique [21]. During COVID-19 pandemic, the researchers of [32] employed Machine Learning as well as Deep Learning algorithms to determine the shipment timeframes of diagnostics and vaccines in the supply chain. The authors of [33] utilized Machine Learning algorithms to address the issue of digit recognition. To detect the people propagating fake news on Twitter, [34] mined the users' tweets to obtain their 360-degree profiling using Machine Learning techniques, namely, Logistic Regression, Random Forest and Decision Tree.

Some of the Deep Learning models applied for the classification of fake news are Recurrent Neural Network (RNN), LSTM [10], Gated Recurrent Unit (GRU), CNN and Bidirectional Long Short Term Memory (BiLSTM). The authors of [21] compared NLP, LSTM and GRU models for predicting fake information on Twitter. Experimental results showed that LSTM and GRU models could not classify satire news and classify this as real news, whereas, NLP model could be able to classify this news as fake. Then, [22] applied three Deep Neural Network RNN variant models to train the dataset, including LSTM, LSTM with Drop out regularization, and LSTM integrating with CNN. Usually, RNN, as well as CNN, require more training data. However, the proposed LSTM variant achieved better performance. But, as suggested earlier, LSTM integrating CNN could not excel the LSTM variant as it had insufficient training data. It required even more training data. Then, [25] experimented that neural network models were showing better performance on the large dataset of over 100000 samples. Besides, [12] demonstrated how the abundance of corpora could help to increase the performance. Using the enriched dataset had a considerable impact on the accuracy. Next, Modified LSTM and Modified Gated GRU with one to three layers were applied to classify COVID-19 tweets. Both the techniques outperformed six Machine Learning algorithms employed by [17].

A hybrid method was proposed to classify the fake news using COVID-19 dataset. Hybridization combined numerous CNN branches with LSTM layers of various kernel sizes and filters. The pooled features were extracted from CNN and were fed into RNN variant LSTM for training and classification of news articles [20]. The Deep Hybrid Learning approach comes in different forms like early fusion and late fusion. In Early fusion, prior to feature extraction, the fusion process happens. In Late fusion, the features are extracted, and the fusion process happens [50]. The authors of [30] utilized the late fusion technique by utilizing Deep Neural Network (1D-CNN) as a feature extractor from the Bangla news and Machine Learning algorithms, namely, Random Forest, AdaBoost, K-Nearest Neighbours, SVM and Decision Tree for classification of fake and real news.

The contribution of this research is as follows:

- A novel scrapping technique is developed for creating COVID-19 Twitter dataset for fake news detection
- A novel Deep Neural Network feature extractor algorithm is built to neurally process the tweets from the created dataset
- A novel Augmenter layer is constructed to extract the neurally processed feature from the Deep Neural Network feature extractor to augment the Machine Learning classification layers
- A novel Neurally Augmented Machine Learning classification layers namely, Neurally Augmented Naive Bayes, Neurally Augmented Random Forest, Neurally Augmented Gradient Boosting and Neurally Augmented Extreme Gradient Boosting, which do not overfit the training procedure are built for assessing the classification performance of the proposed model

2 Literature Survey

This section narrates a detailed view of the existing methodologies and datasets applied for fake news identification and the effect of each methodology, along with its pros and cons.

Network analysis required collective human knowledge to determine the truth of new information and verify the veracity of claims. However, this approach had been employed for the conference call records, e-mails and not for social media tweets or microblogs [36]. Linguistic cue method found the fake news by the content writing style of the information manipulators [36]. However, this approach was imperfect because the issues of verification and credibility were given less priority [36]. Expert-oriented fact-checking required professionals to manually evaluate the credibility of the news. Crowd-sourcing fact-checking allowed collective decisions from the group of people to examine the truthfulness of the news. Computational-oriented fact-checking checked the news as authentic or fake automatically. This approach identified fake news by ignoring the article content and considering topic features alone [37].

2.1 Datasets

Some of the existing fake news datasets for COVID-19 were COVID-19 FN dataset, FakeCovid, ReCOVery, CoAID and FANG-COVID. COVID-19 FN dataset comprised of

10,700 tweets in English language. FakeCovid dataset contained 5,182 news articles in many different languages with 2, 116 news articles alone in English Language. ReCOVery dataset included 2, 029 news articles in English language. CoAID dataset comprised of 4,251 news articles, social media posts and claims in English Language. FANG-COVID dataset included 41, 242 German news articles [FANG-COVID].

2.2 Feature Engineering in Machine Learning and Deep Learning

The effect of feature extraction in Machine Learning and Deep Learning applied for classifying fake news, along with the pros and cons are as follows: Bag of Words technique calculated the individual word frequency. This word frequency aided in the identification of word usage patterns [37]. Bag of words technique failed to consider the context and position of a word, when the textual content was converted into vector representations. Besides, these representations resulted in data sparsity issues while considering the social media news content [37]. N-gram was used to determine the probability distribution over word sequence of ‘n’ length and was capable of retaining the context [17]. While using n-grams, the model trained on one genre did not provide accurate predictions on another genre. If the training and test corpus were not similar, n-grams could not give better results. Besides, two of the challenges of n-grams were the Out of Vocabulary (OOV) words and sparse data issues [17]. When using machine learning algorithms to do text classification, TF-IDF was a better feature choice than N-Gram [38]. Relevance or importance of the word in a document could be quantified using TF-IDF by assigning weightage to the words [19]. TF-IDF technique yielded sparse data samples, particularly for the Online Social Network data, as it had only 100 or fewer words [19]. These vector representations of words were domain-specific [20]. Latent Semantic Analysis, a dimensionality reduction technique, was used to minimize the number of features than the previous techniques by preserving the original semantic structure of the space and original characteristics variance [14]. Latent Semantic Analysis performed well for the long documents. Moreover, because of the large size of the data, the efficiency of LSA was reduced as it required more computing time and extra storage space. LSA did not perform well on the analogy tasks. Feature engineering was domain-specific, time consuming and required human involvement and domain knowledge. Performing automatic feature engineering to extract features automatically without any direct human involvement was a major challenge in Machine Learning. Hence, depending on Deep Learning, feature extraction was employed to extract the features automatically.

Doc2Vec generated the vector representation of the document irrespective of length. Doc2Vec and TF-IDF worked well for linear classification tasks. However, TF-IDF data representations outperformed the Doc2Vec representations [42]. Word2Vec expressed each word in the textual content as dense vector representations with unified dimension and meaning. Word2Vec was capable of capturing word-to-word relationships, including semantic and syntactic relationships. In the case of a data sequence, there was a strong correlation among the sequence's local data. The adjacent words were highly related, and even a word's context could approximately predict the middle word. As vectors and words are one-to-one, the problem of polysemous (word with multiple meanings) words could not be solved. Word2Vec failed to capture the word co-occurrence at a global level. Besides, word2vec could not handle Out of Vocabulary (OOV) words. GloVe technique performed better than word2vec in analogy tasks. While constructing the vectors, GloVe considered the word pair-word pair relationship than the word-word relationships, as it tends to provide significant meaning to the vectors. GloVe technique utilized global information and co-occurrence matrix. 1D-CNN was used to learn and extract the local features automatically from the training data [20]. Deep Learning models did not necessitate manual feature engineering; instead, they have the auto feature extraction capability [24].

2.3 Machine Learning

The effect of Machine Learning techniques applied for classifying fake news and the pros and cons are as follows: Using experience, Naive Bayes classifier predicted the membership probabilities for each class [36]. Naïve Bayes classifier was recommended as a better classifier for the limited dataset by [25]. Naïve Bayes assumed the features to be unrelated and independent. Support Vector Machine performed well on more concise and smaller datasets. It was capable of dealing with high-dimensional spaces and was memory efficient. SVM was flexible because it could be used to determine or categorize numbers. However, it took more training time on larger datasets [36]. Logistic regression could classify binary inputs because of better predictive performance in probability values. It performed well with both short and long input text, but, it struggled to capture complex relationships [41]. Decision tree transformed the data into the representation of a tree. Decision Tree had a high likelihood of overfitting [15]. During the training period, KNN learned nothing as it was a lazy learner, so it took less training time and much faster than other classifiers. However, it failed to work well with large datasets [17]. Ensemble learners, namely Random Forest, Gradient Boosting, and XGBoost, produced high accuracy because using a specific learning technique, multiple models were trained to increase the model's performance and lower the total error rate. Gradient Boosting tried to reduce the errors of the weak learner models and combine the predictions from the weak learners to become a strong learner. The extension of Gradient Boosting technique is the XGBoost, which reduces the time computation and computational complexity of the Gradient Boosting technique [26, 29, 48].

2.4 Deep Learning

Rather than Machine Learning algorithms, Deep Learning models worked very well with the unstructured data [24]. CNN and RNN could identify the patterns in the text data that are complicated [27].RNN could process a variety of word embedding vectors of any dimension. However, RNN had the drawback of memory accesses and gradient exploding. Hence, LSTM was created to address the above issue [28, 9]. LSTM was employed to examine data with varying lengths of time. Large data sequences have long been a problem for neural networks. Hence, [13] employed LSTM to handle a large data sequence for the fake tweet classification purpose. However, LSTM failed to focus on the whole context, which only process the text in the forward direction.NLP model took into account the context and the tone of an article, whereas the LSTM and GRU failed to consider [21]. Thus, the context information offered by the upcoming words will be ignored in this scenario. BiLSTM was applied to process the input sequence from both forward and backward directions to classify fake news [28]. The recent battle against fake news on COVID-19, as well as the uncertainty surrounding it, demonstrated the need for a hybrid approach to classify fake news [37].

The summary of Feature Extraction, Machine Learning, and Deep Learning techniques used for fake news classification in existing literatures are depicted in Figure 2.

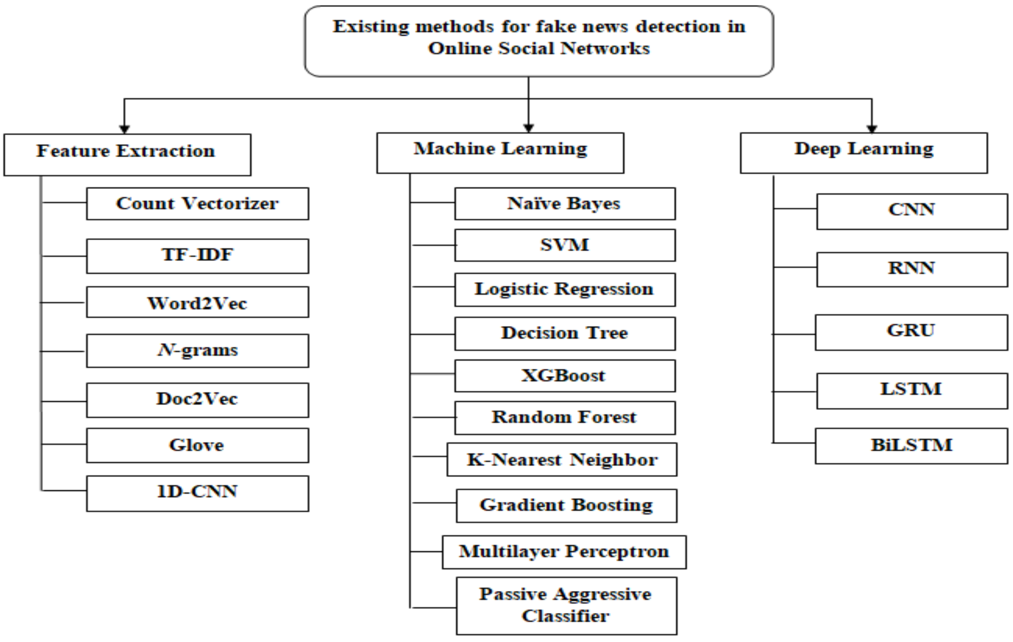


Figure 2: Summary of Feature Extraction, Machine Learning, and Deep Learning techniques used for fake news classification in existing literatures.

2.5 Outcome of the literature survey

From the above literature survey, it is observed that the classification performance of the Machine Learning and Deep Learning techniques are influenced by three factors: Dataset, Feature Engineering as well as Model architecture defined or determined by hyper parameters of the model. When it comes to dataset, the existing datasets for the COVID-19 fake news detection is too small to be processed by the Deep Learning models. While considering feature engineering, the existing feature extractors require handcrafted feature engineering and domain knowledge. Besides, the feature extractor fails to extract the features automatically and consider the semantic nature of the text. The existing feature extractors learn and process the features repeatedly by consuming maximum time and memory storage. While considering model architecture, the existing fake news detection models utilize either Machine Learning or Deep Learning techniques. Hybridization of these techniques is essential to improve the classification performance. Hence, there is a need to solve these issues to improve the classification performance.

To address the above issues, there is a need to create a COVID-19 Twitter dataset with large quantity of tweets and a novel feature extractor, which can preserve the semantic nature of the tweet during feature extraction, which in turn reduce the processing time and memory consumption. In addition, there is a need for a novel model to neurally process the tweets and uses it to augment the Machine Learning model. Hence, this research work focuses in these aspects to improve the classification performance.

3 Proposed Methodology

The core idea of this work is, the use of Deep Neural Network to augment the classic Machine Learning, which has a significant impact on the COVID-19 fake news classification performance. The section 3.1 presents the proposed architecture. The section 3.2 – 3.4 present the COVID-19 twitter dataset creation, tweet pre-processing and tweet vector representation. The section 3.5 – 3.7 present the Neurally Augmented Model (NAM), which is constructed by integrating three Convolutional layers, three MaxPooling layers, Fully Connected layer, Augmenter layer and Neurally Augmented (NA) Machine Learning classification layer in its architecture.

3.1 Proposed Architecture

The proposed architecture operates in two phases. The phase-1 creates the Deep Neural Network feature extractor to neurally process the tweets and extract the neurally processed features. The phase-2 creates an Augmenter layer, which extracts the neurally processed features from the fully connected layer of phase-1 and uses it to augment the classic Machine Learning classification layers. The overall architecture of the proposed model is depicted in Figure 3.

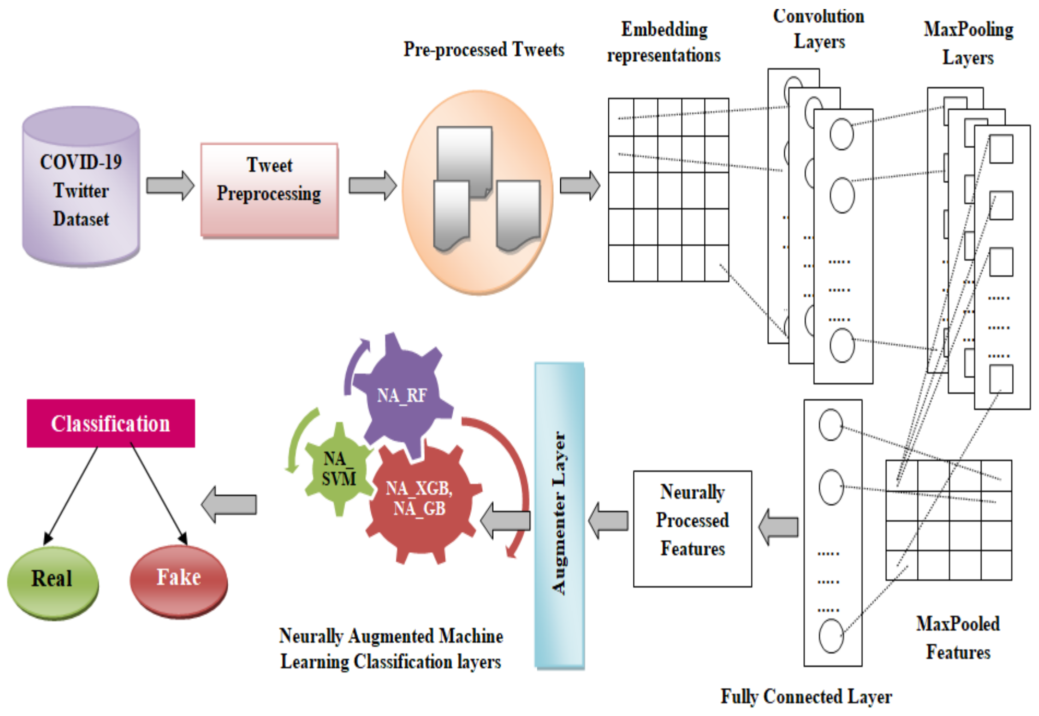


Figure 3: Architecture of Neurally Augmented Model.

3.2 Dataset

This research work developed a novel snscrape script in python to create 50,000 tweets on

COVID-19 dataset in English language from Twitter. The COVID-19 keywords and hashtags namely, “SARS-CoV-2”, “coronavirus”, “covid”, “@CDC”, “@WHO”, “vaccine”, and “ncov2019” are provided as input to extract the tweets published on the duration from January 2020 to January 2022. Using the script, 30000 tweets are gathered by crawling tweets from official and verified Twitter handles of sources such as, the World Health Organization, the Centers for Disease Control and Prevention, the Government of India, and the Indian Council of Medical Research. These 30000 tweets are real tweets as they are collected from government authentic information sources. The tweets are annotated as real as they provide useful information regarding government policies concerning COVID-19, safety guidelines, and the significance of vaccination. The remaining 20000 tweets are collected from general public posts on Twitter, unverified Twitter accounts which may not be an authentic source of information and hence considered fake tweets. The collected tweets have three attributes: Tweet id, Tweets, and Label. Tweet id is the unique identification of tweets. Tweets are running text with 140 characters. Label is the tag added to the tweets that notify a model of a specific tweet so it may learn from it. Some samples of Real and Fake COVID-19 tweet information of the dataset are illustrated in Table.1.

COVID – 19 News Information	Source	Label
The [#COVID19] Delta variant is dangerous and is continuing to evolve & mutate, which requires constant evaluation & careful adjustment of the public health response. - @DrTedros	(WHO) Twitter	Real
A CDC study shows mRNA #COVID19 vaccines reduce risk of infection by 91%. If you are vaccinated & still get COVID-19, there are other benefits of vaccination, like fewer sick days & reduced risk of symptoms like fever or chills.	(CDC) Twitter	Real
ICMR study shows #COVAXIN neutralises against multiple variants of SARS-CoV-2 and effectively neutralises the double mutant strain as well.@MoHFW_INDIA@DeptHealthRes#IndiaFightsCOVID19#LargestVaccineDrive	(ICMR) Twitter	Real
The symptoms of COVID-19 Delta Variant do not include fever and cough and Delta variant gives negative RT-PCR results.	Facebook, Twitter	Fake
The protein S generated by COVID-19 vaccines is toxic and damages many tissues in our body.	Twitter	Fake
Anaesthesia must not be used to those who have just received Covid-19 vaccine	Social Media	Fake

Table 1. Sample dataset.

3.3 Tweet pre-processing

The created COVID -19 Twitter dataset should be free from noisy and missing data for the model to accurately classify the reliable information against the fake news. For this, pre-processing is required on the COVID-19 Twitter dataset, DS. The DS consisting of n tweets is represented as DS = $\{(T_1, Y_1), (T_2,Y_2),.....,(T_n, Y_n)\}$, where, $Y \in \{real, fake\}$. Now, each T_i gets pre-processed by removing noisy, duplicate and missing data to reduce the storage space and computation time. Next, the irrelevant data in tweets, such as, tweet url, square brackets, double spaces, and punctuation, are deleted as they are unnecessary while processing the textual content. Then, the Natural Language Processing technique, such as tokenization is done on the tweets to split the tweets as individual tokens. Tokenization is performed using Keras library’s Tokenizer function. Once the tweets get tokenized, the number of words in the tokens is W. Then the stop words are removed from the tweets as these are less informative words that have no contribution while processing the text. Next, lowercasing is applied to alter the case of words. Finally, stemming is done to reduce a particular word to its root word. Finally, the tweets in DS are pre-processed and are ready for actual processing. The obtained pre-processed dataset, denoted by PD = {(PT_1, Y_1), (PT_2, Y_2),......,(PT_n, Y_n)}, can maximize the accuracy of the classification model. The entire process of tweet pre-processing is explained in algorithm 1.

```
Algorithm 1: Covid-19 Tweet Pre-processing
Input: DS consisting of tweets $T_{i=}$ {($T_1$, $Y_1$),  ($T_2$, $Y_2$),....., ($T_n$, $Y_n$) }
Output: Pre-processed Dataset, PD
for each $T_i \in$ DS
  preprocess ($T_i$):
        $PT_i \leftarrow \phi$
        PD $\leftarrow \phi$
      //URL Removal
            $PT_i \leftarrow$ Eliminate ($T_i$, “https (s)?://(.||[a-z]||[A-Z]||[0-9]+”]))
      // Punctuation removal
        for each char (c) in $PT_i$ :
          if c $\notin$ punctuations:
              $PT_i$ = $\leftarrow$ $PT_i$ U c
          end if
        end for
      // Tokenization
            tokens $\leftarrow$ Tokenization ($PT_i$)
      // Stopword removal
        W $\leftarrow$ Number of words in tokens
        S $\leftarrow$ Number of words in Stop words list
        for j = 1 to W do
        for i = 1 to S do
          if W(j) == S (i) then
              $PT_i \leftarrow$ eliminate W(j)
          end if
        end for
        end for
      // Lowercasing
        for q in range (len($PT_i$)):
          if $PT_i$ [q].isupper():
              $PT_i \leftarrow PT_i$ U $PT_i$ [q].lower()
          end if
        end for
        for each $PT_i$, apply PorterStemmer function (PS)
              $PT_i \leftarrow$ PS ($PT_i$)
        end for
        PD $\leftarrow$ PD U $PT_i$
      end for
return PD
```

3.4 Tweet vector Representation

Once the pre-processed dataset, PD is obtained, each tweet, PT_i ($1 \leq i \leq n$) consists of a sequence of words. This sequence of words has to be converted into embedding vector representations to be processed by the Deep Neural Network feature extractor to extract the features from it. Initially, the sequence of words gets tokenized into individual words as PT_i = {w_1, w_2,.,....,w_m}by the task of tokenization.

Next, each word w_i ($1 \leq i \leq m$) in PT_i is embedded using the Glove pre-trained word embedding ‘glove.twitter.27B.100d.txt’ Dictionary (GD) to generate the embedding vectors.

Each word w_i is represented as q dimensions. The 'q'-dimensional embedding vectors of each word is generated by mapping w_i with each word Dw_i in GD. $\mathbb{R}^q$ is the q-dimensional vector space of words. $w_m \epsilon \mathbb{R}^q$ is the q-dimensional embedding vector of the mth word. Similarly, the dense embedding vectors for all words in the tweets consisting of features with q dimensions f_1^q is generated and is represented as in (1),

$$ET = \{w_{1,f_1^q}, w_{2,f_1^q}, \dots\dots, w_{m,f_1^q}\} \qquad (1)$$

The pictorial representation of tweet vector representation is illustrated in Figure 4.

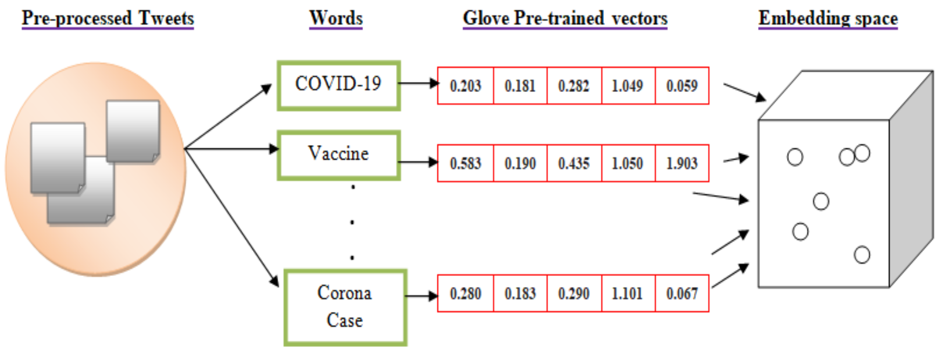


Figure 4: Tweet vector representation by mapping words into numerical vectors.

Once the embedding vectors are generated, the semantically related vectors are identified to reduce the size of embedding vectors, in order to get the reduced optimal features. Hence, to obtain semantic-aware embedding vector representations, semantic similarity is calculated for all the embedding vectors. The general manner of similarity calculation in Vector Space Model is modified by considering similarity among the features. The semantic similarity between the embedding vectors $(w_{1,f_1^q}, w_{j=i+1,f_1^q})$ is calculated for all the feature dimensions of the words and is measured by (2),

$$\text{simf}\,(w_{i,f_1^q}, w_{j=i+1,f_1^q}) = \frac{w_{i,f_1^q} \; * \; w_{j=i+1,f_1^q}}{\sqrt{w_{i,f_1^q}, w_{i,f_1^q}} \; \sqrt{w_{j=i+1,f_1^q}, w_{j=i+1,f_1^q}}} \qquad (2)$$

where,

$w_{i,f_1^q} \; * \; w_{j=i+1,f_1^q}$ indicate the product of vectors

$\sqrt{w_{i,f_1^q}, w_{j=i+1,f_1^q}}$ indicate the length of the vector w_{1,f_1^q} of d dimensions

$\sqrt{w_{j=i+1,f_1^q}, w_{j=i+1,f_1^q}}$ indicate the length of the vector $w_{j=i+1,f_1^q}$ of d dimensions

Based upon the simf value, the two embedding vectors $(w_{1,f_1^q}, w_{j=i+1,f_1^q})$ with similar value are combined into single vector representation. Similarly, the semantic similarity is calculated for all embedding vectors in ET and similar representations are provided for the semantically related words. These similar representations can effectively reduce the memory storage to a large extent. The final embedding representation is updated in ET.

Phase – 1

3.5 1D ConvNet feature extractor

A Deep Neural Feature extractor namely, 1D ConvNet is proposed, which automatically extracts the features without handcrafted feature approach designed by experts, as they are self-learners. In addition, it has salient properties such as, sub sampling and weight sharing, which reduces the complexity of the neural network structure and parameters. Besides, this feature extractor is built to neurally process the tweets and extracts the optimal features out of it. Initially, the obtained embedding representations ET is given as input to the proposed feature extractor as the initial embedding layer. This is followed by three Convolutional layers, three Max Pooling layers and a fully connected layer along with Dropout and Zero Padding in its architecture.

Each Semantic-Aware embedding representations $\{w_{1,f_1^q}, w_{2,f_1^q}, \ldots\ldots, w_{m,f_1^q}\}$from ET enters the first 1D Convolutional layer (l_1) with hyperparameter tuned are 128 filters, kernel size of 5, activation function 'relu', MaxPooling1D layer with a pool size of 2 and a regularization dropout layer of dropout rate as 0.25. This dropout layer is added to minimize over fitting. Now, slide 1D Kernel of size (5 X 1) over the input embedding representations to obtain the 1D Convolutional output feature maps. The generated output feature maps are of small size. Hence, to control the reduction of output dimension while applying large filter size, zero padding is added before the 1D convolution block. By applying zero padding, optimal features are extracted from the Convolutional layer. Then Maxpooling1D layer reduces the feature representation's dimensionality, however preserving the significant informative features.

The operation of 1St 1D Convolutional layer (l_1) is illustrated in Figure 5.

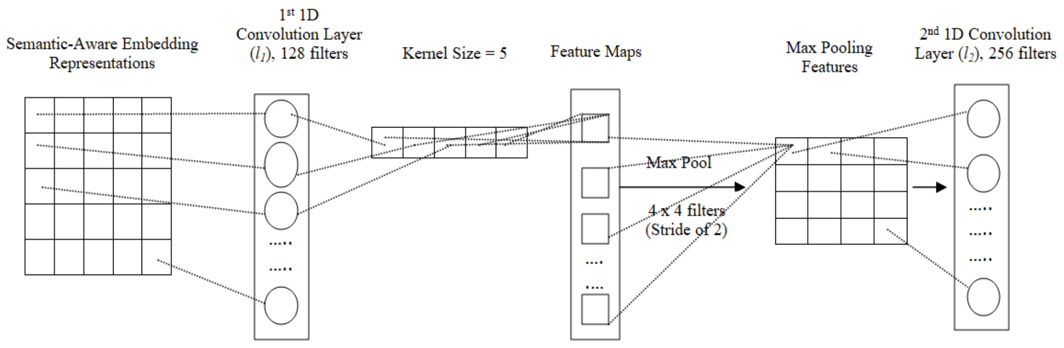

Semantic-Aware Embedding Representations
1st 1D Convolution Layer (l_1), 128 filters
Kernel Size = 5
Feature Maps
Max Pooling Features
2nd 1D Convolution Layer (l_2), 256 filters
Max Pool
4 x 4 filters (Stride of 2)

Figure 5: Computation of 1D Convolutional operation.

The next convolution layer (l_2) having 256 filters, kernel size of 4, activation function 'relu', dropout layer with a dropout rate of 0.25, and 2nd MaxPooling1D layer with a pool size of 2.

This layer is followed by one more convolution layer (l_3) with 256 filters, kernel size of 5, activation function ‘relu’, dropout layer with a dropout rate of 0.25, and 3rd MaxPooling1D layer with a pool size of 4. The fully connected layer has input with 256 units, dropout rate as 0.25 and ‘relu’ as the activation function. This fully connected layer is the last layer of the feature extractor.

The hyperparameter configurations of 1D-ConvNet feature extractor are illustrated in Table.1.

Table 1: Hyperparameter Configurations of 1D ConvNet feature extractor.

Name of the parameter	Value
Max_Sequence_Length	500
Embedding dimensions	100
Number of 1D convolution	3
Number of filters in each layer	128, 256
Kernel size	4, 5
Number of MaxPooling 1D	3
Pool size	2, 4
Activation function	ReLU
Number of hidden units present	256
Number of flatten layer	1
Number of dense layer	1
Dropout	0.25
Strides	2

Then the size of the output feature map by projecting the input on the kernel at each 1D Convolutional layer is calculated by (3),

$$O = \frac{I - K + 2 * P}{\text{Strides} = 2} + 1 \qquad (3)$$

where,

O = Size of output feature map

I = Size of SA

K = Kernel size

P = Zero Padding,

Strides indicates kernel movement

A narrow convolution is applied between ET and Kernel K from the layer l_1 to l_2. The operation of 1D forward propagation from the layer l_1 to l_2 is represented as given in (4),

$$O_n^{(l_2)} = \sum_{m=1}^{N^{(l_1)}} 1d\_cnn(ET_m^{(l_1)}, wk_{mn}^{(l_1)}) + b_n^{(l_2)} \qquad (4)$$

where,

$ET_m^{(l_1)}$ indicate the input of mth neuron at layer (l_1)

$O_n^{(l_2)}$ indicate the output of nth neuron at layer (l_2)

1d_cnn indicate Convolutional Operation

$wk_{mn}^{(l_1)}$ indicate the weight of the kernel filter from mth neuron in layer (l_1) to the nth neuron in layer (l_2)

$b_n^{(l_2)}$ indicate the bias of nth neuron in layer (l_2)

$N^{(l_1)}$ indicate the kernel filters in layer (l_1).

Applying the ReLu activation function to the output of the Convolutional layer is given by (5),

$$R_n^{(l_2)} = \mathrm{ReLu}(O_n^{(l_2)}) \tag{5}$$

Where,

$R_n^{(l_2)}$ is the intermediate output of (l_2) before applying MaxPooling.

This layer is again followed by the 1st MaxPooling1D layer. The MaxPooled output of the Convolutional layer (l_2) is given by (6),

$$MP_n^{(l_2)} = \mathrm{MaxPooling}\,(R_n^{(l_2)}) \tag{6}$$

Where,

$MP_n^{(l_2)}$ will be the input of next Convolutional layer l_2.

Similarly, this operation of 1D forward propagation is repeated for all Convolutional layers (l_i, $2 \le i \le p$) present in 1D ConvNet feature extractor. Then pooled semantic features are given to the fully connected layer, which flattens the learned features into one-dimensional array and constructs a single lengthy feature map $\{f_1, f_2, \ldots\ldots, f_m\}$ and is given by NF as in (7).

$$NF = \{f_1, f_2, \ldots\ldots, f_m\} = \text{ flatten } (MP^{(l_i)}) \tag{7}$$

Now, the Neurally processed Features (NF) is ready for training the Neurally Augmented Machine Learning classifiers in phase-2. The entire process of Neurally processed feature construction is presented in algorithm 2.

Algorithm 2: Neurally processed Feature algorithm

Input: Pre-Processed dataset (PD), Glove Pre-trained Word Embedding Dictionary (GD)
Output: Neurally processed Features (NF)

```
// Tweet Embedding
$\mathbb{R}^q$ 🡐 q-dimensional vector word space
ET 🡐 $\phi$
for each $PT_i \in PD$
   for each $w_i \in PT_i$
      for each $Dw_i \in GD$
```

```
            if ($w_i$ = = $Dw_i$) then
               $w_i \leftarrow Dw_i$
            end if
      end for
      end for
   end for
   ET ← ET U $w_i$
   Sem_rep (ET)

   // Function for Semantic Representation of Embedded Tweets
   Sem_rep (ET):
            for each $w_{i,f_1^q} \in$ ET
                  for each $w_{j=i+1,\ f_1^q} \in$ ET
                        SV = simf ($w_{i,f_1^q}$, $w_{j=i+1,\ f_1^q}$)
                             if SV = 0 then
                                    ET ← ET \ $w_{i,f_1^q}$
                               end if
                  end for
              end for

   // Function for similarity calculation
    simf $\left(w_{i,f_1^q}, w_{j=i+1,f_1^q}\right)$:
              val  =  $\frac{w_{i,f_1^q}, w_{j=i+1,f_1^q}}{\sqrt{w_{i,f_1^q},w_{i,f_1^q}}\sqrt{w_{j=i+1,f_1^q},w_{j=i+1,f_1^q}}}$
   return val

   // Function for extracting Neurally processed features
    1D-CNN (ET):
      NF ← ϕ
              $ET^{(l_1)}$ ← input to Convolutional layer ($l_1$)
            for each $l_i$ , ($2 \le i \le p$)
                        $O_n^{(l_i)} \leftarrow \sum_{m=1}^{N^{(l_{i-1})}} 1d\_cnn(ET_m^{(l_{i-1})}, wk_{mn}^{(l_{i-1})}) + b_n^{(l_i)}$
                         $R_n^{(l_i)} \leftarrow$ ReLu ($O_n^{(l_i)}$)
                        $MP_n^{(l_i)} \leftarrow$ MaxPooling ($R_n^{(l_i)}$)
                        $f_i \leftarrow$ flatten $\left(MP_n^{(l_i)}\right)$
               end for
     NF ← NF U $f_i$
     return NF
```

Phase – 2

3.6 Construction of Augmenter layer

Once the Neurally processed Features (NF) are extracted in phase - 1, a novel Augmenter layer is constructed in such a way that 1D ConvNet feature extractor augments the Machine Learning classification layers by stacking process. The extracted features are used to train the Neurally Augmented Machine Learning classification layers.

3.7 Neurally Augmented Machine Learning classification layers

For classification purpose, the existing Machine Learning classifiers namely, Naïve Bayes, Random Forest, Gradient Boosting, and XGBoost are modified to include the Neurally processed features, NF. Hence, this phase constructs Neurally Augmented (NA) Machine Learning classification layers such as, NA_Naive Bayes, NA_Random Forest, NA_Gradient Boosting, and NA_XGBoost below the augmenter layer, which do not overfit the training procedure. These classifiers will provide more accurate results as the model is rigorously trained with large input datasets. More the training, better the classifier accuracy, hence, 80% of the pre-processed tweets are given for training and 20% of pre-processed tweets are given for testing.

The pre-processed dataset, PD = {(PT_1, Y_1), (PT_2, Y_2),......,(PT_n, Y_n)} is split into the training dataset denoted by TD = {(PT_1, Y_1), (PT_2, Y_2),.....,(PT_c , Y_c)} where, Y ∊ {real, fake}, and the testing dataset denoted by SD = {PT_{c+1} ,PT_{c+2},.....,PT_n }. The training dataset is used for training the Neurally Augmented Machine Learning classifiers as follows:

Neurally Augmented Naive Bayes Classification

A NA_Naive Bayes Classification layer is constructed using Training Dataset TD to identify the best class for the test dataset SD. This layer performs COVID-19 tweet classification based on the principle of Bayes theorem, which depends upon conditional probabilities, prior probability, and evidence to identify posterior probability. The posterior probability of a particular tweet PT_i from SD belonging to a class Y_i based on Naive Bayes is computed as in (8),

$$\mathbf{p(Y_i|PT_i) = \frac{p(PT_i|Y_i) * p(Y_i)}{p(PT_i)}} \tag{8}$$

where,

Y_i denotes the class labels, i ∊ {real, fake}

$p(Y_i)$ denotes the prior probability

$p(PT_i)$ denotes the evidence

Likelihood $p(PT_i \mid Y_i)$ denotes the conditional probability

$p(Y_i \mid PT_i)$ denotes the posterior probability

Each PT_i is represented as Neurally processed features NF = {f_1, $f_{2,....,}$ f_m }. In general, Naive Bayes considers that the features are independent of each other. Hence, the NA_Naive Bayes classification layer assumes the Neurally processed features $p(f_1, f_{2,....,} f_m \mid Y_i)$ are independent to each other given the class Y_i. Likelihood or Conditional probabilities are estimated directly from TD and is computed as,

$$p(f_1, f_2, \ldots., f_m|Y_{i=real}) = p(f_1|Y_{i=real}) \,.\, p(f_2|Y_{i=real}) \ldots \ldots.. p(f_m|Y_{i=real}) = \prod_{k=1}^{m} p(f_k\,|Y_{i=real})$$

$$p(f_1, f_2, \ldots., f_m|Y_{i=fake}) = p(f_1|Y_{i=fake}) \,.\, p(f_2|Y_{i=fake}) \ldots.. p(f_m|Y_{i=fake}) = \prod_{k=1}^{m} p(f_k\,|Y_{i=fake})$$

Prior probability $p(Y_i)$ denotes the probability of a feature occuring in a particular class and is given by (9) and (10),

(9)

$$p(Y_{i=real}) = \frac{Y_{i=real}}{Y_{i=real,\ fake}}$$

$$p(Y_{i=fake}) = \frac{Y_{i=fake}}{Y_{i=real,\ fake}} \quad (10)$$

$p(f_1, f_2, \ldots, f_m)$ denotes the evidence that a particular feature is independent from the class label.

To find the accurate class for the SD, the decision rule of NA_Naive Bayes classification layer is made to consider the maximum a posteriori or most likely class and is computed as (11),

$$Y' = \text{argmax}_{i\,\in\{real,fake\}}\ p(Y_i) \prod_{k=1}^{m} p(\,f_k \mid Y_i) \quad (11)$$

For every new tweet in SD, the probability of such tweet is computed and the final predicted class (fake or real) is decided based on the maximum a posterior or maximum probability.

The entire NA_Random Forest classification layer process is presented in algorithm 3.

Algorithm 3: Neurally Augmented Naive Bayes Classification

Input: Training dataset, TD:= {(PT$_1$, Y$_1$), (PT$_2$, Y$_2$),.....,(PT$_c$, Y$_c$)}, Test dataset, SD:= {PT$_{c+1}$, PT$_{c+2}$,.....,PT$_n$ }, Neurally processed Features, NF:= { f$_1$, f$_{2,....,}$ f$_m$}

Output: class labels of SD:- Y $'$∊ Y$_i$

```
NA_NaiveBayes (TD, SD, NF):
        $p(TD_{fake}) = \frac{(TD, Y_{i\in fake})}{|TD|}$
        $p(TD_{real}) = \frac{(TD, Y_{i\in real})}{|TD|}$
        for each $f_{i\varepsilon real}$ in NF
            conditionalprob ($f_{i\varepsilon real}$) ← cp($f_{i\varepsilon real}$ | $Y_{i\varepsilon real}$)
        end for
        for each $f_{i\varepsilon fake}$ in NF
            conditionalprob ($f_{i\varepsilon fake}$) ← cp($f_{ii\varepsilon fake}$ | $Y_{i\varepsilon fake}$)
        end for


// Classification
        Posteriorprob ($Y_{i\varepsilon real}$ | $f_{i\varepsilon real}$) ← conditionalprob ($f_{i\varepsilon real}$) * p($TD_{real}$)
        Posteriorprob ($Y_{i\varepsilon fake}$ | $f_{i\varepsilon fake}$) ← conditionalprob ($f_{i\varepsilon fake}$) * p($TD_{fake}$)
        mx← max (Posteriorprob ($Y_{i\varepsilon real}$ | $f_{i\varepsilon real}$), Posteriorprob ($Y_{i\varepsilon fake}$ | $f_{i\varepsilon fake}$))
                i ← posteriorprob($Y_i$ | $f_i$) (mx))
                Y’← i
```

```
    return Y'
```

Neurally Augmented Random Forest Classification

A NA_Random Forest Classification layer is built using Training Dataset TD to identify the best class for the test dataset SD. In general, Random Forest constructs n number of trees for performing classification in order to improve the overall accuracy. Hence, NA_Random Forest builds n trees that work on the principle of ensemble learning by combining predictions from n trees to improve the classification performance and to make the classification model more robust. Building n trees require bootstrap datasets BD_i ($1 \leq i \leq n$), which are subsets from TD and Neurally processed Features NF. Bootstrap datasets are randomly chosen subsets using random sampling with replacement technique and each subset trains each tree in order to get predictions of each subset.

To build a tree, a subset of features NF from BD_i is selected randomly. Deciding on the maximum number of features to split a node in a tree takes more computation; however, selecting the subset of features will speed up the process of tree learning. Hence, among NF features, a random feature sf is designated as the root node 'r'. Next, the root node r is split into m children nodes using the random split feature from remaining sf-1 features. The procedure is repeated for constructing the remaining n-1 trees. After training, the test dataset is passed to all the newly built trees to predict the classification results of each tree. Finally, all the classification results are subject to majority voting to determine the maximum possibility of the class for SD. The entire process of NA_Random Forest classification layer is explained in algorithm 4.

Algorithm 4: Neurally Augmented Random Forest Classification

Input: Training dataset, TD:= $\{(PT_1, Y_1), (PT_2, Y_2),.....,(PT_c, Y_c)\}$, Test dataset, SD:= $\{PT_{c+1}, PT_{c+2},.....,PT_n\}$, Neurally processed Features (NF), n trees in Forest V

Output: class labels of SD:- $Y' \in Y_i$

```
Buildtree (TD, NF):
    V ← Φ
        for i = 1 to n do
            $BD_i$ ← Random Sampling with replacement (TD)
            $V_i$ ← NA_RandomForestTreeLearn ($BD_i$, NF)
            V ← V U $V_i$
        end for

NA_RandomForestTreeLearn ($BD_i$, NF):
for each $BD_i$
```

```
    select sf $\subseteq$ NF
        r $\leftarrow$ random(sf)
    split r into sub nodes from (sf-1)
end

// Classification
classification(V, SD):
for each $V_i \in V$
    Traverse $PT_i$ over $V_i$
        $C_i \leftarrow$ predictclass
end for
Y' = majorityvoting {$C_1$ ($PT_{c+1}$) , $C_2$ ($PT_{c+2}$),....., $C_n$ ($PT_n$ )}
return Y'
```

Neurally Augmented Gradient Boost Classification

A Neurally Augmented Gradient Boosting Classification layer is built using Training Dataset TD to identify the best class for the test dataset SD based on Neurally processed Features NF. NA_Gradient Boosting is based on the principle of ensemble learning where n weak learner trees are created sequentially which reduces the errors made by the previous trees. The proposed classification layer aims to minimize the loss function (i.e. errors); hence, the weak learners are combined to form a strong learner, which improves the accuracy of classification. NA_Gradient Boosting initializes the model with a constant value using log (odds), which is given by (12),

$$M_0(x) = \text{argmin}\gamma \sum_{i=1}^{n} L(Y_i \ ,\gamma) \tag{12}$$

where,

L – Loss function

Y_i - Observed classification labels of TD, Y ∊ {real = 1, fake = 0},

γ - Predicted probability obtained using log (odds)

argmin indicates determining the log (odds) value at which the loss function is minimal. Hence, this minimal loss function is the initial prediction for base model $M_0(x)$. Initial prediction probability γ is calculated as in (13) and (14),

$$log(odds) = \log_e (pr / (1\text{-}pr)) \tag{13}$$

gg

(14)

$$\gamma = \frac{1}{1 + e^{-(\log_e (pr / (1-pr)))}}$$

Loss function is used to identify residuals. Once, γ is calculated, residual r is computed for every PT_i in TD as in (15),

$$r_i = Y_i - \gamma \tag{15}$$

For each model $M_{mo}(x)$ where, $1 \leq mo \leq N$, compute residuals, construct the trees, calculate probabilities based on NF and residuals. Now these residual values are used to construct tree t_1 by branching NF instead of Y_i labels. Then the weight of each leaf node w_i is computed as in (16),

$$w_i = \frac{\sum r_i}{\sum[\gamma * (1 - \gamma)]} \tag{16}$$

This NA_Gradient Boosting classification layer generates learners, which minimizes the loss function of the present model. Now every PT_i in TD, compute new prediction probability value γ_i as given in (17) and (18),

$$g = r_i + learning\_rate * w_i \tag{17}$$

$$\gamma_i = \frac{1}{1 + e^{-(g)}} \tag{18}$$

Learning rate can be any value between 0 and 1, hence, 0.2 is chosen in this work, which scales the new tree contribution resulting in accurate direction of prediction. Once, γ_i of t_i is calculated, residual of γ_i is computed and next t_2 is constructed, again w_i, γ_i of t_2 is computed. The same process is repeated until overall residuals get minimized and becomes closer to the actual value. $M_0(x)$ along with predictions on (NF, residuals ($M_0(x)$)) and residuals of first model together constitutes $M_1(x)$. Similarly, $M_1(x)$ along with predictions on (NF, residuals ($M_1(x)$)) and residuals of ($M_2(x)$) together constitutes $M_2(x)$. Finally, $M_{mo-1}(x)$ along with predictions on (NF, residuals ($M_{mo-1}(x)$)) and residuals of ($M_{mo}(x)$) together constitutes $M_{mo}(x)$ and is given by (19).

$$M_{mo}(x) = M_{mo-1}(x) + learning\_rate * K(NF, r_i(M_{mo}(x)) \tag{19}$$

To classify the tweet in SD, the prediction p is computed using (20),

$$p = \gamma + learning\_rate * w_i(t_1) + learning\_rate * w_i(t_2) +.....+ learning\_rate * w_i(t_n)$$

Then from p, prediction probability pp is calculated by,

$$pp = \frac{1}{1+e^{-p}} \tag{20}$$

If pp is greater than 0.5, then assign the predicted class Y ' of tweet as real, otherwise the predicted class Y' is fake. The entire process of NA_Gradient Boosting classification layer is explained in algorithm 5.

Algorithm 5: Neurally Augmented Gradient Boosting Classification

Input: Training dataset, TD:= {(PT_1, Y_1), (PT_2, Y_2),.....,(PT_c , Y_c)}, Test dataset, SD:= {PT_{c+1} , PT_{c+2},.....,PT_n }, Neurally processed Features (NF)
Output: class labels of SD:- Y ' ∈ Y_i

```
Initialize model, $M_0(x) = \text{argmin}\gamma \sum_{i=1}^{n} L(Y_i\ ,\gamma)$
 //Construction of learning models
for each model, mo = 1 to N:
$M_{mo}(x) \leftarrow \Phi$
       // Calculation of residuals
     for every tweet $PT_i$ in TD do
          calculate r
                    $r_i(M_{mo}(x)) \leftarrow Y_i - \gamma$
        // Construction of Trees
        fit a tree $t_{mo}$ by branching NF with r and create leaf nodes
                 $le_j(M_{mo}(x)) \leftarrow$ leaf nodes
       // Calculation of weight for each leaf node
        for each $le_j(M_{mo}(x))$ j $\epsilon$1 to k
                    compute weight $w_i(le_j(M_{mo}(x)))$
                    $w_i\ (le_j(M_{mo}(x))) = \frac{\sum r_i}{\sum \text{ of each } \gamma(1-\gamma) \text{for each } le_j\ (M_{mo}(x))}$
            end for
      // Prediction Probability value for each NF in TD
                     $g = r_i(M_{mo}(x)) + learning\_rate * w_i\ (le_j(M_{mo}(x)))$
                    $\gamma_i(M_{mo}(x)) = \frac{1}{1+\ e^{-(g)}}$
         $M_{mo}(x) \leftarrow M_{mo}(x) \cup \gamma_i(M_{mo}(x))$
  end for
end

// Classification
        for tweet $PT_i$ in SD
                      $p = \gamma + learning\_rate * w_i\ (le_j(M_1(x))) + learning\_rate * w_i\ (le_j(M_2(x))) +.....+ learning\_rate * w_i\ (le_j(M_{mo}(x)))$
                      $pp = \frac{1}{(1+e^{-p})}$
               if pp $\leq$ 0.5 then
                     Y ' $\epsilon$ real
               else
                     Y ' $\epsilon$ fake
      end for
end
```

Neurally Augmented XGBoost Classification

A Neurally Augmented XGBoost Classification layer is built on the extended version of NA_Gradient Boosting to minimize it's time computation. Neural-Augmented XGBoost Classification layer assumes a constant value as initial probability γ and computes the residuals r_i for all tweets in the TD as given by (21),

(21)

$$r_i = Y_i - \gamma$$

Next, tree is constructed based on the binary splits using the computed residuals. Then similarity weight is computed for each split with root node, left subtree and right subtree as in (22).

$$sim\_w_i = \frac{\sum r_i}{\sum[\gamma * (1 - \gamma)] + \lambda} \tag{22}$$

where,

λ is a regularization constant used to prevent over fitting

Based on the similarity weight, gain is computed for each split. The split with maximum gain is chosen and the tree is grown to maximum levels and is computed by (23),

$$gain = sim\_wt(leftsubtree) + sim\_wt(rightsubtree) - sim\_wt(rootnode) \tag{23}$$

Then a new probability is calculated as in (24) and (25),

$$g = log\ (odds) + learning\_rate * sim\_wt(node)(PT_i) \tag{24}$$

$$new\_\gamma_i = \frac{1}{1 + e^{-(g)}} \tag{25}$$

Again residuals are computed for each tweet in TD. This process is repeated until residuals gets minimum.

After the split, whether we should do the splitting or not is decided by pruning. Pruning is basically executed in XGBoost by a cover value, $\gamma(1 - \gamma)$, if the gain is less than the cover value, the branch is cut.

The entire process of NA_XGBoost classification layer is explained in algorithm 6.

Algorithm 6: Neurally Augmented XGBoost Classification

Input: Training dataset, TD:= {(PT$_1$, Y$_1$), (PT$_2$, Y$_2$),.....,(PT$_c$, Y$_c$)}, Test dataset, SD:= {PT$_{c+1}$, PT$_{c+2}$,.....,PT$_n$ }, Neurally processed Features (NF)
Output: class labels of SD:- $Y' \in Y_i$

```
// Calculation of residuals
  for every tweet PT$_i$ in TD do
     calculate r, residual
        $r_i \leftarrow Y_i - \gamma$
        $\gamma \leftarrow$ initialProbability
  end for

Treeconstruction(NF, r$_i$):
begin
```

```
  // Construction of Trees
  for each tree $t_n$
    split a binary tree $t_n$ by branching NF into node $n_j$ with $r_i$
        where, $j \in$ {rootnode, leftsubtree, rightsubtree}
    // Calculation of similarity weight for each node
       for each $n_j$ in $t_n$ calculate
            $sim\_wt\ (n_j) = \frac{\sum r_i}{\sum \gamma(1-\gamma)+\lambda}$
                $\lambda \leftarrow$ regularization constant
       end for
// Calculate gain for each split($s_i$)
          gain($s_i$)$\leftarrow$ sim_wt (leftsubtree) + sim_wt (rightsubtree) - sim_wt (rootnode)
          ($s_i$) with max(gain) is grown to levels
end

// Calculation of new probability
for every tweet $PT_i$ in TD do
       $log\ (odds) \leftarrow \log_e (\frac{\gamma}{1-\gamma})$
       g $\leftarrow$ log(odds) + learning_rate * sim_wt ($n_j$) from grown tree according to NF in ($PT_i$)
       $new\_\gamma_i \leftarrow \frac{1}{1+ e^{-(g)}}$
       // calculate r
       new_$r_i \leftarrow Y_i - new\_\gamma_i$
Treeconstruction(NF, new_$r_i$ )

// Classification
       for tweet $PT_i$ in SD
            $p = \gamma +$ learning_rate $*$ $sim\_wt(t_1)$ + learning_rate $*$ sim_wt($t_2$) +.....+
                      learning_rate $*$ $sim\_wt(t_n)$
            $pp = \frac{1}{(1+e^{-p})}$
            if pp $\leq$ 0.5 then
                 $Y' \epsilon$ real
            else
                 $Y' \epsilon$ fake
       end for
end
```

4 Experimental Setup & Result Analysis

The experimental setup and evaluation metrics of the proposed model are presented in sections 4.1 and 4.2. The result analysis of Neurally Augmented Model, which is done by comparing it with other classifiers based on Machine Learning and Deep Learning algorithms, is discussed in Section 4.3.

4.1 Experimental set-up

The Experiment is performed in system settings with core resources of Intel(R) Core(TM) i5 Processor, 8GB RAM, 500GB Hard Disk and Graphics Processing Units (GPU) environment. The experimentations and evaluations of the proposed model are done using

Python 3.7 along with the Scikit-learn library, Tensor flow, Keras, Numpy and Pandas package. The Deep Learning-based feature extractor is implemented using the Tensor Flow sequential API. The newly created COVID-19 Twitter dataset is given as input to assess the proposed network's performance.

4.2 Evaluation metrics

The performance measures for classifying fake news in the COVID-19 dataset are manifested in this section. The efficiency of the proposed model is validated using evaluation metrics, such as, True Positive (TP), False Negative (FN), False Positive (FP), True Negative (TN), Precision, Recall, and F1 Score to measure Accuracy as well as Area Under the Receiver Operating Characteristic (AUROC) Score.

True Positive – The values that are predicted correctly as actual positives.

False Negative – Negative samples incorrectly predicted as actual positives

False Positive – Positive samples incorrectly predicted as actual negatives

True Negative – The samples that are identified correctly as actual negatives

Precision metric exhibits the accuracy of the positive class and measures whether the prediction of the positive class is correct as defined in (26) and is given by,

$$\text{Precision} = TP / (TP + FP) \tag{26}$$

Recall is measured as the fraction of positive classes correctly detected to the total classes as defined in (27) and is given by,

$$\text{Recall} = TP / (TP + FN) \tag{27}$$

F1 Score is the weighted average score or harmonic mean of true positive (recall) and precision as defined in (28) and is given by,

$$\text{F1 Score} = 2 * [(P * R) / (P + R)] \tag{28}$$

Accuracy is measured as the fraction of total of True Positive, True Negative to the total of True Positive, False Positive, True Negative, and False Negative as defined in (29) and is expressed as,

$$\text{Accuracy} = [(TP + TN)/ (TP + TN + FP + FN)] \tag{29}$$

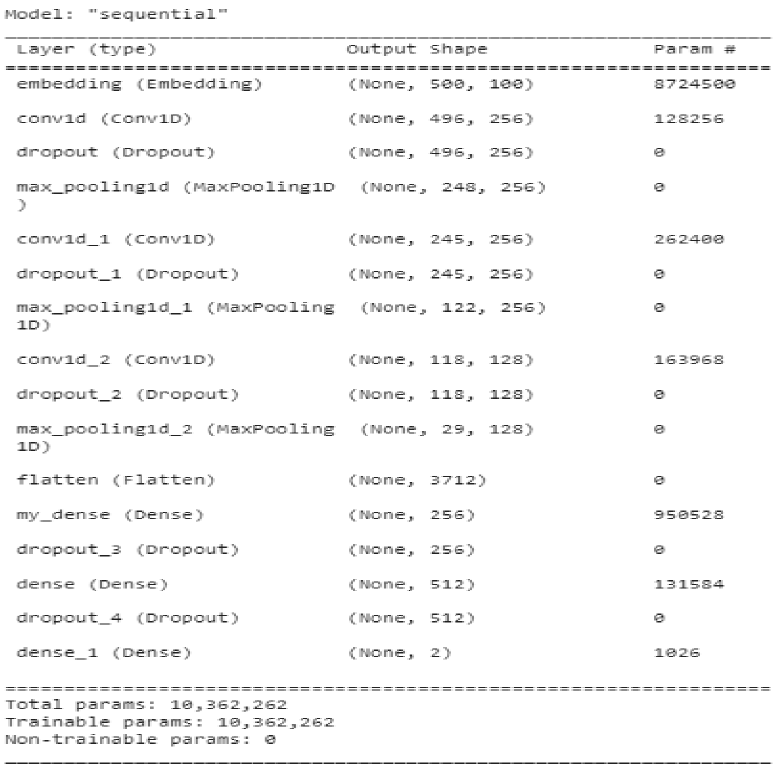
The ROC curve is used to visualize the performance of NAM. It is referred to as a probability curve. True positive rate gets plotted on the y-axis against the false positive rate on the x-axis. The Area Under the Curve (AUC) is a critical parameter for assessing the classification accuracy of the model. The larger the AUC measurement, the model can better be able to discern between authentic and false news. AUROC Score is the measure of predicting power of the model to differentiate the classes.

4.3 Result analysis

Phase-1

This section discusses the results obtained when tested on the created COVID – 19 Twitter dataset. The model summary of the proposed 1D ConvNet feature extractor is given in

Figure 6.

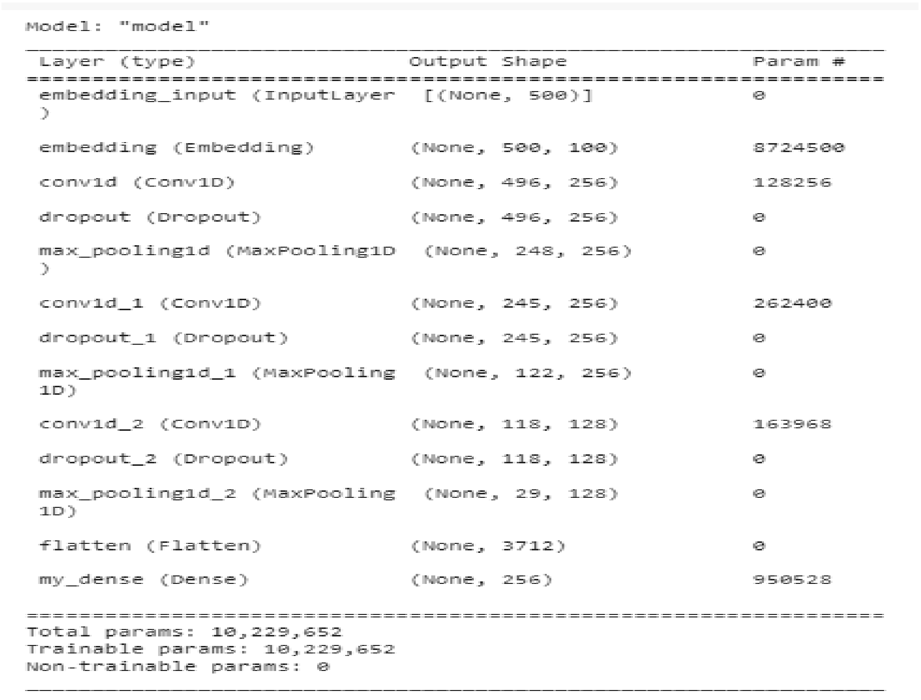


```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 500, 100)          8724500

 conv1d (Conv1D)             (None, 496, 256)          128256

 dropout (Dropout)           (None, 496, 256)          0

 max_pooling1d (MaxPooling1D  (None, 248, 256)         0
 )

 conv1d_1 (Conv1D)           (None, 245, 256)          262400

 dropout_1 (Dropout)         (None, 245, 256)          0

 max_pooling1d_1 (MaxPooling  (None, 122, 256)         0
 1D)

 conv1d_2 (Conv1D)           (None, 118, 128)          163968

 dropout_2 (Dropout)         (None, 118, 128)          0

 max_pooling1d_2 (MaxPooling  (None, 29, 128)          0
 1D)

 flatten (Flatten)           (None, 3712)              0

 my_dense (Dense)            (None, 256)               950528

 dropout_3 (Dropout)         (None, 256)               0

 dense (Dense)               (None, 512)               131584

 dropout_4 (Dropout)         (None, 512)               0

 dense_1 (Dense)             (None, 2)                 1026

=================================================================
Total params: 10,362,262
Trainable params: 10,362,262
Non-trainable params: 0
_________________________________________________________________
```

Figure 6: Model Summary of 1D ConvNet feature extractor.

Phase – 2

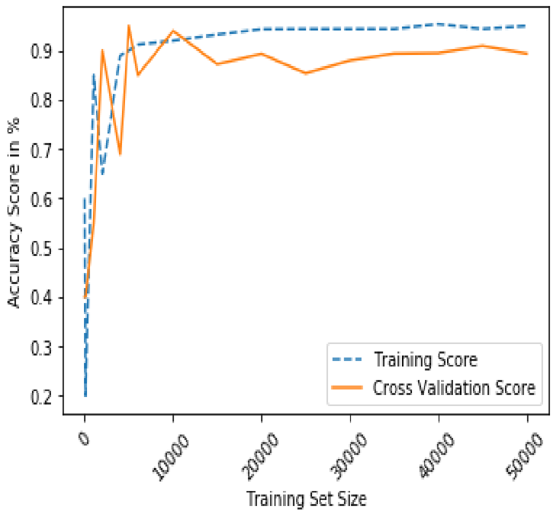
The Augmenter layer extracts Neurally processed features from the fully connected layer of 1D ConvNet feature extractor and use it to augment the Machine Learning classification layers, NA_Naive Bayes (NA_NB), NA_Random Forest (NA_RF), NA_Gradient Boosting (NA_GB) and NA_XGBoost (NA_XGB) for classification. The model summary of Augmenter layer is depicted in Figure 7.



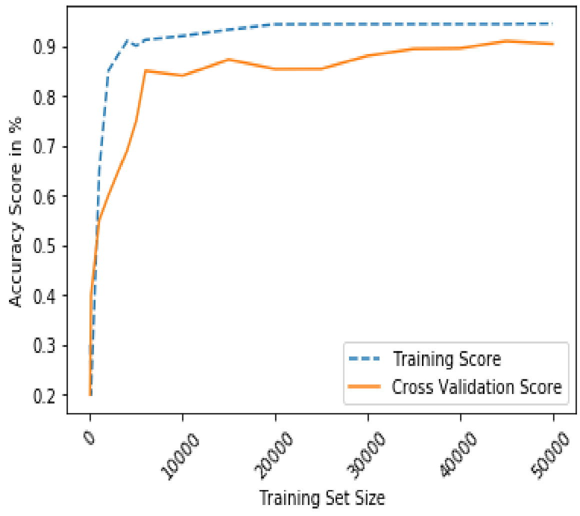
```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_input (InputLayer  [(None, 500)]            0
 )

 embedding (Embedding)       (None, 500, 100)          8724500

 conv1d (Conv1D)             (None, 496, 256)          128256

 dropout (Dropout)           (None, 496, 256)          0

 max_pooling1d (MaxPooling1D  (None, 248, 256)         0
 )

 conv1d_1 (Conv1D)           (None, 245, 256)          262400

 dropout_1 (Dropout)         (None, 245, 256)          0

 max_pooling1d_1 (MaxPooling  (None, 122, 256)         0
 1D)

 conv1d_2 (Conv1D)           (None, 118, 128)          163968

 dropout_2 (Dropout)         (None, 118, 128)          0

 max_pooling1d_2 (MaxPooling  (None, 29, 128)          0
 1D)

 flatten (Flatten)           (None, 3712)              0

 my_dense (Dense)            (None, 256)               950528

=================================================================
Total params: 10,229,652
Trainable params: 10,229,652
Non-trainable params: 0
_________________________________________________________________
```

Figure 7: Model Summary of the Augmenter Layer.

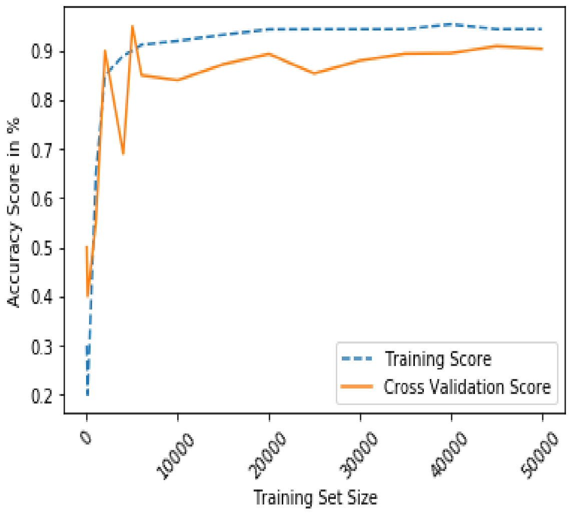
Learning curve is plotted to visualize the error in the predictions made by the Neurally Augmented Machine Learning classification layers changes, when the training set size increases or decreases. The Learning curve plots for NA_NB, NA_RF, NA_GB and NA_XGB is depicted in Figure 8(a), 8(b), 8(c) and 8(d) respectively.



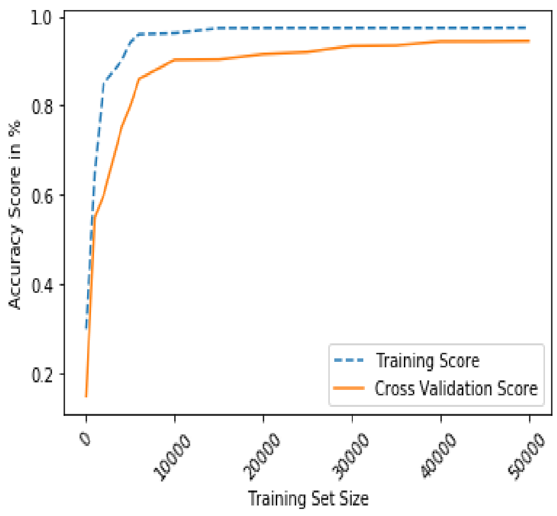
(a) Learning curve of NA_Naive Bayes



(b) Learning curve of NA_Random Forest



(c) Learning curve of NA_Gradient Boosting



(d) Learning curve of NA_XGBoost

Figure 8: Learning curves of Neurally Augmented Machine Learning classification layers

In Figure 8, Accuracy Score is plotted in X-axis and Training set size is plotted in Y-axis. From Figure 8(a), it is observed that the training score initially increases and decreases at the starting point and the prediction errors gets reduced and a constant training accuracy score is maintained. Cross validation score increases above the training score initially, again it decreases and a constant accuracy score is achieved as the training data gets added. From Figure 8(b), it is observed that the training score is accurate as it gradually increases when the training set size increases. Cross validation score decreases at a point after training set size 20000 and again increases slowly after 30000. From Figure 8(c), it is observed that the training score is accurate as the dada gets added. Cross validation score increases and decreases initially and when the data gets added, the accuracy improves. From Figure 8(d), it is observed that both the training score and cross validation score improves by adding the data. It is observed that the training score and cross validation score converge with the increase in training set size in all the cases of Neurally Augmented Machine Learning classification layers. The prediction errors made by each classification layer get reduced when training data is added resulting in improved accuracy.

The performance comparison of the proposed Neurally Augmented Model (NAM) in terms of accuracy, AUROC Score, precision, recall and F1 Score are depicted in Figure 9.

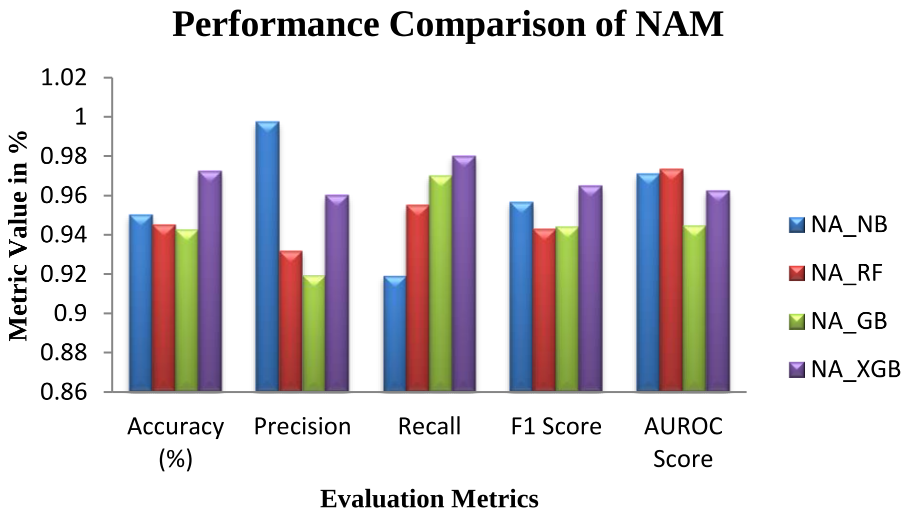
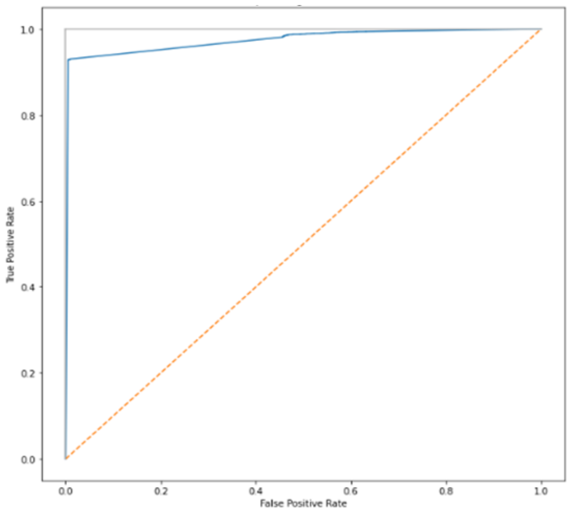
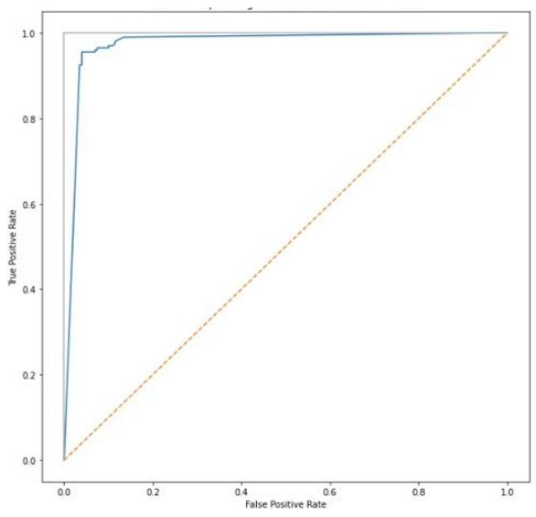


Figure 9: Performance comparison of Neurally Augmented Models.

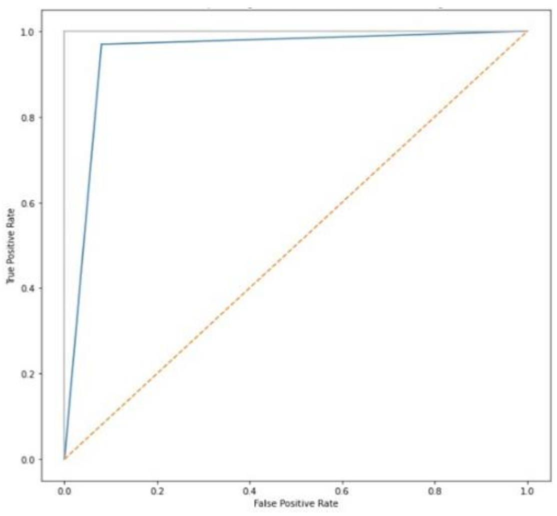
From Figure 5, it is found that NA_NB obtains 95.01% accuracy, 0.9975 precision, 0.9191 recall, 0.9567 F1 Score and 0.9711 AUROC Score. NA_RF achieves 94.5 % accuracy, 0.9317 precision, 0.955 recall, 0.943 F1 Score and 0.9733 AUROC Score. NA_GB obtains 94.25% accuracy, 0.9194 precision, 0.97 recall, 0.944 F1 Score and 0.9447 AUROC Score. NA_XGB achieves 97.25 percentage accuracy, 0.96 precision, 0.98 recall, 0.965 F1 Score and 0.9624 AUROC Score. Among all the NA_Machine Learning classification layers, NA_XGB performs well with the highest accuracy, recall and F1 Score. While considering AUROC Score, NA_RF performs well with the highest AUROC Score. The ROC curve of NA_NB, NA_RF, NA_GB and NA_XGB classification layers are depicted in Figure 10 (a), 10 (b), 10 (c) and 10 (d), respectively.



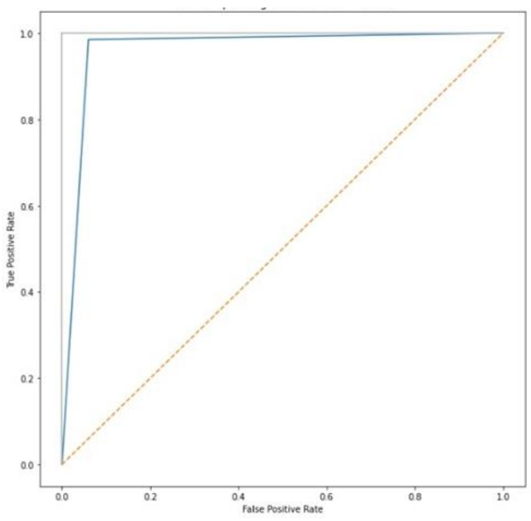
(a) ROC curve of NA_Naive Bayes



(b) ROC curve of NA_Random Forest



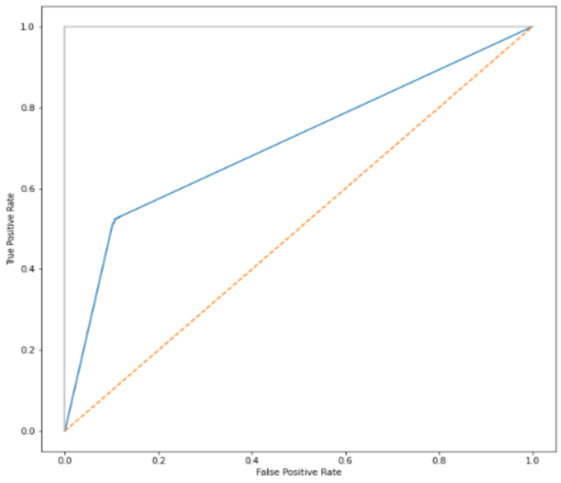
(c) ROC curve of NA_Gradient Boosting



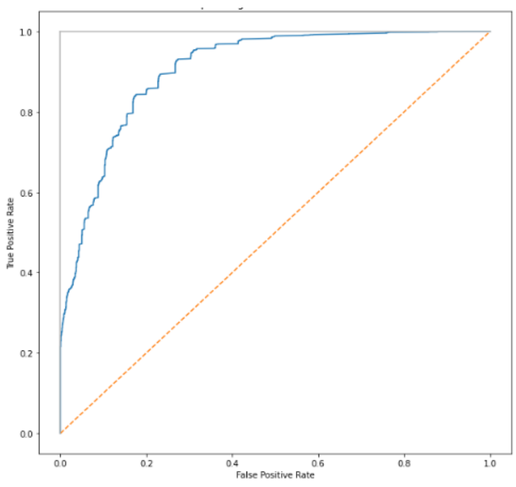
(d) ROC curve of NA_XGBoost

Figure 10: ROC curves of Neurally Augmented Machine Learning classification layers

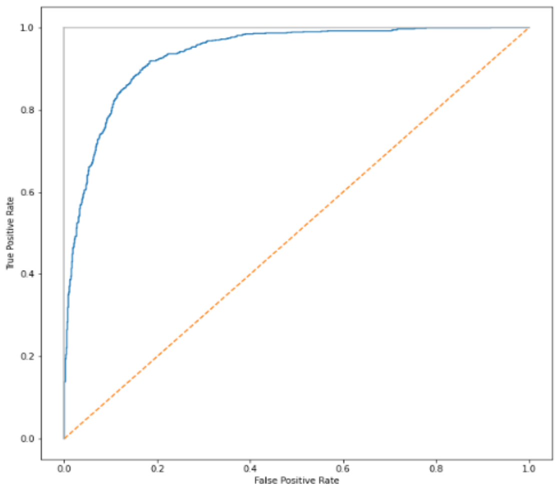
From Figure 10 (a), it is observed that area under the curve of False Positive Rate range is higher resulting in 0.9711 AUROC Score. From Figure 10 (b), it is observed that area under the curve of False Positive Rate range is higher resulting in 0.9733 AUROC Score. From Figure 10 (c), it is observed that area under the curve of False Positive Rate range is higher resulting in 0.9447 AUROC Score. From Figure 10 (d), it is observed that area under the curve of False Positive Rate range is higher resulting in 0.9624 AUROC Score. The ROC curve of Neurally Augmented Machine Learning classification layers is compared with the non-neurally augmented Machine Learning classifiers. The ROC curve of non-neurally augmented Machine Learning classifiers namely, Naive Bayes, Random Forest, Gradient Boosting and XGBoost are depicted in Figure 11 (a), 11 (b), 11 (c) and 11 (d), respectively.



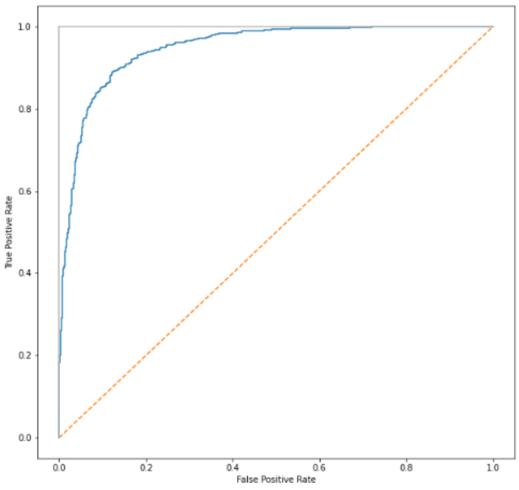
(a) ROC curve of NA_Naive Bayes



(b) ROC curve of NA_Random Forest



(c) ROC curve of NA_Gradient Boosting



(d) ROC curve of NA_XGBoost

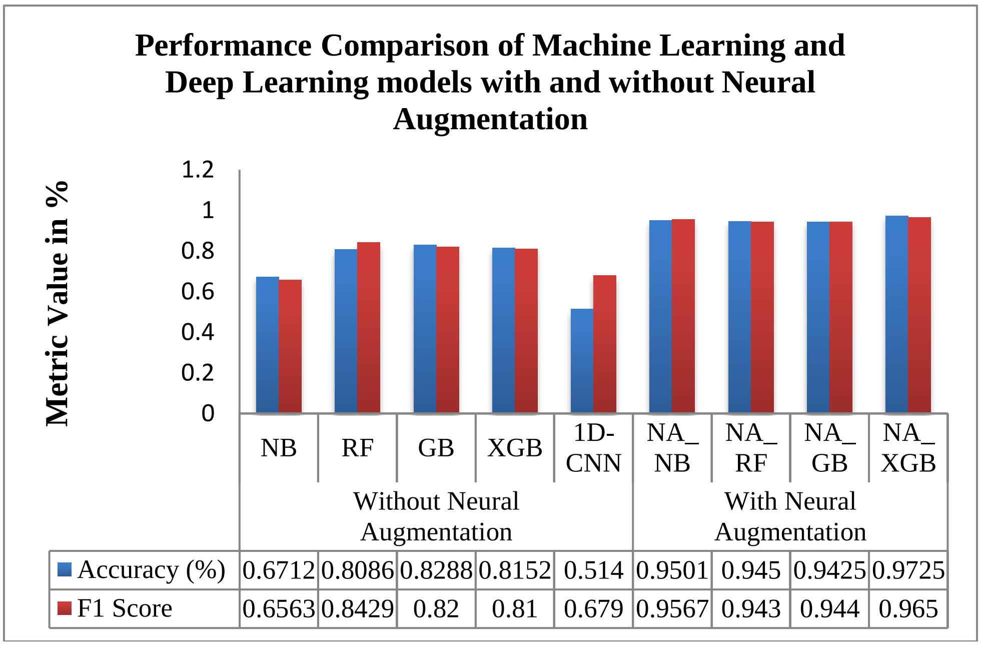
Figure 11: ROC curves of non-neurally Augmented Machine Learning classifiers

From Figure 11 (a), it is observed that area under the curve of False Positive Rate range is lower resulting in 0.7754 AUROC Score. From Figure 11 (b), it is observed that area under the curve of False Positive Rate range is higher resulting in 0.9057 AUROC Score. From Figure 11 (c), it is observed that area under the curve of False Positive Rate range is higher resulting in 0.9147 AUROC Score. From Figure 11 (d), it is observed that area under the curve of False Positive Rate range is higher resulting in 0.9279 AUROC Score. Neurally Augmented Naive Bayes shows 20% increased performance over non-neurally Augmented Naive Bayes. Neurally Augmented Random Forest shows 7% increased performance over non-neurally Augmented Random Forest. Neurally Augmented Gradient Boosting shows 3% increased performance over non-neurally Augmented Gradient Boosting. Neurally Augmented XGBoost shows 4% increased performance over non-neurally Augmented XGBoost. Considering the false positive rate range, Neurally Augmented Machine Learning classifier layers classifies the real and fake news better than the non-neurally augmented classifiers.

5 Discussion

In online social networks, the most challenging task is the spread of fake news. Posting fake news regarding COVID-19 is still making the situation worse. The major factors influencing Machine Learning and, Deep Learning techniques are feature extraction and data. This work aims to design a Neurally Augmented Model for the accurate, automatic and timely detection of COVID-19 fake news on Twitter platforms. NAM is created with the capability of automatic feature extraction and to augment the Deep Neural Network with the classic Machine Learning. A Deep Neural Network feature extractor, 1D ConvNet, is used for automatic feature extraction. To deal with the limited datasets, COVID-19 twitter datasets are created with 50000 tweets and are given as input to NAM. The tweet dataset undergoes pre-processing to remove noise from the dataset. After pre-processing, the Semantic-Aware tweet vector representation of tweets is created using a pre-trained embedding, ‘glove.twitter.27B.100d.txt’. These vector representations form an embedding matrix, which is added as the first embedding layer of the feature extractor. This extractor extracts the Neurally processed significant features from the tweets. These features are extracted by the Augmenter layer from the fully connected layer of the feature extractor and use it to augment the Machine Learning classification layers for classification. NA_NB, NA_RF, NA_GB, and NA_XGB, are trained to learn the patterns and is made to classify the test tweets. The performance of NAM is measured in terms of accuracy, precision, recall, F1-Score and AUROC Score. NA_XGB classification layer obtains 97.25% accuracy, 0.96 precision, 0.98 recall, 0.965 F1 Score and 0.9624 AUROC Score to detect the fake news.

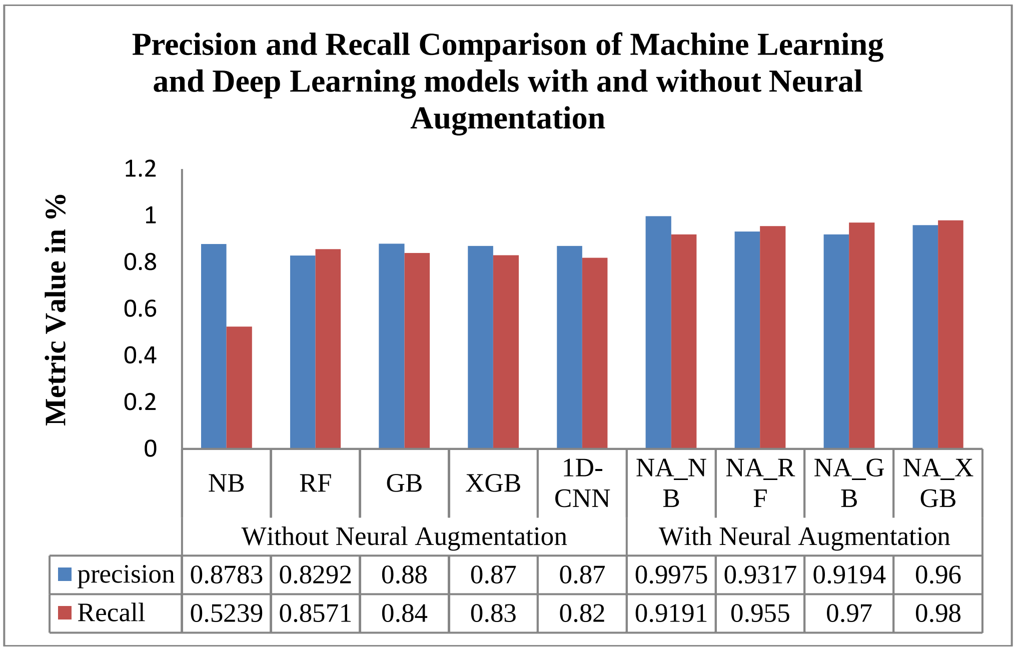
To further investigate the performance of the proposed model, experiments are done by comparing the proposed model with the Machine Learning and, Deep Learning models without neural augmentation in terms of Accuracy, F1 Score as depicted in Figure 12.



	NB	RF	GB	XGB	1D-CNN	NA_NB	NA_RF	NA_GB	NA_XGB
🟦 Accuracy (%)	0.6712	0.8086	0.8288	0.8152	0.514	0.9501	0.945	0.9425	0.9725
🟥 F1 Score	0.6563	0.8429	0.82	0.81	0.679	0.9567	0.943	0.944	0.965

Figure 12: Performance Comparison of Machine Learning and Deep Learning models with and without Neural Augmentation.

From Figure 12, it is found that the existing NB classifier without Neural Augmentation achieves 67.12% accuracy and 0.6563 F1 Score. RF achieves 80.86% accuracy and 0.8429 F1 Score. GB obtains 82.88% accuracy and 0.82 F1 Score. XGB achieves 81.52% accuracy and 0.81 F1 Score. Existing Deep Learning model 1D-CNN without Neural Augmentation obtains 51.4% accuracy and 0.679 F1 Score. In contrast, the proposed NA_NB obtains 95.01% accuracy and 0.9567 F1 Score. NA_RF obtains 94.5% accuracy and 0.943 F1 Score. NA_GB obtains 94.25% accuracy and 0.944 F1 Score. NA_XGB obtains 97.25% accuracy and 0.965 F1 Score. Machine Learning, and Deep Learning models with and without neural augmentation in terms of Precision and recall are depicted in Figure 13.



	Without Neural Augmentation					With Neural Augmentation			
	NB	RF	GB	XGB	1D-CNN	NA_NB	NA_RF	NA_GB	NA_XGB
🟦 precision	0.8783	0.8292	0.88	0.87	0.87	0.9975	0.9317	0.9194	0.96
🟥 Recall	0.5239	0.8571	0.84	0.83	0.82	0.9191	0.955	0.97	0.98

Figure 13: Precision and recall Comparison of Machine Learning, Deep Learning models with and without Neural Augmentation

From Figure 13, it is found that the existing Machine Learning model NB without Neural Augmentation achieves 0.8783 precision and 0.5239 recall. RF obtains 0.8292 precision and 0.8571 recall. GB obtains 0.88 precision and 0.84 recall. XGB achieves 0.87 precision and 0.83 recall. Deep Learning model 1D-CNN without Neural Augmentation obtains 0.87 precision and 0.82 recall. In contrast, NA_NB obtains 0.9975 precision and 0.9191 recall. NA_RF obtains 0.9317 precision and 0.955 recall. NA_GB obtains 0.9194 precision and 0.97 recall. NA_XGB obtains 0.96 precision and 0.98 recall.

Consequently, the proposed Neurally Augmented Model outperforms the Machine Learning and, Deep Learning models without Neural Augmentation by 12% and 10% in terms of accuracy and F1 Score.

5.1 Performance analysis of NAM on the benchmark datasets

The proposed model has experimented on the benchmark datasets like LIAR, IFND, ISOT and Constraint Shared Task ‘COVID-19 Fake News Dataset’ to further evaluate its robustness. The LIAR dataset is a freely accessible benchmark labelled dataset obtained from politifact.com with 1.4MB size. It consists of 12K short news samples with six labels false, true, mostly true, half true, pants-on-fire and barely true. Indian Fake News Dataset (IFND) is a publicly available large-scale Indian dataset that includes text of 4.77MB in size. It consists of 48K news samples with binary labels as fake or true. ISOT dataset is collected from reuters.com and kaggle.com with 86.6 MB size. It consists of 44K news samples with binary labels as fake or true. Each news sample has a length of more than 200 characters. COVID-19 Fake News Dataset in Constraint shared task ‘COVID 19 FN’ is collected from social media posts, press releases, news articles and tweets from government accounts with

2MB in size. It consists of 10K tweet samples with binary labels as real or fake. The experimentation results are shown in Figure 14.

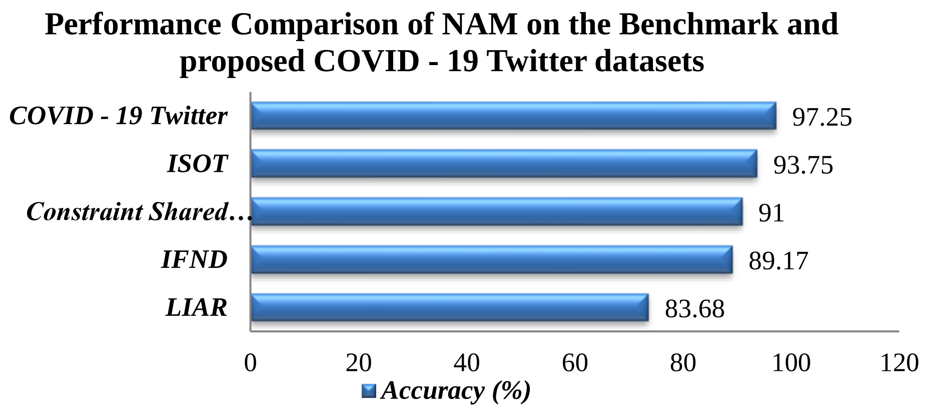


Figure 14: Performance Comparison of NAM model on the benchmark datasets and proposed COVID-19 Twitter dataset.

From Figure 14, it is revealed that the proposed model achieves 83.68%, 89.17%, 93.75%, 91% and 97.25% accuracy for the LIAR, IFND, ISOT, COVID 19 FN benchmark datasets and the proposed COVID-19 Twitter dataset, respectively. Consequently, the proposed model performs well for the COVID-19 Twitter dataset than all the benchmark datasets.

5.2 Performance analysis of the proposed model and existing works based on the benchmark datasets

The performance analysis of the proposed model on the four benchmark datasets is compared with the previous works on the benchmark datasets and is tabulated in Table-2.

Table 2: Performance comparison of the proposed model with existing works on ISOT, IFND, LIAR and COVID-19 FN benchmark datasets.

Ref	Techniques	Datasets				ISOT		IFND		LIAR		COVID 19 FN	
		ISOT	IFND	LIAR	COVID 19	Accuracy	F1 Score	Accuracy	F1 Score	Accuracy	F1 Score	Accurac	F1 Score
[43]	DT, SGD	✓	-	-	-	89	-	-	-	-	-	-	-
	SVM, LR	✓	-	-	-	84	-	-	-	-	-	-	-
	Linear SVM	✓	-	-	-	92	-	-	-	-	-	-	-
	KNN	✓	-	-	-	83	-	-	-	-	-	-	-
[44]	NB	-	✓	-	-	-	-	87.5	-	-	-	-	-
	RF	-	✓	-	-	-	-	89	-	-	-	-	-
[45]	Bagging	-	-	✓	-	-	-	-	-	-	70	-	-

	AdaBoos	-	-	✓	-	-	-	-	-	-	70	-	-
	RF	-	-	✓	-	-	-	-	-	-	65	-	-
	Extra Trees	-	-	✓	-	-	-	-	-	-	62	-	-
	XGBoos t	-	-	✓	-	-	-	-	-	-	62	-	-
[4 6]	NB	-	-	✓	-	-	-	-	-	60	56	-	-
	SVM	-	-	✓	-	-	-	-	-	59	59	-	-
	LR	-	-	✓	-	-	-	-	-	58	58	-	-
	RF	-	-	✓	-	-	-	-	-	58	55	-	-
	Decision Tree	-	-	✓	-	-	-	-	-	57	57	-	-
	Gaussian Naïve	-	-	✓	-	-	-	-	-	56	56	-	-
	KNN	-	-	✓	-	-	-	-	-	50	50	-	-
[4 7]	SVM	-	-	✓	-	-	-	-	-	55.76 +/-	-	-	-
	Naïve Bayes	-	-	✓	-	-	-	-	-	56.37 +/-	-	-	-
	Auto encoder	✓	-	-	-	81.25	-	-	-	-	-	-	-
	Auto encoder	✓	-	-	-	79.25	-	-	-	-	-	-	-
[7]	DT	-	-	-	✓	-	-	-	-	-	-	85.3 7	85.39
	GDBT	-	-	-	✓	-	-	-	-	-	-	86.9 6	86.96
Pr op	NAM	✓	✓	✓	✓	93.75	93.89	89.17	92.02	83.68	84.42	91	91.04

From Table 2, it is found that the proposed model obtains 93.75%, 89.17%, 83.68% and 91%accuracy for the ISOT, IFND, LIAR and COVID-19 FN benchmark datasets, respectively. The existing works utilizing classic Machine Learning techniques obtain accuracy ranging from 81.25% to 92% on the ISOT dataset, 87.5% to 89% on the IFND dataset, 50% to 60% on the LIAR dataset and 85.37% to 86.96% on the COVID-19 FN dataset. Similarly, the proposed model obtains the highest F1 Score of 93.89, 92.02, 84.42 and 91.04 on the ISOT, IFND, LIAR and COVID 19 FN datasets, respectively.

From the overall result analysis, it is concluded that the proposed model with Augmentation outperforms the existing non-neural augmented Machine Learning, and Deep Learning models and achieves high accurate detection of fake news. This model is successful in detecting fake news prevailing on the Twitter platform.

6 Conclusion

This paper creates a Neurally Augmented model, which augments 1D ConvNet with the

Machine Learning classification layers. 1D ConvNet processes the features neurally and these high-level neurally processed features are automatically extracted using the Augmenter layer and are augmented to create the NA_Naive Bayes, NA_Random Forest, NA_Gradient Boosting and NA_XGBoost classification layers for training and classification. The performance of the proposed model is assessed by experimenting it on the created COVID-19 Twitter dataset. The experimental results show that the proposed model classifies the COVID-19 fake news propagating on Twitter by achieving 97.25% accuracy, 0.96 precision, 0.98 recall, 0.965 F1 Score and 0.9624 AUROC Score. In addition, the proposed model is compared with non-neural augmented Machine Learning and Deep Learning models and it is found that there is a 12% increase in accuracy and a 10% increase in F1 Score. To investigate the robustness of the proposed model, it is tested on the proposed dataset and achieved a 5% increase in accuracy over the benchmark datasets. As a future research direction, the model could be extended further by using Deep Transfer Learning and unified word embeddings for feature extraction. Besides, the neutrosophic field could be utilized to handle the prediction errors to improve the classification performance further.

References

1. K. Shu, A. Sliva, S. Wang, J. Tang, H. Liu, Fake news detection on social media: A data mining perspective. ACM SIGKDD explorations newsletter. vol.19, no.1, 2017, pp. 22 - 36, https://doi.org/10.1145/3137597.3137600.
2. A. Bovet, H. A. Makse, Influence of fake news in Twitter during the 2016 US presidential election. Nature communications. vol. 10, no.1, 2019, pp. 1-14, https://doi.org/10.1038/s41467-018-07761-2.
3. G. Shrivastava, P. Kumar, R. P. Ojha, P. K. Srivastava, S. Mohan, G. Srivastava, Defensive modelling of fake news through online social networks. IEEE Transactions on Computational Social Systems, vol. 7, no.5, 2020, pp. 1159-1167, https://doi.org/10.1109/tcss.2020.3014135.
4. C. Castillo, M. Mendoza, B. Poblete, "Information credibility on twitter," In Proceedings of the 20th international conference on World wide web, 2011, pp. 675-684, https://doi.org/10.1145/1963405.1963500.
5. M. Al-Zaman, COVID-19-Related social media fake news in India. Journalism and Media, vol.2, no.1, 2021, pp.100-114, https://doi.org/10.3390/journalmedia2010007.
6. T. K. Das, P. M. Kumar, Big data analytics: A framework for unstructured data analysis. International Journal of Engineering Science & Technology. vol.5, no.1, 2013, pp.153.
7. P. Patwa, S. Sharma, S. Pykl, V. Guptha, G. Kumari, M. S. Akhtar, T. Chakraborty, "Fighting an infodemic: Covid-19 fake news dataset," In International Workshop on Combating On line Ho st ile Posts in Regional Languages dur ing Emerge ncy Si tuation, 2021, pp. 21-29. Springer, Cham, arXiv:2011.03327.
8. S. Selva Birunda, R. Kanniga Devi, "A Review on Word Embedding Techniques for Text Classification," In Innovative Data Communication Technologies and Application, 2021, pp. 267-281. Springer, Singapore, https://doi.org/10.1007/978-981-15-9651-3_23.
9. A. Shewalkar, D. nyavanandi, S. Ludwig, Performance Evaluation of Deep neural networks Applied to Speech Recognition: Rnn, LSTM and GRU. Journal of Artificial Intelligence and Soft Computing Research, 9(4), 2019, 235–245. https://doi.org/10.2478/jaiscr-2019-0006.
10. E. M. Mahir, S. Akhter, M. R. Huq, "Detecting fake news using machine learning and deep learning algorithms," In 2019 7th International Conference on Smart Computing & Communications (ICSCC), 2019, pp. 1-5. IEEE, https://doi.org/10.1109/icscc.2019.8843612.

11. V. Suhasini, N. Vimala, A Hybrid TF-IDF and N-Grams Based Feature Extraction Approach for Accurate Detection of Fake News on Twitter Data. Turkish Journal of Computer and Mathematics Education, vol, 12, no.6, 2021, pp. 5710-5723.
12. F. I. Adiba, T. Islam, M. S. Kaiser, M. Mahmud, M. A. Rahman. Effect of corpora on classification of fake news using naive Bayes classifier. International Journal of Automation, Artificial Intelligence and Machine Learning, vol. 1, no.1, 2020, pp. 80-92.
13. H. Kudarvalli, J. Fiaidhi, Experiments on Detecting Fake News using Machine Learning Algorithms. International Journal of Reliable Information and Assurance, 8, 2020, pp. 15-26, DOI:10.21742/IJRIA.2020.8.1.03.
14. N. R. de Oliveira, D. S. Medeiros, D. M. Mattos, A sensitive stylistic approach to identify fake news on social networking. IEEE Signal Processing Letters, 27, 2020, pp. 1250-1254, https://doi.org/10.1109/lsp.2020.3008087.
15. S. Helmstetter, H. Paulheim, "Weakly supervised learning for fake news detection on Twitter," In 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), 2018, pp. 274-277. IEEE, https://doi.org/10.1109/asonam.2018.8508520.
16. D. Keskar, S. Palwe, A. Gupta, "Fake news classification on twitter using flume, n-gram analysis, and decision tree machine learning technique," In Proceeding of International Conference on Computational Science and Applications, 2020, pp. 139-147. Springer, Singapore, https://doi.org/10.1007/978-981-15-0790-8_15.
17. D. S. Abdelminaam, F. H. Ismail, M. Taha, A. Taha, E. H. Houssein, A. Nabil, CoAID-DEEP: an optimized intelligent framework for automated detecting COVID-19 misleading information on Twitter. Ieee Access, 9, 2021, pp. 27840-27867, https://doi.org/10.1109/access.2021.3058066.
18. S. S. Birunda, R. Kanniga Devi, "A Novel Score-Based Multi-Source Fake News Detection using Gradient Boosting Algorithm," In 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), 2021, pp. 406-414. IEEE, https://doi.org/10.1109/icais50930.2021.9395896.
19. X. Dong, U. Victor, L. Qian, Two-path deep semi supervised learning for timely fake news detection. IEEE Transactions on Computational Social Systems, vol.7, no.6, 2020, pp. 1386-1398, https://doi.org/10.1109/tcss.2020.3027639.
20. J. A. Nasir, O. S. Khan, I. Varlamis, Fake news detection: A hybrid CNN-RNN based deep learning approach. International Journal of Information Management Data Insights, vol.1, no.1, 2021, pp. 100007, https://doi.org/10.1016/j.jjimei.2020.100007.
21. E. Cueva, G. Ee, A. Iyer, A. Pereira, A. Roseman, D. Martinez, "Detecting Fake News on Twitter Using Machine Learning Models," In 2020 IEEE MIT Undergraduate Research Technology Conference (URTC), 2020, pp. 1-5. IEEE, https://doi.org/10.1109/urtc51696.2020.9668872.
22. O. Ajao, D. Bhowmik, S. Zargari, "Fake news identification on twitter with hybrid cnn and rnn models," In Proceedings of the 9th international conference on social media and society, 2018, pp. 226-230, https://doi.org/10.1145/3217804.3217917.
23. M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, E. Muharemagic, Deep learning applications and challenges in big data analytics. Journal of big data. vol. 2, no.1, 2015, pp. 1-21, https://doi.org/10.1186/s40537-014-0007-7.
24. K. Karthikayani, A. R. Arunachalam, "A survey on deep learning feature extraction techniques," In AIP Conference Proceedings, vol. 2282, no. 1, 2020, pp. 020035. AIP Publishing LLC, https://doi.org/10.1063/5.0028564.
25. J. Y. Khan, M. Khondaker, T. Islam, A. Iqbal, S. Afroz,A benchmark study on machine learning methods for fake news detection, 2019, arXiv:1905.04749, https://doi.org/10.1016/j.mlwa.2021.100032.
26. I. Ahmad, M. Yousaf, S. Yousaf, M. O. Ahmad, Fake news detection using machine learning

ensemble methods. Complexity, 2020, https://doi.org/10.1155/2020/8885861.
27. P. Bahad, P. Saxena, R. Kamal, Fake news detection using bi-directional LSTM-recurrent neural network, Procedia Computer Science, 165, 2019, pp. 74-82, https://doi.org/10.1016/j.procs.2020.01.072.
28. R. K. Kaliyar, A. Goswami, P. Narang, "A Hybrid Model for Effective Fake News Detection with a Novel COVID-19 Dataset." In ICAART, (2), 2021, pp. 1066-1072, https://doi.org/10.5220/0010316010661072.
29. C. Brunner, A. Ko, S. Fodor, An Autoencoder-Enhanced Stacking Neural Network Model for Increasing the Performance of Intrusion Detection. Journal of Artificial Intelligence and Soft Computing Research, 12(2), 2021, 149–163. https://doi.org/10.2478/jaiscr-2022-0010.
30. Q. A. R. Adib, M. H. K. Mehedi, M. S. Sakib, K. K. Patwary, M. S. Hossain, A. A. Rasel, “A Deep Hybrid Learning Approach to Detect Bangla Fake News,” in 2021 5th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), 2021, pp. 442-447. IEEE, https://doi.org/10.1109/ismsit52890.2021.9604712.
31. S. S. Birunda, R. K. Devi, Improving Energy Efficient Aspect of Spam Classification Framework using Ensemble Machine Learning. Solid State Technology, vol. 63, no.5, 2020, pp.9032-9039.
32. M. B. Mariappan, K. Devi, Y. Venkataraman, M. K. Lim, P. Theivendren, Using AI and ML to predict shipment times of therapeutics, diagnostics and vaccines in e-pharmacy supply chains during COVID-19 pandemic. The International Journal of Logistics Management, 2022, https://doi.org/10.1108/ijlm-05-2021-0300.
33. R. Kanniga Devi, G. Elizabeth Rani, "A Comparative Study on Handwritten Digit Recognizer using Machine Learning Technique," In 2019 IEEE International Conference on Clean Energy and Energy Efficient Electronics Circuit for Sustainable Development (INCCES), 2019, pp. 1-5. IEEE, https://doi.org/10.1109/incces47820.2019.9167748.
34. R. Kanniga Devi, A Machine Learning-based Online Social Network Analysis for 360-degree User Profiling. International Journal of Innovative Technology and Exploring Engineering (IJITEE), vol. 9, no. 2S2, 2019, pp. 992-998, https://doi.org/10.35940/ijitee.b1103.1292s219
35. C. Tapsai, P. Meesad, C. Haruechaiyasak, Natural Language Interface to Database for Data Retrieval and Processing. Applied Science and Engineering Progress, vol. 14, no.3, 2021, pp. 435-446, https://doi.org/10.14416/j.asep.2020.05.003.
36. K. Stahl, Fake news detection in social media, California State University Stanislaus, 6, 2018, 4-15.
37. D. D. Beer, M. Matthee, Approaches to identify fake news: A systematic literature review, In International Conference on Integrated Science, 2020, pp. 13-22, Springer, Cham, https://doi.org/10.1007/978-3-030-49264-9_2.
38. R. Ahuja, A. Chug, S. Kohli, S. Gupta, P. Ahuja, The impact of features extraction on the sentiment analysis, Procedia Computer Science, 152, 2019, 341-348, https://doi.org/10.1016/j.procs.2019.05.008.
39. A. K. Yadav, S. K. Borgohain, Sentence generation from a bag of words using N-gram model, In 2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies, 2014, pp. 1771-1776, IEEE, https://doi.org/10.1109/icaccct.2014.7019414.
40. D. Kim, D. Seo, S. Cho, P. Kang, Multi-co-training for document classification using various document representations: TF–IDF, LDA, and Doc2Vec, Information Sciences, 477, 2019, 15-29, https://doi.org/10.1016/j.ins.2018.10.006.
41. M. S. Mokhtar, Y. Y. Jusoh, N. Admodisastro, N. Pa, A. Y. Amruddin, Fakebuster: Fake news detection system using logistic regression technique in machine learning. International Journal of Engineering and Advanced Technology, 9(1), 2019, 2407-2410, https://doi.org/10.35940/ijeat.a2633.109119.
42. M. M. Truşcă, Efficiency of SVM classifier with Word2Vec and Doc2Vec models. In Proceedings

of the International Conference on Applied Statistics, Vol. 1, No. 1, 2019, pp. 496-503), https://doi.org/10.2478/icas-2019-0043.

43. H. Ahmed, I. Traore, S. Saad, Detection of online fake news using n-gram analysis and machine learning techniques, In International conference on intelligent, secure, and dependable systems in distributed and cloud environments, 2017, pp. 127-138, Springer. https://doi.org/10.1007/978-3-319-69155-8_9.
44. D. K. Sharma, S. Garg, IFND: a benchmark dataset for fake news detection. Complex & Intelligent Systems, 2021, 1-21.
45. L. Waikhom, R. S. Goswami, Fake news detection using machine learning. In Proceedings of International Conference on Advancements in Computing & Management (ICACM), 2019.
46. S. Garg, D. K. Sharma, Phony news detection using Machine Learning and Deep-Learning techniques, In 2020 9th International Conference System Modelling and Advancement in Research Trends, 2020, pp. 27-32, IEEE. https://doi.org/10.1109/smart50582.2020.9337120.
47. V. T. Priyanga, J. P. Sanjanasri, V. K. Menon, E. A. Gopalakrishnan, K. P. Soman, Exploring fake news identification using word and sentence embeddings. Journal of Intelligent & Fuzzy Systems, 2021, 1-8.
48. S. A. Ludwig, Applying a Neural Network Ensemble to Intrusion Detection. Journal of Artificial Intelligence and Soft Computing Research, 9(3), 2019, 177–188. https://doi.org/10.2478/jaiscr-2019-0002.
49. U. Khurana, H. Samulowitz, D. Turaga, Feature engineering for predictive modeling using reinforcement learning, In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 32, No. 1, 2018, https://doi.org/10.1609/aaai.v32i1.11678.
50. A. R. Kothamachu, B. Chakraborty, Real Time Gait based Person Authentication using Deep Hybrid Network. In 2021 IEEE 4th International Conference on Knowledge Innovation and Invention (ICKII), 2021, (pp. 155-159). IEEE, https://doi.org/10.1109/ICKII51822.2021.9574763.