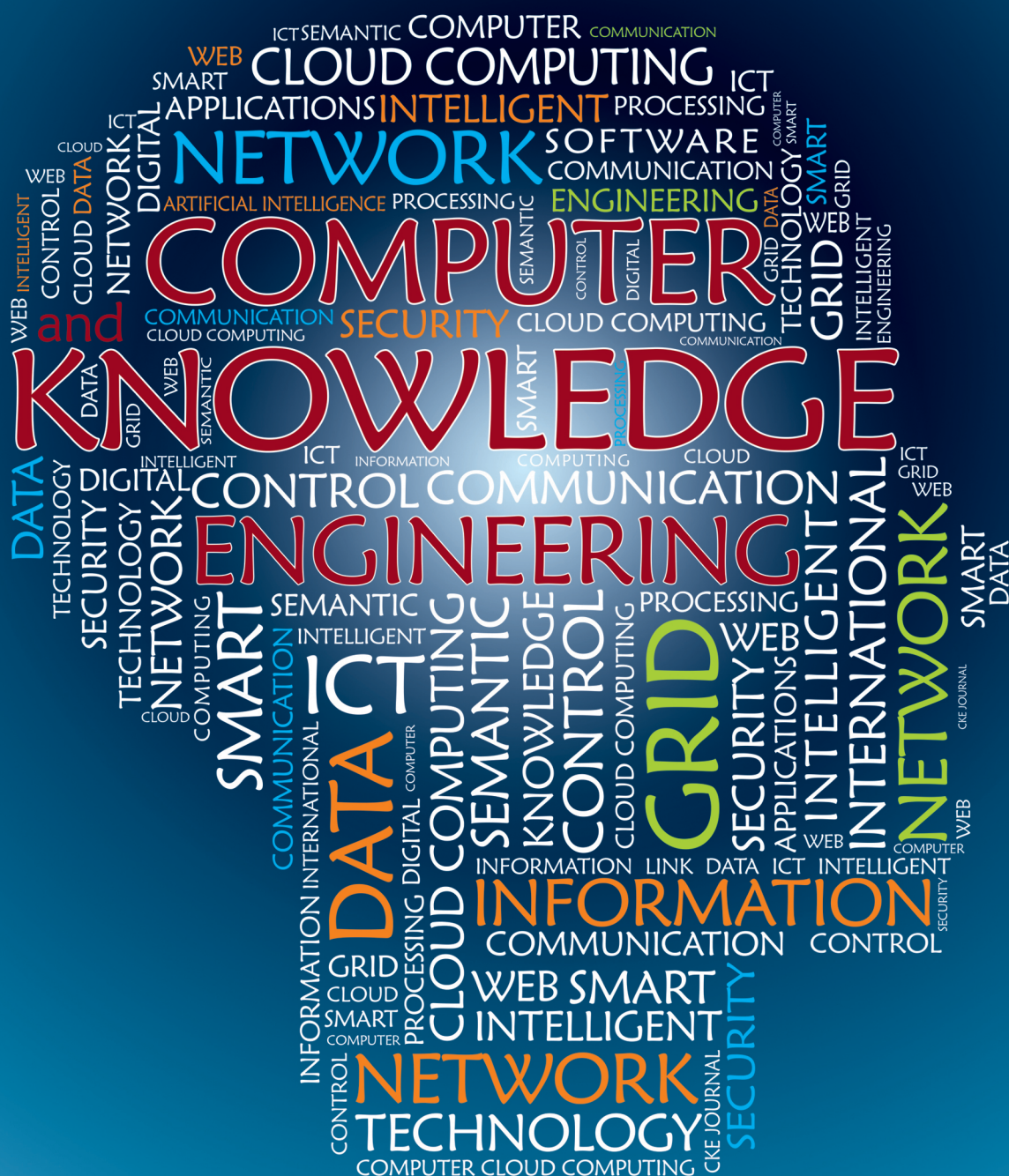


**ISSN: 2717-4123**





Journal of  
**COMPUTER AND KNOWLEDGE  
ENGINEERING**

Ferdowsi University of Mashhad

**ISSN: 2717-4123**

---

**General Director:** S. A. Hosseini Seno

**Editor-in-Chief:** M. Kahani

**Publisher:** Ferdowsi University of Mashhad

---

**Editorial Board:**

Mahmoud Naghibzadeh	Professor	Ferdowsi University of Mashhad, Iran
Mohammad H Yaghmaee-Moghaddam	Professor	Ferdowsi University of Mashhad, Iran
Dick H Epema	Professor	Delft Technical University, the Netherlands
Rahmat Budiarto	Professor	University Utara Malaysia, Malaysia
Mohsen Kahani	Professor	Ferdowsi University of Mashhad, Iran
Mohammad R Akbarzadeh-Tootoonchi	Professor	Ferdowsi University of Mashhad, Iran
Madjid Fathi	Professor	University of Siegen, Germany
Hossein Nezamabadi-pour	Professor	Bahonar University of Kerman, Iran
Ahmad Ghafarian	Professor	University of North Georgia, USA
Hamid Reza Pourreza	Professor	Ferdowsi University of Mashhad, Iran
Hadi Sadoghi-Yazdi	Professor	Ferdowsi University of Mashhad, Iran
Seyed Amin Hosseini Seno	Associate Professor	Ferdowsi University of Mashhad, Iran
Abedin Vahedian-Mazloun	Associate Professor	Ferdowsi University of Mashhad, Iran
Ebrahim Bagheri	Associate Professor	Ryerson University, Canada
Hossein Asadi	Associate Professor	Sharif University of Technology, Iran
Mahdi Kargahi	Associate Professor	University of Tehran, Iran
Hamid Reza Ekbia	Associate Professor	Indiana University, USA
Seyed Hassan Mirian Hosseinabadi	Associate Professor	Sharif University of Technology, Iran
Abbas Ghaemi Bafghi	Associate Professor	Ferdowsi University of Mashhad, Iran
Farhad Mahdipour	Associate Professor	Kyushu University, Japan

---

**Administrative Director:** T. Hooshmand

---

Journal of Computer and Knowledge Engineering

Faculty of Engineering, Ferdowsi University of Mashhad

P. O. Box. 91775-1111, Mashhad, I.R. IRAN

Tel: +98 51 38806024, Fax: +98 51 38763301, Email: [cke@um.ac.ir](mailto:cke@um.ac.ir), Site: [cke.um.ac.ir](http://cke.um.ac.ir)

## CONTENTS

<b>Exploring Effective Features in ADHD Diagnosis among Children through EEG/Evoked Potentials using Machine Learning Techniques</b>	Faezeh Rohani- Kamrad Khoshhal Roudposhti- Hamidreza Taheri-Ali Mashhadi Andreas Mueller	1
<b>Embedding Knowledge Graph through Triple Base Neural Network and Positive Samples</b>	Sogol Haghani - Mohammad Reza Keyvanpour	11
<b>Efficient and Deception Resilient Rumor Detection in Twitter</b>	Milad Radnejad - Zahra Zojaji Behrouz Tork Ladani	21
<b>A Deep Neural Network Architecture for Intrusion Detection in Software-Defined Networks</b>	Somayeh Jafari Horestani Somayeh Soltani -Seyed Amin Hosseini Seno	31
<b>Meta-Learning for Medium-Shot Sparse Learning via Deep Kernels</b>	Zohreh Adabi-Firuzjaee Sayed Kamaledin Ghiasi-Shirazi	45
<b>The Impact of Preprocessing Techniques for Covid-19 Mortality Prediction</b>	Soodeh Hosseini - Zahra Asghari Varzaneh	57





# Exploring Effective Features in ADHD Diagnosis among Children through EEG/Evoked Potentials using Machine Learning Techniques\*

Research Article

Faezeh Rohani<sup>1</sup> Kamrad Khoshhal Roudposhti<sup>2</sup> Hamidreza Taheri<sup>3</sup> Ali Mashhadi<sup>4</sup> Andreas Mueller<sup>5</sup>

**Abstract:** With the aid of intelligent system approaches, the present study aimed at extracting and investigating effective features for detecting Attention-Deficit/Hyperactivity Disorder (ADHD) in children. With this end in view, 103 children, aged from 6 to 10, were recruited for this study, among which 49 cases were assigned to the treatment group (ADHD children) and the remaining 54 cases to the control group (healthy children). The disorder diagnosis was performed using the well-known, relevant psychological questionnaires and clinical interviews with expert psychologists. Data collection consisted of EEG signals in eyes open and eyes closed states, as well as GO/NOGO task for about 3 hours for every participant. The extracted features consisted of the amplitudes and latency in Event-Related Potential (ERP) and the power spectrum in the sleep mode signals. Approximately 826 features of 19 channels were extracted in the standard 10-20 system and different task conditions. A set of features were selected with the aid of the feature selection methods, and then the selected features were analyzed by neuroscientists, and the irrelevant ones were removed. Next, the classification methods and their performance evaluation were applied. Finally, the best results in terms of the corresponding feature vector and classification method were presented. The healthy and ADHD groups were classified with 75.8% accuracy using the Support Vector Machine (SVM) method. The results showed that the use of selection of effective features with the aid of intelligent system techniques under the supervision of experts leads us to reach robust biomarkers in the detection of disorders.

**Keywords:** Attention Deficit Hyperactivity Disorder (ADHD), EEG/Evoked Potentials, Feature Extraction, Feature Selection

## 1. Introduction

Psychiatric disorders are complex because psychological, biological, and genetic factors influence cognition, emotions, and behavior in certain areas [1]. With questionnaires and clinical interviews, it has been found that the diagnosis of disorders relies on mental descriptions and external observations. Therefore, such diagnoses are prone to error due to the complexity of psychiatric disorders, intrinsic mentality, and even the use of the *Diagnostic and Statistical Manual of Mental Disorders*, 5th Edition: DSM-5 [2] diagnostic guide. Accordingly, researchers have made

significant efforts to obtain biological markers of mental disorders [3-10]. Most of these markers are genetic, biochemical, blood epigenetic, and blood plasmatic [11, 12]. However, some of these markers are electroencephalographic letters, induced potentials, and magnetic resonance imaging [13]. Unhealthy groups and healthy individuals have complex characteristics and are difficult to detect using individual markers. Henceforth, the symptoms of the diagnosis can be obtained by different neurobiological pathways [14]. Attention-Deficit Hyperactivity Disorder (ADHD), a neurological disorder, affects an estimated 4% to 12% of school-aged children worldwide [15]. Based on DSM-5, this disorder consists of three types, namely hyperactive and impulsive, inattentive, and combined [2].

The present study investigated and extracted the Electroencephalography (EEG) and Event-Related Potential (ERP) features that have been studied concerning the EEG and ERP indicators and brain function of ADHDs [16-19]. The principal advantage of using ERP includes the possibility of nonaggressive cognitive processes in milliseconds [20]. In recent years, machine learning methods have been widely used in the medicine and health realms [21-23]. Nevertheless, in psychiatry, due to limitations such as the absence of data, fear of distancing from diagnostic measures, and inadequate knowledge, this technique has been applied less frequently. However, the needs suggest that the combinatorial biomarkers have better performance compared with individual values [24].

Extensive research at the Switzerland Brain and Trauma Foundation has shown that biological boundaries can be traced through the stimulated potential to create biological markers (a measurable indicator for biological conditions) [25]. Moreover, in this research center, psychological neuroscience is used as an indicator to identify a specific disorder in the brain. The foundation also believes that none of the markers can help the diagnosis alone but that the diagnosis must be made through the proper usage of a set of these markers [6]. In this view, researchers using machine learning methods for the separation of ADHD and control groups in adults (74 cases in the ADHD group, 74 cases between 18-50 years old in the control group) observed that with GO/NOGO task, the accuracy of the Support Vector Machine (SVM) method was 92 % [6].

In another study, researchers using machine learning

\* Manuscript received: 12 June 2021, Revised, 26 June 2021, Accepted, 10 September 2021.

<sup>1</sup> PhD Candidate, Department of Computer Engineering, Lahijan Branch, Islamic Azad University, Lahijan, Iran.

<sup>2</sup> Corresponding author. Assistant Professor, Department of Computer Engineering, Lahijan Branch, Islamic Azad University, Lahijan, Iran. **Email:** kamrad@liau.ac.ir.

<sup>3</sup> Professor, Department of Motor Behavior, Ferdowsi University of Mashhad, Mashhad, Iran.

<sup>4</sup> Professor, Department of Clinical Psychology, Ferdowsi University of Mashhad, Mashhad, Iran.

<sup>5</sup> Professor in Brain and Trauma Foundation, Grisons/Switzerland, Chur, Switzerland.

methods on 117 adults (67 participants in the ADHD group and 50 participants in the control group) showed that the classification accuracy for separating groups was about 69.2% in Visual Continuous Performance Test (VCPT) mode and 72.6 and 70.9% in eyes closed and eyes open states. However, in the form of scoring, the results showed up to 82.3 % change [26].

Oztoprak *et al.*, using the time-frequency amplitude characteristics of EPR with strop test, classified the ADHD and control groups with 100% accuracy using the SVM method. This accuracy was for 3 to 5 features in the delta frequency band. In their study, all participants were male and in the age range of 6 to 12 years old, and the sample included 44 cases in the ADHD group and 38 cases in the control group [27].

Helgadotter *et al.* had 310 participants in the ADHD group and 351 participants in the control group, aged from 5.8 to 14. Their method accuracy rate was about 81% when analyzed by age and 73% the other way round (i.e., not based on age) [3].

Heinrich *et al.* investigated the neural mechanisms of motor control using the potentials in combination with MRI, obtaining a classification rate of 90% in a linear analysis. The

study suggested that both cognitive and motor inhibition should be regarded as fundamental problems in children with ADHD [28].

Meuller *et al.* used machine learning techniques to separate ADHD from healthy participants. Their experimental EEG and ERP data were collected from 181 ADHD and 147 healthy participants. Spectral power, ERP amplitude, and latency measures were extracted and used as a feature vector for the input of their machine-learning framework. ADHD patients and healthy participants were classified by logistic regression model with accuracy values between 72% and 76%, while their specificity values slightly decreased over time (between 64% and 67%) [29].

During the review of the related literature, various studies have reported good EEG classification capability and ERP. These methods had different accuracy rates according to the selection of different effective features, their numbers of features, and the applied classification technique. Therefore, the number of features and the type of features are effective in obtaining accuracy. With this end in view, this study aims at extracting effective features to diagnose ADHD in children under the supervision of neuroscientists. Figure 1 shows the workflow of the current study.

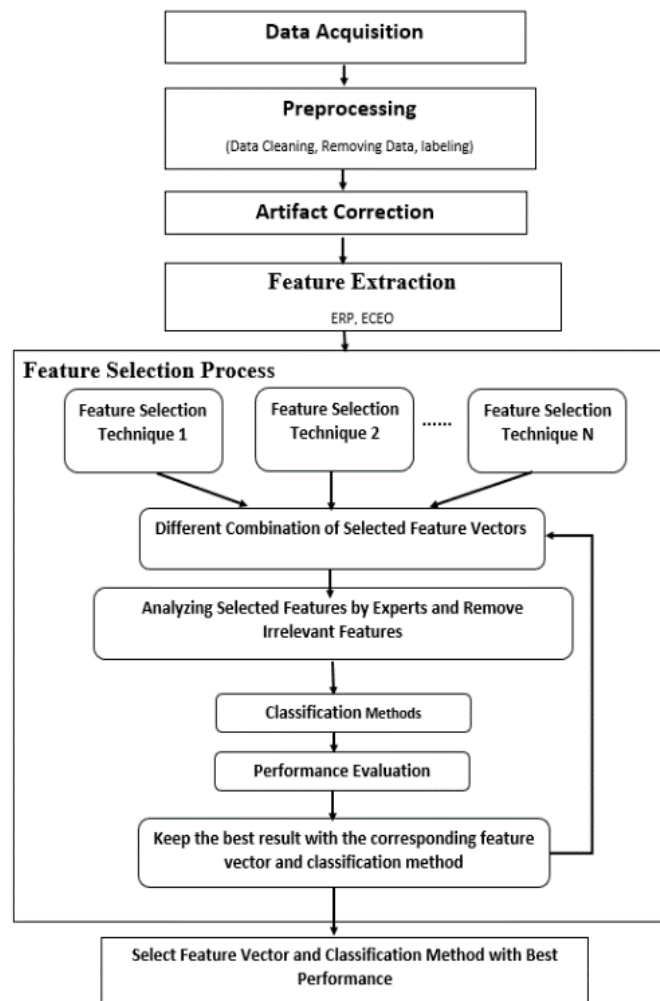


Figure 1. Workflow of the research framework. ECEO denotes EEG signals from eyes closed and eyes open states

## 2. Data collection

### 2.1. Participants

The participants consisted of 103 participants from 7 to 10 years old. According to the DSM-5, 49 participants were diagnosed with ADHD (22 females, 27 males), and the remaining 54 participants were healthy participants (24 females, 30 males). The ADHD participants were recruited from clinics, and the members of the control group were selected from summer leisure classes of Ferdowsi University of Mashhad, Iran. Deprivation criteria in this study were an IQ scoring below 75, epilepsy, and comorbidities disorder with ADHD. Control patients who consumed a drug were not included in the study. The ADHD patients who had medication under the supervision of their doctors did not take drugs before testing. Therefore, all the participants did not receive any medication at the time of testing.

### 2.2. Procedure

Data was collected in the motor behavior lab at Ferdowsi University from July 2019 to February 2020. All ADHD participants were screened medically by medical doctors. As the first step in this project, parents filled out a set of such questionnaires as Child Behavior Checklist (CBCL), AMEN, ADHD, Cognitive Change Index (CCI), and the Swanson, Nolan, and Pelham (SNAP). For the IQ test, the Riven test was applied [30]. Participants were tested in a single session for about 3 hours, including recording their EEGs/ERPs and taking the IQ tests. The parents were aware of this study and agreed to use clinical data for research purposes. They had signed consent forms before the start of the study.

### 2.3. EEG and ERP task

EEG was recorded for 10 minutes (5 minutes with eyes closed and 5 minutes with eyes opened), and ERP was recorded for 20 minutes. The ERP test was Go/NOGO task that contained 400 trials. This task had four conditions, namely A-A (animal-animal), A-P (animal-plant), P-H (Plant-Human), and P-P (plant-plant). Each condition involved 100 trials. The task had novel sounds along with human images in the P-H state. The details of this task are provided in [5].

### 2.4. Data recording and pre-processing

The EEG was recorded with the aid of the "NeuroAmp® x23" and "ERPrec software" (BEE Medic GmbH, Switzerland). The Raw EEG was analyzed by Matlab. The sampling rate of the input signals was 500 HZ, and it was referenced with linked-earlobes and filtered by band-pass between 0.5 and 50 HZ with a 45-55 Hz notch filter. The Electro-Cap electrode application system (19channel, Electro-Cap, International Inc, USA) that worked with the international 10-20 system was used in the present study. The impedance for all electrodes was not more than five kOhm. Neuronal activity of 19 brain channels including Fp1, Fp2, F3, F4, F7, F8, Fz, C3, C4, Cz, T3, T4, T5, T6, P6, P3, P4, Pz, O1, and O2 and linked earlobes and such frequency bands as Delta (0.5-4 Hz), Theta (4-8 Hz), Alpha (8-12 Hz), Beta (12-30), and Gamma (30-50 Hz) were recorded.

For artifact removing, the starting raw EEGs were first removed. Then eye-blink and horizontal eye movements were detected, with the aid of independent component analysis (ICA) decomposition removed from the EEGs. The remaining artifacts were removed from the slow (e.g., sweat artifact)/fast (e.g., muscle artifacts) wave correction (i.e., excessive activity in the 0-3 Hz and 20-50 Hz frequency bands). Finally, the amplitudes range of more than 100  $\mu$ V were removed.

## 3. Method

### 3.1. Feature extraction

In signal processing, features are generally divided into the time, frequency, and time-frequency domains. The time-domain characteristics refer to directly extracted features from the signal itself without altering such signal spaces as mean, standard deviation, energy and power, entropy, skewness, kurtosis, auto-regressive coefficient, zero-crossing percentile, and Hjorth parameters [31-39].

The purpose of applying a mathematical transformation to a signal is to obtain additional information that is not available in the original raw signal. However, time domain-based analysis of the signals is popular, but in many cases, the useful information of the signal lies in its frequency content, which is called the signal spectrum. Simply, the spectrum of a signal represents the frequencies' amplitude in that signal. Examples of approaches for extracting frequency range features are the Fourier transform, Short-Term Fourier Transform (STFT), spectral entropy, spectral centroid, spectral spread, spectral roll-off, harmonic parameters, and power spectral density [40- 43].

According to the description of the extraction feature, the features extracted in this study included the density spectrum of 5 frequency bands and 17 channels of EEGs in eyes closed and eyes opened states. The spectral power density was a description of power distributed over the frequencies in the limited data set signal, so the power spectrum density unit was the power in each frequency unit (watts per Hz). The density spectrum indicates at what frequencies the signal strength changes are weaker and at what frequencies they are stronger.

Amplitude and latency peaks were extracted for ERP in eight task conditions for the 17 channels [5]. The conditions included four main states (A-A, A-P, P-P, and P-H) and four mixture conditions amid all states (A-A/P, A-P-A-A, P-P/H, P-H-P-P). For ERP, usually, the first, second, and third peaks from the curves would be extracted.

The VCPT has two stimuli, and usually, the features should be extracted on the second stimulus, and the events and peaks are examined after the second stimulus appearance. In this case, the peaks will be considered after the second stimulus appearance and are positive or negative. The first positive peak is called P100, the second P200, and the third P300. The first negative peak is called N100, and the second negative peak is called N200, and this cycle, as shown in Figure 2 [44], continues. Therefore, knowing that the second stimulus appears in 1,400 milliseconds, the signal analysis time interval can be from 1,300 to 2,400 milliseconds, and in cases where it is necessary to check the events of the first stimulus, the time interval is between 300

to 1,100. Besides, to align all the signals, a baseline is set in the range of 1,300 to 1,400 milliseconds.

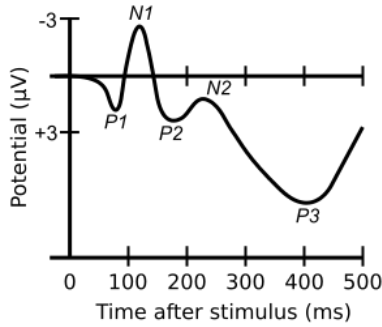


Figure 2. A waveform showing several ERP components, including the N100 (labeled N1) and P300 (labeled P3). Note that the ERP is plotted with negative voltages upward, a common but not universal practice in ERP research.

In ERP, to obtain the appropriate peaks, the average ERP diagrams were considered for all participants. Moreover, to obtain the lowest and highest points along with the signals, curves of the time window, which are one of the features of ERP components, were considered. The size of the time window was fixed at 45% of the time interval from the highest peak to the adjacent peak in the average main ERP curve. To reach the main peak in this time window, different methods such as measuring the area under the ERP curves in the time window range or measuring the curve in the specified time window are applied. In this study, the curve range method has been used. Another list of features, including arousal index, reaction time, theta/beta ratio, C3/C4 index, and omission and commission error, was also extracted. Features related to reaction time, commission, and omission are behavioral parameters compared with other features that are characteristic of the brain.

One of the major points in extracting features is to identify the important frequency bands for specific disorders. Based on the past studies, it has been found that the significant frequency bands in the diagnosis of ADHD are F3, F4, F8, Fz, C3, C4, Cz, T5, T6, P2, O1, and O2. However, since the purpose of the study was to obtain more variant characteristics, all frequency bands except FP1 and FP2 (due to artifact in the data and meanness in ADHD) were examined. The importance of the features is described in the feature selection part below.

### 3.2. Feature selection

A set of features has been extracted from the EEG/ERP signals, and it is evident that all of these features did not relate to ADHD. Thus, it was necessary to reduce features to achieve effective features, prevent over-fitting, and reduce computational efforts [45]. Therefore, in this study, to limit the number of features, a combinational approach using intelligent feature selection methods with a neuroscientist's supervision was proposed. Based on this approach, several feature selection methods have been used to select different sets of effective features. Then neuroscientists examined the selected features and selected a set of effective features.

One of the feature selection methods that was used in the present study was the combined Hybrid Structured sparse learning method [46]. This method is the same as the

regression of Least-squares, which contains two regulating modes, L1-norm and L2.1-norm, as follows:

$$\min_w J(W) = \|X^T W - Y\|^2 + \gamma_1 \|W\|_{1.1} + \gamma_2 \|W\|_{2.1} \quad (1)$$

Equation 1 is a target function in which  $X = [x_1, x_2, \dots, x_n] \in R^{d \times n}$  where  $n$  training samples and  $d$  features are applied, and  $Y = [y_1, y_2, \dots, y_c] \in R^{n \times c}$  where  $c$  is the number of classes for each  $x_i$  training data. By finding the optimal values of the parameters  $\gamma_1$  and  $\gamma_2$ , the optimal coefficient matrix for each feature of  $x_i$  can be obtained. To get the best  $k$  features, the features would be sorted based on their effectiveness, and then the  $k$  feature is selected with the highest rank.

The sequential floating forward selection (SFFS) [47] is another implemented feature selection method in the present study. This algorithm finds an optimal subset of features by addition (adding a new feature to the subset of previously selected features) and subtraction (removing a feature from the subset of previously selected features).

Therefore, amongst all the features selected by automatic methods, after being analyzed by an expert, a set of features were finally selected. Table 1 shows the group of features.

Table 1. The group of features

Group	Features name
EC/EO/VCPT	Arousal index
EC/EO/VCPT	Theta/beta ratio
EC/EO	frequency spectra (coherence)
Behavioral in VCPT	Omission errors
	Commission errors
	Reaction time
ERP	Min amplitudes
	Max amplitudes
	Min latency
	Max latency

### 3.3. Classification

Supervised machine learning methods work in such a way that in them, a set of input vectors such as  $X = \{x_n\}$  and the corresponding output vector  $T = \{t_n\}$  are given. The goal for the machine, using those training data for the new  $x$  input, is to be able to predict  $t$  [48]. In this regard, two distinct modes can be considered. Regression, in which  $t$  is a continuous variable and classification and belongs to a discrete set. In the learning process, the system first needs to be trained, and then in the testing process, the trained system is used to predict the output concerning the new input values. Support Vector Machine (SVM) is a well-known supervised machine learning method and one of the simplest types of SVMs (i.e., linear SVM), which finds a hyperplane that separates sets of positive and negative samples with the maximum distance. A couple of the most accurate approaches, SVM and ensemble classification models, were used and reported in this study.

### 3.4. Cross-validation and evaluation

In the supervised learning methods, there are two sets of data

(i.e., train data set and the test data set), which are managed in different ways for validation. Here, the K-fold method was used for validation. K-fold cross-validation is one of the most common methods of validating machine learning systems. In this method, the whole set of data is divided into K equal parts. From the K parts, K-1 parts are used as a set of training data, based on which the model is constructed, and with the remaining part, the testing process is performed. The number of repetitions of this process will be K times such that each K part is used only once for evaluation, and the accuracy for the model is calculated each time. In this evaluation method, the final accuracy of the system will be equal to the average of all obtained K accuracies [49].

**Confusion matrix:** This matrix shows how the classification technique works. This is according to the separate input datasets for different class categories [50]. In what follows, TP, TN, FN, and FP and their relationships in the present study are explained.

- True Negative (TN) = correctly rejected. This rate indicates the number of records whose true category has been negative, and the classifier has identified them as negative. In this study, it is the correct diagnosis of the control group, the participants who have been correctly diagnosed as healthy ones.
- False Positive (FP) = incorrectly identified. The misdiagnosis with ADHD, meaning control group participants who have been misdiagnosed with ADHD.
- False Negative (FN) = incorrectly rejected. The misdiagnosis of the control group. That is the participants who were ADHD but were misdiagnosed as healthy ones.
- True Positive (TP) = correctly identified. Correct diagnosis of ADHD, participants who were in the ADHD group and were diagnosed with ADHD.

**Accuracy:** The most important criterion for determining the performance of the classification technique is the accuracy criterion. This measure computes the total accuracy of a classification and illustrates that the designed classification correctly classifies a few percent of the entire set of experimental records. The accuracy of the classification based on the concepts expressed in the confusion matrix is calculated by the following equation:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

**Scoring:** The main scoring criterion is to evaluate the performance of the Receiver Operating Characteristic (ROC) area under the receiver operating characteristic curve (AUC). This criterion shows the overall performance of a model by combining the actual-positive rate (sensitivity) and the false positive rate (1-specificity). For binary classifiers, the AUC value varies from 0.5 to 1, in which 1 indicates the full performance of a classifier [51].

#### 4. Results

The effectiveness of the proposed method in this paper has been investigated with the aid of data collected from control group children and children with ADHD. In all classification processes, the 5-fold cross-validation approach was applied to validate the model, and for evaluation, accuracy criteria from the confusion matrix of each classifier were calculated. To stabilize the final output of the classifiers and provide a reliable answer based on the evaluation criteria, the results were an average of 10-trial classification.

In the first step, the data was presented directly to the classifiers without selecting the subset of features. In the second step, the data was first presented to the feature selection algorithms and then to the classifiers. After obtaining their accuracies, the features were checked by the neuroscience specialist, and then the features were given to the classifiers again. The final output is shown in Table 2. The total number of features was 826, the number of features in each section was 30, 5, and 37, and finally, the number of effective features that have been obtained in combination methods was about 113 features.

Based on the results, all of the selected methods and features were not approved by the specialist, so according to the expert's opinion and previous studies, combining the features was necessary to obtain the appropriate accuracy to separate the control group from the ADHD group. Moreover, based on the results, 37 features were approved by experts [9, 52] for the data of this study that had an accuracy of 61.9%, which slightly showed the specific characteristics of this research data.

Table 2. The performance of different feature selection techniques and classifier models

o Features	Feature Selection	Model	TP	TN	FP	FN	ACC	AUC	Expert Approved
826	No Feature Selection	Tree	80	67	20	33	73.8	0.74	–
826	No Feature Selection	Ensemble RUS Boosted tree	85	61	15	39	73.8	0.68	–
113	Combine	SVM-Linear	83	67	17	23	75.8	0.75	Yes
37	Neuroscience	Ensemble Subspace Discriminant	69	54	31	46	61.9	0.58	Yes
30	Hybrid Structured Sparse Learning (HSSL)	Logistic Regression	81	88	19	12	84.5	0.90	–
5	Sequential Floating Forward Selection (sffsAB)	Cosine KNN	96	59	4	41	78.6	0.83	–
64	Sequential Floating Forward Selection Standard (SffsSt)	Ensemble Subspace Discriminant	70	76	30	24	70.9	0.75	–



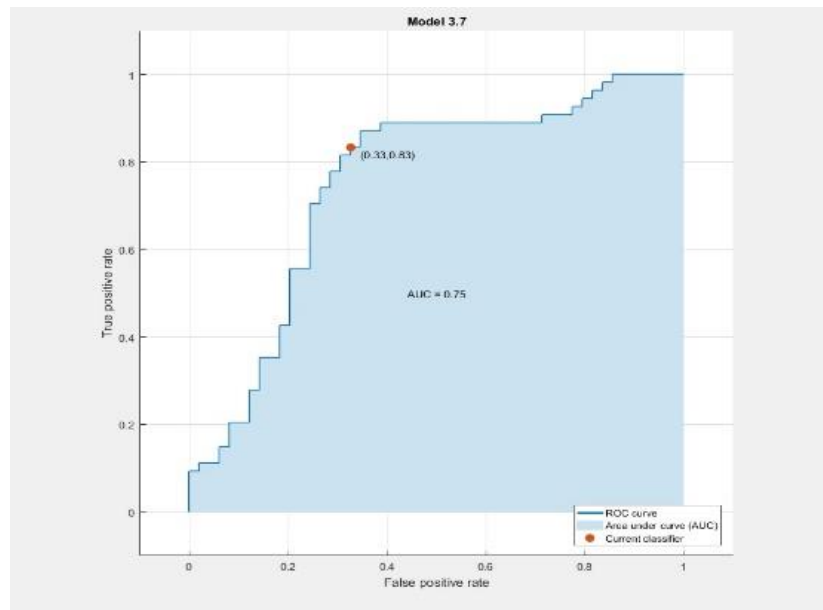


Figure 3. ROC score of the selected method

The methods which used the feature selection method of HSSL and SFFS with 84.5% and 78.6% accuracy were not approved by the neuroscientist, and to the best of neuroscientist's knowledge, most of the selected features were not relevant to the diagnosis of ADHD. Therefore, under the supervision of the neuroscientist, a small number of significant features were selected as effective features.

By combining the features obtained from the selection methods that have been approved by the specialists and the proposed and approved features of the neuroscientist concerning the significance of ADHD and behavioral features, 113 features were obtained with a 75.8% accuracy rate. As shown in Table 2, using the SVM method, the correct detection rate of ADHD (TP) and control (TN) were 83% and 67%, respectively. Accordingly, the misdiagnosis of ADHD (FP) and control (FN) groups were 17% and 33%, respectively. Figure 3 shows the ROC diagram of the classifier result.

## 5. Discussion

In this paper, all the mentioned features were extracted from the raw signal in the closed and open eye modes, as well as ERP and behavioral features. To select the best features, we used the methods of selecting the feature of the HSSL and SFFS. The method of extracting and selecting the feature vector from raw signals significantly impacts the obtained results. Consequently, we tried to use brain signal processing and extract the best features in diagnosing ADHD in the first stage. Then those features were approved by a specialist.

In the present study, features included the theta, beta, and alpha frequency bands of Pz, O1, O2, T5, T6, C6, Cz, Fz, C3, C4, F3, F4, and F8 electrodes, the maximum and minimum latencies, and the highest and lowest domains in ERP. The effective features were obtained through feature selection methods with the approval of neuroscientists, and finally, for classification, the linear SVM was used. The feature vector with 113 features, which was obtained with a combination strategy, was used for the classification process by the SVM method. The obtained result showed that the

accuracy of the proposed approach was 75.8%.

Due to changes in brain functionality and the instability of their brain signals, the diagnosis of ADHD in children aged 6 to 10 is very limited in the literature. Therefore, to compare with previous studies, the same research method and executive protocol must be applied to record data. This is a research constraint that limits comparison with accessible studies. Table III summarizes the studies conducted on the diagnosis of ADHD in children.

As shown in Table 3, different methods have been used in different studies for data collection. Moreover, the applied tests and the data registration conditions were different. One of the advantages of the present study is using all conditions in one setting: raw signal and ERP signal.

Some studies like [3], have only used closed-eye data for diagnosis and analysis, in which case the type of data and the number of participants examined affected the results. In [3], due to the large number of participants, one of the prominent features was the age of the participants, while the number of participants of the present study was fewer, and all conditions, that is, raw signal (eyes closed and eyes opened) and the event-dependent potential were used.

In some studies like [27], only male participants were recruited, and ERP was also performed by color stroop test. In such studies, with about 3 to 5 behavioral features (omission and commission error), an accuracy of 99.5% was achieved. With respect to what experts claim, this number of features is not acceptable and comparable with the present study. In this study, with a few features, the observed accuracy was above 80%. However, some of the features were approved by the experts as criteria for ADHD detection.

In [56], to diagnose ADHD through the pre-forehead cortex, NIRS data, stroop test, and behavioral data were collected where with the aid of SVM, the accuracy rate was 86%. The difference between this method and the one in the current study is the type of data collection procedure followed.

Table 3. Studies conducted on the diagnosis of ADHD in children. SVM-RFE denotes support vector machine recursive feature.

Ref.	Number of participants	Age/ gender	Accuracy	Classification method	Feature selection method	Selected feature(s)	Device and system	Data collection method
[27]	70 ADHD 37 Control	6 to 12 Boy	99.5%.	SVM	(SVM-RFE)	Three features (behavioral features) include omission, commission, errors	64 electrode 10-10 system	ERP with Stroop task
[54]	62 ADHD 39 Control	7 to 16 Boy/ girl	58-63% 85%	Statistical analysis with Ancona	No	Theta/beta ratio Theta at Cz Beta at Cz Omission errors	19 channel 10-20 system Mitsar 201	ERP Go/No Go
[55]	7 ADHD 7 Control	8 to 12 Boy/girl	-	statistical analysis	No	ERP Spectral perturbation Inter-trial coherence Time locked on each stimulus Omission, commission errors reaction time	14 channels (Fz, F3, F4, Cz, C3, C4, Pz, P3, P4, Oz, O3, O4, and M1-M2 for the left and right mastoids), 10-20 system Ant company	ERP Go/No Go
[56]	108 ADHD 108 Control	~10 Boy/girl	86%	SVM	No	Reaction time Behavioral	10-20 system NIRS <sup>i</sup> system	Reverse Stroop task
[3]	310 ADHD 350 Control	5.8 to 14 Boy/girl	76%	SVM	No	20 features Age dependent Coherence Power, Relative power	17 electrodes: Fz, F3, F4, F7, F8, Cz, C3, C4, T3, T4, T5, T6, Pz, P3, P4, O1 and O2 NicoletOne	Eyes Closed EEG
[28]	19 ADHD 21 Control	9 to 14 Boy/girl	90%	Statistical analysis	statistical analysis with Ancona	ERP	BrainAmp 10-20 system	ERP Go/No Go and TMS data

## 6. Conclusion

In this study, with the aid of intelligent techniques under a neuroscientist's supervision for diagnosing ADHD, a new strategy was proposed to select effective EEG/ERP-based features. A new dataset was also collected for applying and evaluating the proposed method. The limitations of previous researches were discussed it was tried to improve them. The automatic feature selection techniques usually try to find a set of features that increase the accuracy measurement. Since the number of samples is limited, the automatic techniques can be affected by the experimental-based artifacts and can find some irrelevant features that can increase the system's accuracy for that specific dataset but might not work in others. Thus, we have proposed an expert's supervision-based feature selection technique to achieve an acceptable result with the expert's approval. In this study, due to the characteristics of the data, the effective feature was confirmed by experts. As experts stated, integrating all dimensions (including lifestyle, questionnaire, interview, and psychiatric examination) is essential in the diagnostic process [57]. In short, the results are promising and can be expended by taking into account such factors as the effects of age on more data samples. By increasing the number of features, the feature selection techniques show a weak performance or will be a time-consuming task. Thus, using optimization methods for the mentioned purpose can be a proper solution for future related works.

## Declaration of competing interest

The authors have no conflict of interest to disclose.

## 7. Acknowledgements

A special thanks goes to the Brain and Trauma Foundation in Switzerland, headed by Dr. Andreas Mueller and his coworker Gian Candrian, who supported the study of Switzerland Opportunity, the HBI Foundation, and the BioMed Institute, which provided the hardware and software needed to record the signal. We also appreciate the Ferdowsi University School of Sports Science and Soroush Psychological Clinic for their help in collecting data and selecting samples. Without the support of these groups, the current research would not be possible at its best quality. We also appreciate all the children and their parents who patiently accompanied us.

## 7. References

- [1] Guntern, G. "Auto- organization in human systems", Behavioral Science, Vol. 27(4), pp. 323-337, 1982.
- [2] Association, A.P., "Diagnostic and statistical manual of mental disorders (DSM-5®)", American Psychiatric Pub, 2013.
- [3] Helgadóttir, H., Gudmundsson, Ó. Ó., Baldursson, G., Magnússon, P., Blin, N., Brynjólfssdóttir, B., ... & Johnsen, K., "Electroencephalography as a clinical tool

- for diagnosing and monitoring attention deficit hyperactivity disorder: a cross-sectional study", *BMJ open*, Vol. 5(1), 2015.
- [4] Müller, A., Candrian, G., Kropotov, J., "ADHS-Neurodiagnostik in der Praxis", Springer-Verlag, 2011.
- [5] Mueller, A., Candrian, G., Kropotov, J. D., Ponomarev, V. A., & Baschera, G. M., "Classification of ADHD patients on the basis of independent ERP components using a machine learning system", In *Nonlinear biomedical physics*, Vol. 4, No. 1, pp. 1-12, BioMed Central, June, 2010.
- [6] Mueller, A., Candrian, G., Grane, V. A., Kropotov, J. D., Ponomarev, V. A., & Baschera, G. M., "Discriminating between ADHD adults and controls using independent ERP components and a support vector machine: a validation study", *Nonlinear biomedical physics*, Vol. 5(1), pp. 1-18, 2011.
- [7] Dubreuil-Vall L, Ruffini G, Camprodon J. A., "Deep learning convolutional neural networks discriminate adult adhd from healthy individuals on the basis of event-related spectral eeg", *Frontiers in neuroscience*. Apr 9;14:251, 2020.
- [8] Furlong S, Cohen J. R, Hopfinger, J., Snyder, J., Robertson, M. M., Sheridan, M. A., "Resting-state EEG Connectivity in Young Children with ADHD", *Journal of Clinical Child & Adolescent Psychology*, Aug 18:1-7, 2020.
- [9] Kaiser A, Aggensteiner PM, Holtmann M, Fallgatter A, Romanos M, Abenova K, Alm B, Becker K, Döpfner M, Ethofer T, Freitag CM. "EEG Data Quality: Determinants and Impact in a Multicenter Study of Children, Adolescents, and Adults with Attention-Deficit/Hyperactivity Disorder (ADHD)", *Brain Sciences*, Feb, Vol. 11(2), pp. 214, 2021.
- [10] Tosun M. Effects of spectral features of EEG signals recorded with different channels and recording statuses on ADHD classification with deep learning. *Physical and Engineering Sciences in Medicine*. May 27:1-0, 2021.
- [11] Cubero-Millán, I., Ruiz-Ramos, M. J., Molina-Carballo, A., Martínez-Serrano, S., Fernández-López, L., Machado-Casas, I., ... & Muñoz-Hoyos, A., BDNF concentrations and daily fluctuations differ among ADHD children and respond differently to methylphenidate with no relationship with depressive symptomatology. *Psychopharmacology*, Vol. 234(2), pp. 267-279, 2017.
- [12] Wang, L. J., Li, S. C., Lee, M. J., Chou, M. C., Chou, W. J., Lee, S. Y., & Kuo, H. C., "Blood-borne MicroRNA biomarker evaluation in attention-deficit/hyperactivity disorder of Han Chinese individuals: an exploratory study", *Frontiers in psychiatry*, 9, 2018.
- [13] Kropotov, J. D., Grin-Yatsenko, V. A., Ponomarev, V. A., Chutko, L. S., Yakovenko, E. A., & Nikishina, I. S., "ERPs correlates of EEG relative beta training in ADHD children", *International journal of psychophysiology*, Vol. 55(1), pp. 23-34, 2005.
- [14] Insel, T. R., & Cuthbert, B. N., "Brain disorders? Precisely", *Science*, Vol. 348(6234), pp. 499-500, 2015.
- [15] Krieger, V., & Amador-Campos, J. A., Assessment of executive function in ADHD adolescents: contribution of performance tests and rating scales. *Child Neuropsychology*, Vol. 24(8), pp. 1063-1087, 2018.
- [16] Yang, M. T., Hsu, C. H., Yeh, P. W., Lee, W. T., Liang, J. S., Fu, W. M., & Lee, C. Y., "Attention deficits revealed by passive auditory change detection for pure tones and lexical tones in ADHD children", *Frontiers in human neuroscience*, Vol. 9, pp. 470, 2015.
- [17] Lenartowicz, A., & Loo, S. K., "Use of EEG to diagnose ADHD", *Current psychiatry reports*, Vol. 16(11), pp. 498, 2014.
- [18] Kakuszi, B., Tombor, L., Papp, S., Bitter, I., & Czobor, P., "Altered response-preparation in patients with adult ADHD: A high-density ERP study", *Psychiatry Research: Neuroimaging*, Vol. 249, pp. 57-66, 2016.
- [19] Snyder, S. M., Rugino, T. A., Hornig, M., & Stein, M. A., "Integration of an EEG biomarker with a clinician's ADHD evaluation", *Brain and behavior*, Vol. 5(4), e00330, 2015.
- [20] Banaschewski, T., & Brandeis, D., "Annotation: what electrical brain activity tells us about brain function that other techniques cannot tell us—a child psychiatric perspective", *Journal of child Psychology and Psychiatry*, Vol. 48(5), pp. 415-435, 2007.
- [21] Khalifa, M., "Health Analytics Types, Functions and Levels: A Review of Literature", *ICIMTH*, pp. 137-140, 2007.
- [22] Meskó, B., Hetényi, G., & Györfy, Z., "Will artificial intelligence solve the human resource crisis in healthcare?", *BMC health services research*, Vol. 18(1), pp. 1-4, 2018.
- [23] Islam, M. S., Hasan, M. M., Wang, X., & Germack, H. D., "A systematic review on healthcare analytics: application and theoretical perspective of data mining", In *Healthcare*, Vol. 6, No. 2, pp. 54, *Multidisciplinary Digital Publishing Institute*, June, 2018.
- [24] Jollans, L., & Whelan, R., "Neuromarkers for mental disorders: harnessing population neuroscience", *Frontiers in psychiatry*, Vol. 9, pp. 242, 2018.
- [25] Mandal, A. <https://www.news-medical.net/health/What-is-a-Biomarker.aspx>.
- [26] Tenev, A., Markovska-Simoska, S., Kocarev, L., Pop-Jordanov, J., Müller, A., & Candrian, G., "Machine learning approach for classification of ADHD adults", *International Journal of Psychophysiology*, Vol. 93(1), pp. 162-166, 2014.
- [27] Öztoprak, H., Toyçan, M., Alp, Y. K., Arıkan, O., Doğutepe, E., & Karakaş, S., "Machine-based classification of ADHD and nonADHD participants using time/frequency features of event-related neuroelectric activity", *Clinical Neurophysiology*, Vol. 128(12), pp. 2400-2410, 2017.
- [28] Heinrich, H., Hoegl, T., Moll, G. H., & Kratz, O., "A bimodal neurophysiological study of motor control in attention-deficit hyperactivity disorder: a step towards core mechanisms?", *Brain*, Vol. 137(4), pp. 1156-1166, 2014.
- [29] Müller A, Vetsch S, Pershin I, Candrian G, Baschera GM, Kropotov JD, Kasper J, Rehim HA, Eich D., "EEG/ERP-based biomarker/neuroalgorithms in adults with ADHD: Development, reliability, and application in clinical practice", *The World Journal of Biological Psychiatry*, May 7, 2019.

- [30] Raven, J., Court, J. H., "Manual for Raven's progressive matrices and vocabulary Scales", 1991, San Antonio, TX: Harcourt Assessment, 2003, updated 2004.
- [31] Zoubek, L., Charbonnier, S., Lesecq, S., Buguet, A., & Chapotot, F., "Feature selection for sleep/wake stages classification using data driven methods", *Biomedical Signal Processing and Control*, Vol. 2(3), pp. 171-179, 2007.
- [32] Tzanetakis, G., & Cook, P., "Musical genre classification of audio signals", *IEEE Transactions on speech and audio processing*, Vol. 10(5), pp. 293-302, 2002.
- [33] Tsallis, C., Mendes, R., & Plastino, A. R., "The role of constraints within generalized nonextensive statistics", *Physica A: Statistical Mechanics and its Applications*, Vol. 261(3-4), pp. 534-554, 1998.
- [34] Mormann, F., Andrzejak, R. G., Elger, C. E., & Lehnertz, K., "Seizure prediction: the long and winding road", *Brain*, Vol. 130(2), pp. 314-333, 2007.
- [35] Shannon, C. E., "A mathematical theory of communication", *ACM SIGMOBILE mobile computing and communications review*, Vol. 5(1), pp. 3-55, 2001.
- [36] Rényi, A., "On measures of entropy and information. In Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability", Volume 1: Contributions to the Theory of Statistics. The Regents of the University of California, 1961.
- [37] Nai-Jen, H., & Palaniappan, R., "Classification of mental tasks using fixed and adaptive autoregressive models of EEG signals", *In The 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Vol. 1, pp. 507-510, IEEE, September, 2004.
- [38] Bai, J., & Ng, S., "Tests for skewness, kurtosis, and normality for time series data", *Journal of Business & Economic Statistics*, Vol. 23(1), pp. 49-60, 2005.
- [39] Ansari-Asl, K., Chanel, G., & Pun, T., "A channel selection method for EEG classification in emotion assessment based on synchronization likelihood", *In 2007 15th European Signal Processing Conference*, pp. 1241-1245, IEEE, September, 2007.
- [40] Tang, W. C., Lu, S. W., Tsai, C. M., Kao, C. Y., & Lee, H. H., "Harmonic parameters with HHT and wavelet transform for automatic sleep stages scoring", *REM*, Vol. 365, pp. 8-6, 2007.
- [41] Peeters, G., Giordano, B. L., Susini, P., Misdariis, N., & McAdams, S., "The timbre toolbox: Extracting audio descriptors from musical signals", *The Journal of the Acoustical Society of America*, Vol. 130(5), pp. 2902-2916, 2011.
- [42] Kıymık, M. K., Güler, İ., Dizibüyük, A., & Akın, M., "Comparison of STFT and wavelet transform methods in determining epileptic seizure activity in EEG signals for real-time application", *Computers in biology and medicine*, Vol. 35(7), pp. 603-616, 2005.
- [43] Percival, D. B., Walden, A. T., "Wavelet methods for time series analysis", Cambridge university press; 2000.
- [44] Kaiser, A., Aggensteiner, P. M., Baumeister, S., Holz, N. E., Banaschewski, T., & Brandeis, D., "Earlier versus later cognitive event-related potentials (ERPs) in attention-deficit/hyperactivity disorder (ADHD): a meta-analysis", *Neuroscience & Biobehavioral Reviews*, Vol. 112, pp. 117-134, 2020.
- [45] Jenke, R., Peer, A., & Buss, M., "Feature extraction and selection for emotion recognition from EEG", *IEEE Transactions on Affective computing*, Vol. 5(3), pp. 327-339, 2014.
- [46] Park, K. S., Choi, H., Lee, K. J., Lee, J. Y., An, K. O., & Kim, E. J., "Emotion recognition based on the asymmetric left and right activation", *International Journal of Medicine and Medical Sciences*, Vol. 3(6), pp. 201-209, 2011.
- [47] Ververidis, D., & Kotropoulos, C., "Fast and accurate sequential floating forward feature selection with the Bayes classifier applied to speech emotion recognition", *Signal processing*, Vol. 88(12), pp. 2956-2970, 2008.
- [48] Bishop, C. M., & Tipping, M., "Variational relevance vector machines", *arXiv preprint arXiv:1301.3838*, 2013.
- [49] Borra, S., & Di Ciaccio, A., "Measuring the prediction error. A comparison of cross-validation, bootstrap and covariance penalty methods", *Computational statistics & data analysis*, Vol. 54(12), pp. 2976-2989, 2010.
- [50] Fawcett, T., "An introduction to ROC analysis", *Pattern recognition letters*, Vol. 27(8), pp. 861-874, 2006.
- [51] Hajian-Tilaki, K., "Receiver operating characteristic (ROC) curve analysis for medical diagnostic test evaluation", *Caspian journal of internal medicine*, Vol. 4(2), pp. 627, 2013.
- [52] Kropotov JD. Quantitative EEG, event-related potentials and neurotherapy. Academic Press; 2010.
- [53] Thome, J., Ehlis, A. C., Fallgatter, A. J., Krauel, K., Lange, K. W., Riederer, P., & Gerlach, M., "Biomarkers for attention-deficit/hyperactivity disorder (ADHD)", A consensus report of the WFSBP task force on biological markers and the World Federation of ADHD. *The World Journal of Biological Psychiatry*, Vol. 13(5), pp. 379-400, 2012.
- [54] Ogrim, G., Kropotov, J., & Hestad, K., "The QEEG theta/beta ratio in ADHD and normal controls: sensitivity, specificity, and behavioral correlates", *Psychiatry Research*, Vol. 198(3), pp. 482-488, 2012.
- [55] Baijot, S., Cevallos, C., Zarka, D., Leroy, A., Slama, H., Colin, C., & Cheron, G., "EEG dynamics of a go/nogo task in children with ADHD", *Brain sciences*, Vol. 7(12), pp. 167, 2017.
- [56] Yasumura, A., Omori, M., Fukuda, A., Takahashi, J., Yasumura, Y., Nakagawa, E., ... & Inagaki, M., "Applied machine learning method to predict children with ADHD using prefrontal cortex activity: a multicenter study in Japan", *Journal of attention disorders*, Vol. 24(14), pp. 2012-2020, 2020.
- [57] Bzdok, D., & Yeo, B. T., "Inference in the age of big data: Future perspectives on neuroscience", *Neuroimage*, Vol. 155, pp. 549-564, 2017.





# Embedding Knowledge Graph through Triple Base Neural Network and Positive Samples\*

Research Article

Sogol Haghani<sup>1</sup>

Mohammad Reza Keyvanpour<sup>2</sup>

**Abstract:** Representation learning on a knowledge graph aims to capture patterns in the knowledge graph as low-dimensional dense distributed representation vectors in the continuous semantic space, which is a powerful technique for predicting missing links in knowledge bases. The problem of knowledge base completion can be viewed as predicting new triples based on the existing ones. One of the prominent approaches in knowledge base completion is the embedding model. Currently, the majority of existing knowledge graph embedding models cannot deal with unbalanced entities and relations. In this paper, a new embedding model is proposed, with a general solution instead of using the additional corpus. First, a triple-based neural network is presented to maximize the likelihood of the knowledge bases finding a low-dimensional embedding space. Second, two procedures to generate positive triples are proposed. They produce positive triples and add them to the training data. The policies can capture rare triples, and simultaneously remain efficient to compute. Experiments show that the embedded model proposed in this paper has superior performance.

**Keywords:** Knowledge Graphs, Link Prediction, Positive Samples, Embedding Neural Network, Graph Mining

## 1. Introduction

Knowledge bases like Wordnet [1], YAGO [2], or the Google Knowledge Graph are useful resources used in many AI tasks, which present ways to organize, manage, and retrieve all digital knowledge. A knowledge base can be represented as a set of (head, relation, and tail) triples. Any information can reach from the knowledge base through triples or concatenation of them [3, 4]. Although completeness, accuracy, and high quality of data are the parameters that guarantee their advantage of them, they suffer from incompleteness and a lack of reasoning capability [3]. The problem of knowledge base completion can be viewed as predicting new triples based on the existing ones [6].

One of the promising approaches to knowledge base completion is to embed their entities and relations into low-dimensional vector spaces. The methods define a score function and assign a score to the triple [5, 6]. For any unobserved triple, its plausibility can be predicted by using the learned embedding and the score function. The high-value score will assign to the probable triple [5].

Despite the substantial efforts and great successes in the research, the effectiveness of the embedding methods has not been directly compared. They mostly use various pre-training methods to initialize the embedding vector space. It is still unclear that which pre-training method should be

employed, though it has a considerable effect on the results [7, 8]. Another issue is heterogeneous and unbalanced entities and relations in the knowledge base. Heterogeneity may affect overfitting on simple triples or underfitting on rare ones. A simple triple is the one in which its elements appear in most other triples, while rare triples lack their entities and relations of looking most [9]. In Fig.1 triple  $(F, live\_in, E)$  is such a rare triple that the rate of relation *live\_in* is lower than the other, or triple  $(G, father\_of, H)$  is the other kind rare one, which the degree of *H* is low in comparison to *G*. Alternatively, triple  $(F, born\_in, D)$  is such a simple one. Although embedding methods have a strong ability to model knowledge graphs, it remains challenging faced with heterogeneous data [10].

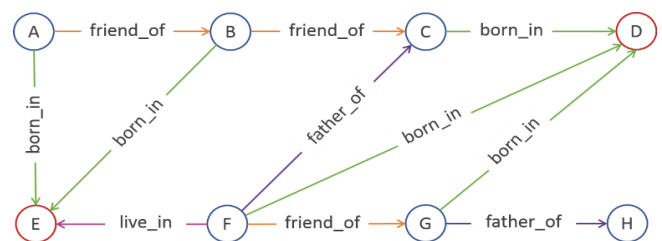


Figure 1. Example of rare and simple triple

The goal of this study is to introduce a novel algorithm that does not require pre-training and can perform and compete while it can deal with unbalanced entities and relations. To that end, two methods are proposed.

First, we propose a new triple-based embedding neural network, to encode the knowledge base to the embedding vector space for entities and relations which maximizes the likelihood of the whole knowledge base. It is a customized, objective function using Stochastic Gradient Descent (SGD) motivated by prior work on natural language processing to the triple structure [11]. The proposed triple-based embedding neural network was used to capture the semantic and syntactic structure of the knowledge base. It takes a knowledge graph as input and produces latent representations for entities and relations. On this subject, we showed that the triple-based embedding neural network used in knowledge base completion obtains proper results in comparison to the state of the arts.

Second, since the embedding models lack in predicting rare triples, two different procedures are introduced to augment the knowledge base to overcome this deficiency. To address this issue, positive triples are generated during the training with a semi-learned embedding vector. Generated triples are added to the training data based on the rate of appearing in previous training data. The rarer triple, the

\* Manuscript received: 22 November 2021, Revised, 07 July 2022; Accepted: 03 October 2022.

<sup>1</sup>. Master, Department of Computer Engineering, Alzahra University, Tehran, Iran.

<sup>2</sup>. Corresponding author. Professor, Department of Computer Engineering, Alzahra University, Tehran, Iran.

Email: Keyvanpour@alzahra.ac.ir

higher the chance of being in the training set. Each procedure uses a specific mechanism in adding positive triples to the training data.

- GNSs (Generate New Samples) generate positive triples after  $\delta$  iteration of learning triple-based embedding neural network, then add them to the training set. The rest of the iterations are worked with the new augmented training set.
- FCSA (Flip Coin Simulated Annealing) decides to generate new triples or use the training sample in the learning process. At the beginning of the process, it rarely generates new triples, and by the time when the embedding vectors learn, it can generate more new triples.

We demonstrate their usefulness by applying them to our triple based neural network. Our extensive experiments on two benchmark datasets show that they achieve superior performance over competitive baselines in two knowledge base completion tasks.

The rest of the paper is structured as follows. Section 2 reviews literature on knowledge base embedding. Section 3 presents our approach. Section 4 presents empirical results. Finally, section 5 includes the conclusion and plan of further work.

## 2. Related works

Various models have been proposed for knowledge graph completion through the link prediction task. Embedding the knowledge graph into a low-dimensional continuous vector space is one of the assuring approaches [12]. Various types of knowledge graph embedding models have been proposed, and they learn the relation between entities using observed triple in the knowledge graph. These models can be classified into three classes: translation-based models, bilinear models, and compositional models [6]. Before proceeding, mathematical notations need to be defined.  $h$ ,  $r$ , and  $t$  denote a head entity, relation, and tail entity, respectively.

The bold letters  $e_h$ ,  $e_r$ , and  $e_t$  denote embeddings of  $h$ ,  $r$ , and  $t$ , respectively, on an embedding space  $\mathbb{R}^d$ .  $E$  and  $R$  represent sets of entities and relations, respectively.

### Translation-based models

The existing translation-based model treats the triple as a relation-specific translation from the head entity to the tail entity. The entity vector obtains the optimal value during the training process by score function, while the relation is regarded as an operator or a translator [5, 12]. Meanwhile, TransE has been introduced as a pioneer in this approach [13]. It is assumed that there is  $e_h + e_r \approx e_t$  equation for each valid triple which assumes that the tail embedding  $e_t$  should be in the neighborhood of  $e_h + e_r$ . TransE is used  $L_2$  to learn embedding vectors. It is not only a simple model but also has a high degree of scalability for modeling complex patterns by embedding dimensions. TransH [10], TransD [9], and TransR [14] are other translation methods. For instance, TransH is a transitional projection. TransD is similar to it, with the difference that it uses the identity matrix of  $d \times k$  size. The dimensionality of the entity and relation vector is considered differently. TransR also uses a rotation transformation for the train. CTransR [14] and TransSparse [9] are an extension of TransR. CTransR considers

correlations under each relation type by clustering diverse head-tail pairs into groups and learning distinct relation vectors for each group. TransSparse focuses on solving the imbalance issues in knowledge graphs, which are ignored by previous translation models. The imbalance means that the number of head entities and that of tail entities in relation could be different.

### Bilinear models

The DistMult [15] is based on a bilinear model where each relation is represented by a diagonal rather than a full matrix. It learns a tensor that is symmetric in the subject and object, while datasets contain mostly non-symmetric triples. ComplEx [12] solves the same issue of DistMult by the idea that multiplication of complex values is not symmetric. ComplEx represents a real-valued tensor  $X \in \mathbb{R}^{N_1 \times N_2 \times N_3}$  as the real part of the sum of  $R$  complex-valued rank one tensors  $u_r^{(1)} \otimes u_r^{(2)} \otimes \bar{u}_r^{(1)}$  where  $r \in \{1, \dots, R\}$  and  $u_r^{(m)} \in \mathbb{C}^{N_m}$

$$f_r(h, t) = \text{Re}(\sum_{r=1}^R u_r^{(1)} \otimes u_r^{(2)} \otimes \bar{u}_r^{(1)}) \quad (6)$$

where  $\bar{u}_r^{(1)}$  is the complex conjugate of  $u_r^{(1)}$ . Bilinear models have more redundancy than translation-based models and so easily become overfitted. Hence, embedding spaces are limited to low-dimensional space. SimpleE [34] are all proved to be fully expressive when embedding dimensions fulfill some requirements. The full expressiveness means these models can express all the ground truth which exists in the data, including complex relations. However, these requirements are hardly fulfilled in practical use. RotatE [35] represents relations as rotations in a complex latent space, with  $h$ ,  $r$ , and  $t$  all belonging to  $\mathbb{C}^d$ . The  $r$  embedding is a rotation vector: in all its elements, the complex component conveys the rotation along that axis, whereas the real component is always equal to 1. The rotation  $r$  is applied to  $h$  by operating an element-wise product (once again noted with  $\odot$  in 1). L1 norm is used for measuring the distance from  $t$ . The authors demonstrate that rotation allows modeling correctly numerous relational patterns, such as symmetry/anti-symmetry, inversion, and composition.

### Compositional models

In the LP field, KG embeddings are usually learned jointly with the weights and biases of the layers; these shared parameters make these models more expressive, but potentially heavier, harder to train, and more prone to overfitting [33]. NTN [16] is one of the most well-known methods in knowledge base completion. The model uses a three-way tensor in its score function. In other words, NTN can replace the standard neural network layer with a three-way tensor layer. Also, using  $\tanh$  for applying the non-linear actions, the score function can be calculated as follows:

$$f_r(h, t) = u_r^T f(e_h^T \underline{W}_r^{[1:k]} e_t + W_{r,1} e_h + W_{r,2} e_t + b_r) \quad (7)$$

where  $\underline{W}_r^{[1:k]} \in \mathbb{R}^{d \times d \times k}$  is a tensor and  $W_{r,1}, W_{r,2} \in \mathbb{R}^{k \times d}$  are weight matrices and  $b_r \in \mathbb{R}^k$  is the bias vector. Despite the fascinating performance, this method is very complicated, and the evaluation results show that representations vectors with the pre-train can reach such a function [17].

HOLE [18] is another method known in this field. This

method has high performance compared to the others. The reason for this function is that it can be applied to a circle of correlation in the score function to represent the space of entities and relations. This method uses a pre-train to create the initial representation space, which causes representation vectors to not have random values at the beginning of the training process and, conversely, have an appropriate initialization.

ConvE [31] performs a global 2D convolution operation on the subject entity and relation embedding vectors after they are reshaped to matrices and concatenated. The obtained feature maps are flattened and transformed through a linear layer, and the inner product is taken with all object entity vectors to generate a score for each triple. Whilst results achieved by ConvE are impressive, the reshaping and concatenating of vectors as well as using 2D convolution on word embeddings is unintuitive. The R-GCN uses a graph convolutional network to obtain an embedding of the triples, then applies DistMult [15] to compute a score for the embeddings.

As pointed out in [8], pre-training is an open question where it is still unclear which pre-training method should be employed. There is no standard, and no priority has been mentioned for it.

### 3. Our approach

In this section, we first propose how the triple-based embedding neural network is worked to represent entities and relations. Second, the detail of generating positive triples and two procedures of how to apply them in learning is provided.

#### 3.1. Triple-Based Embedding Neural Network

Figure 2 shows a perspective of the Triple-based Embedding Neural Network's layers. It consists of three layers. As seen in the figure, the first layer is composed of two parts connected by the weight matrices to the hidden layer. The upper part of the layer is a one-hot vector of the head entity, and the bottom is a one-hot vector of the relation. The hidden layer is a sum of the projection vectors of head and relation. The number of neurons in the last layer is also equal to  $|E|$ , which is equal to the size of the upper part of the first layer. This layer describes the probability of tail with the given of the head and relation. In other words, not only the last layer is not the output but also the embedding vectors are its rows of weight matrices.

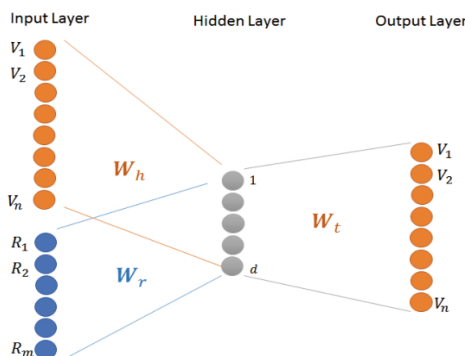


Figure 2. Triple Based Embedding Neural Network

Three weight matrices  $W_h$ ,  $W_r$  and  $W_t$  after training have optimal weights, and each rows of  $W_h^T$ ,  $W_t$  and  $W_r^T$  are a embedding vector for entities  $e_h$ ,  $e_t$  and relation  $e_r$  [11, 13]. The overall process of learning embedded neural network has been presented in algorithm 1.

---

#### Algorithm 1: Triple based Embedding Neural Network

---

*Input:* T: Triples, E: Entities, R: Relations,  $k_{ns}$ : k negative samples

*Output:*  $W_h, W_r, W_t$

*Initializations:*  $W_h$  uniform for each entity as head,  $head \in E$ ,  $W_h$  uniform for each entity as tail,  $tail \in E$ ,  $W_r$  uniform for relation,  $relation \in R$

Create Unigram Table of Negative Triples

foreach n do

  foreach triple do

    Find K negative samples

    Calculate:  $\log(1/(1+e^{-z})) + \log(1/(1+e^z))$

    Update  $W_h, W_r, W_t$ , use SGD.

  end

end

---

The purpose of the Triple-based embedding neural network is to estimate the maximum likelihood of a knowledge base. Accordingly, as shown in algorithm 1 the main loop of learning tries to maximize its likelihood by considering all training triples of the knowledge base. A loss function should minimize the error by considering corrupted triples [3].

It should be noted that the purpose of the method is to learn latent representations, not probable distribution between two entities. Conditional probability  $Pr(t|h, r)$  is considered for triple  $(h, r, t)$ . The goal is to set the parameter  $\theta$  to maximize the probability of the knowledge base (8).

$$\arg \max_{\theta} \prod_{triple \in T} Pr(t|h, r; \theta) \quad (8)$$

$T$  is the list of observed triples or training sets.  $Pr(T = 1|(h, r, t))$  is the probability that the triple  $(h, r, t)$  exists in the training set, or, more precisely, a triple has been observed.

Conversely, the probability of  $Pr(T = 0|(h, r, t)) = 1 - Pr(T = 1|(h, r, t))$  indicates that a triple has not been observed. With these assumptions, the goal is to find the parameters that maximize the likelihood of seeing all the observed triples in the training set:

$$\begin{aligned} & \arg \max_{\theta} \prod_{triple \in T} Pr(T = 1|(h, r, t); \theta) \\ & \approx \arg \max_{\theta} \log \prod_{triple \in T} Pr(T = 1|(h, r, t); \theta) \\ & = \arg \max_{\theta} \sum_{triple \in T} \log Pr(T = 1|(h, r, t); \theta) \end{aligned} \quad (9)$$

The sigmoid function is used to determine the value of  $Pr(T = 1|(h, r, t); \theta)$ , which is defined as:

$$Pr(T = 1|(h, r, t); \theta) = \frac{1}{1+e^{-z}} \quad (10)$$

and it is expected to meet the objective shown in the

formula 11 [11].

$$\arg \max_{\theta} \sum_{triple \in T} \log = \frac{1}{1+e^{-z}} \quad (11)$$

To the triple based embedded neural network structure, the parameter  $z$  is defined as follows:

$$z = (e_h + e_r) \cdot e_t \quad (12)$$

$e_h$ ,  $e_r$ , and  $e_t$  are embedded vectors. They are for *head*, *relation*, and *tail* respectively. These are the rows of  $W_h^T$ ,  $W_r$  and  $W_t^T$  weight matrices. Figure 3 illustrates the explanation of the equation 12 in vector space. According to the cosine similarity, the smaller the angle between  $e_h + e_r$  and  $e_t$ , maximize the dot product [13]. Due to Figure 3, it is desirable that the sum of  $e_h$  and  $e_r$  be parallel with  $e_t$  [11].

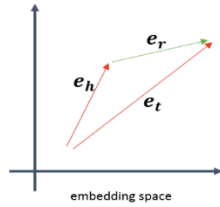


Figure 3. An overview of the relation between  $e_h$ ,  $e_r$ , and  $e_t$  vectors

Due to the structure of Triple-based embedding neural network, corrupted triples are used in learning. The table of corrupted triples with uniform distribution is created []. As shown in 1 the table is generated before the main loop. The relevant question is, can a corrupted generated triple be an observed one. In response, it should be stated that there is no claim to the injection of noise in the learning procedures. Owing to the high dimensionality of entities and their relations, the probability of being a missing triple is low [11]. Finally, gradient descent is used to update the weights. As shown in Algorithm 1, all weight matrices are randomly initialized. By the continuation of the training, optimized weights are obtained.

TransE is one of the popular models on large datasets due to its scalability. Similar to TransE, the time complexity of Triple based neural network is  $O(d)$ , where  $d$  is the size of embedding vectors, it is more efficient than ConvE, NTN, and the neural network models [4].

### 3.2. Generate positive triples

In this section, we start by explaining why to generate positive triples and then describe how to construct them. In the next two sections, the two distinct procedures of how to apply them in the learning model will be illustrated.

Triples are highly heterogeneous in knowledge bases [5]. The diversity is evident both in the type of relation and in the entities. Most of the presented embedding methods are incapable of dealing with such heterogeneity [9]. Therefore, rare entities and relations get an argument. We try to augment rare ones to get a consistent knowledge base. To the best of our knowledge, there has not been an attempt to petition to gain consistent a knowledge base. Inspired by machine vision, data augmentation is used to imbalance classification. Hence, it is being tried to create new images

from existing ones and add to the unbalanced classes [20, 21]. Such a mechanism is needed to balance the knowledge base, though creating new triples from existing ones is not possible in this manner.

To address this problem, we adopted the idea of sequence modeling which is stated that the learning model randomly predicts the next sequence at first, and with learning, the model can correctly predict the following one [22]. In these circumstances, the triple-based embedding neural network is allowed to be learned: the model can generate new triples even as the weight matrices are updating. In other words, after several repetitions, the embedding vectors were found to have reasonably optimized: they were able to predict new instances.

For each entity, all possible triples are created, which it has located as head or tail, and the probability of being a true triple is calculated. Then  $N$  top of the probable triples is nominated to be used in the learning model. These candidates are chosen concerning their rareness: the rarer relation and entity, the more chance to be selected. In other words, a triple has a higher chance of being selected when the head, tail, or relation has been less commonly observed in the training set. The pseudo-code on how to Generate Positive Triples has been shown in Algorithm 2. In the following sections, two strategies named GNSs and FCSSA describe explaining how to use new triples in the learning model.

---

#### Algorithm 2: Generate Positive Triples

---

Generate Positive Triples ( $V, T, R, \lambda, \theta, W_h^T, W_r, W_t^T$ );

Input:  $V$ : entities,  $T$ : Triples (training data),  $R$ : Relations,  $\lambda$ : Size of the selected entity,  $\theta$ : Threshold

Output: Positive Triples

```

for  $i = 0$  to  $\lambda$  do
   $n1 \leftarrow$  Select a node with respect to reverse degree of the node
  foreach relation as  $r$  in Relations do
    foreach node as  $n2$  in entities do
      calculate probability for  $(n1, r, n2)$  and  $(n2, r, n1)$ 
      if probability  $> \theta$  and triple not in Triples then
        temp  $\leftarrow$  triple
      end
    end
  end
  Select  $n$  triples from temp with respect to reverse repetition of relations and add to the Triples
end

```

---

#### A. GNSs

Figure 4 shows the whole process of when to apply GNSs. In the GNSs strategy, the learning procedure stops after  $\delta$  repetitions, and the model starts generating new positive triples. These are created by the updated weights matrices and then add to the training set. Then, the learning model continues training with a new training set. In other words, the new set has the original triples and the new positive triples, which predicts by the semi-learned model. Entities and relations in which there is a higher chance of prediction regarding node reverse degree  $\frac{1}{\deg(entity)}$  and relation repetition  $\frac{1}{|relation|}$  can benefit from the algorithm. The more infrequent relation and entity, the more chance to predict. In other words, a triple has a higher chance of being selected when *head*, *tail* or *relation* has been less commonly observed in the training set. In the opinion of the results of the experiments, selecting a part of the probable triples will increase the performance of the method. According to a thumb rule, the size of the new samples should not be in such a way that eliminates the effect of the original samples.

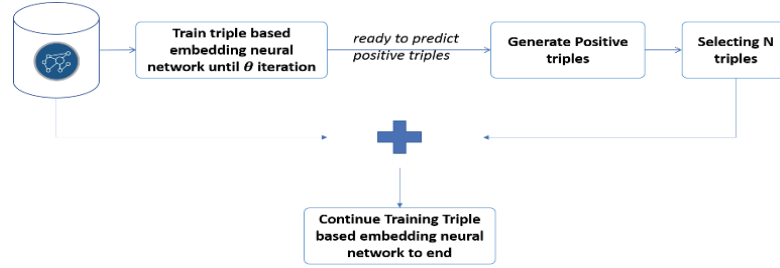


Figure 4. An overview of applying GNSs strategy



Figure 5. An overview of applying FCSA strategy

The time complexity of finding positive triple is  $O(d)$ , and it repeated to  $z$  times where  $z \ll |\text{training data}|$ . As the size of generated positive triples is much less than the training data and they are calculated once in the training, it remains efficient to apply this procedure to the learning.

### B. FCSA

The fundamental idea of FCSA is illustrated in Figure 5. During training, FCSA decides whether a new true triple will be generated or use the original one. In contrast with GNS, the training procedure never pauses. For every sample, we propose to flip a coin and use the true triple or generate the probable one. At the beginning of training, sampling from the model would yield a random triple since the model is not well trained. So, selecting more often, the original samples should help. We thus propose to use a schedule to help the model to generate new triples when it becomes more learned. A sigmoid function is used to decide when new triples can be generated:

$$\epsilon = \frac{1}{1+e^z} \quad (13)$$

$z = \text{total iteration} - \text{current iteration}$

It states that the chances of choosing a new triple are higher at the closure of the learning process and expects the model to sample reasonable triples [22]. If a new sample is selected, FCSA will replace the original. FCSA's goal is to explore newer spaces. As in GNSs, the greater chance is given to probable triples which are more heterogeneous when selecting alternative triples. As the size of training sets remains constant the time complexity of FCSA is  $O(d)$ .

## 4. Experiments

This section proposes an experimental comparison of the proposed method and demonstrates that it can compete with current state-of-the-art methods [3, 18]. The evaluations are based on Wordnet11 and Wordnet18.

### 4.1. Datasets and metrics

To evaluate the proposed method, two datasets Wordnet11 [16] and Wordnet18 [13] were used: both are state-of-the-art

methods. The statistics of these data sets are given in Table 1.

Wordnet11 and Wordnet18 are not only different from each other regarding the size of entities and relations, but also in the structure of the test and the validation set. Each dataset and assessment criteria are described individually in the following sections.

- Wordnet11: Positive and negative samples are indicated in the triple format with a label in test and validation sets. In other words, triples with negative and positive labels are wrong and right triples respectively. Negative triples are constructed from the corruption of positive ones.

### Test methodology

Due to the structure of the dataset, link prediction became a binary classification issue. For each relation, a threshold  $\theta_r$  was determined for evaluation by the validation set. Therefore, the probability of each triple in the test set was compared with its relation threshold: this determined the decision to put a positive or negative label [16].

### Evaluation criteria

Accuracy is a criterion for evaluating this data set, as shown in Equation 14 [23].

$$\text{Accuracy} = \frac{tp+tn}{tp+tn+fp+fn} \quad (14)$$

- Wordnet18: In this dataset, all triples were positive. Therefore, the test methodology and evaluation criteria were based on the triple's rank.

### Test methodology

The rank of the triple was calculated following what is mentioned in [13]. Accordingly, for each examination sample, the tail of the triple was replaced with all entities, and the probability for each of them was calculated. The same procedure is also applied to the head entity. Finally, two lists of all created triples were sorted in descending order by their probability. This procedure is called raw mode, which is composed of all possible triples. Another mode is called filtered, in which all created triples that exist in the training, test, and validation sets are removed except the one that should be evaluated [13].



Table 1. Statistics of the experimental datasets used in this study (and previous works). #Entity is the number of entities, #Relation is the number of relation types, and #Train, #Validation and #Test are the numbers of triples in the training, validation and test sets, respectively

Datasets	#Entity	#Relation	#Train	#Validation	#Test
Wordnet11	38,696	11	112,581	2,609	10,544
Wordnet18	40,943	18	141,442	5,000	5,000

### Evaluation criteria

$MR$ ,  $MRR$ , and  $Hit@k$  are the evaluation criteria used for Wordnet18. The mean of the triple's rank is called the mean rank  $MR$ .  $MR$  is in the range of  $[1, \infty)$ . As  $MR$  gets close to 1, it shows that the proposed method can predict triples at lower ranks [5] which indicates the efficiency of the method.

$$MR = \frac{\sum rank_i}{|N|} \quad (15)$$

The Mean Reciprocal Rank (MRR) is a statistical measure for evaluating each process that presents a list of possible responses to a sample of questions that are arranged with the correct probability. After calculating the rank of all triples, the MRR is calculated as follows:

$$MRR = \frac{1}{|N|} \sum_{i=1}^{|N|} \frac{1}{rank_i} \quad (16)$$

$MRR$  is in the range of  $[0, 1]$ .  $Hit@k$ , like the mean rank criterion, is used to evaluate the prediction of links in the knowledge base. The triple is considered as predicted when the rank is less or equal to  $K$ . Finally, the ratio of predicted triples to the total has been shown as the criterion of  $Hit@k$  (17).

$$Hit@k = \frac{\text{Number of Triples that ranks less or equal than } K}{\text{Number of All Triples}} \quad (17)$$

$Hit@k$  is in the range of  $[0, 1]$ . As the value of this criterion is higher, it shows that most of the triples get a rank lower or equal to  $k$  [18].

### 4.2. Experimental setup

In training the triple-based embedding neural network, two learning rates  $\alpha$  and  $\beta$  are used for entity and relation respectively. The learning rate is validated in  $\{0.001, 0.01, 0.03, 0.05, 0.07, 0.1, 0.15\}$  and the learning rate  $\alpha$  are 0.03 and 0.07 in Wordnet11 and Wordnet18 respectively. Also, the learning rate  $\beta$ , among the values  $\{0.001, 0.005, 0.01, 0.05, 0.1\}$ , 0.005, and 0.001 is validated for Wordnet11 and Wordnet18 respectively. The reason for adding the  $\beta$  learning rate is that the error calculated from the model is added to the relationship weights at a different rate. Since the entity-to-relation ratio is very heterogeneous, two learning parameters are needed to tune the neural network. The appropriate number of negative samples for learning triple base neural network is considered  $kns = 1$  and  $kns = 5$  in Wordnet11 and Wordnet18 [16]. Furthermore, the number of positive triples in the training process in the GNSs strategy is estimated at 500 and 1500 in Wordnet11 and Wordnet18, respectively. By increasing

large numbers of positive triples noise can spread. While, the less samples impact minor effect on results. In GNSs the  $\delta$  is equal to 3/4 total iteration for each data set.

### 4.3. Baselines

This paper compared several state-of-the art relational learning approaches. TransE, TransR, R-GCN, NTN, ComplEx, ConvE and R-GCN comprise our baselines. The results of TransE, R-GCN, TranSparse-DT, and ComplEx are reported from [12] and the results of TransR and NTN from [36], and the rest are from [31]. They are current, state-of-the-art methods and they use the same evaluation protocol.

### 4.4. Results

To specify the effect of each method, four distinct examinations are presented:

1. ENN: Train Triple-based Embedding Neural Network Without Any Strategy;
2. ENN + GNSs: Train Triple-based Embedding Neural Network with GNSs;
3. ENN + FCSEA: Train Triple-based Embedding Neural Network with FCSEA;
4. ENN + GNSs + FCSEA: Train Triple-based Embedding Neural Network with both GNSs and FCSEA strategies

#### 4.4. Results on Wordnet11

The results of the four examinations are provided in Figure 6. To illustrate the different aspects of the neural network's capabilities and proposed strategies, these examinations are presented. We also consider the results by the label of relation, classifying each relation according to its labels. It can be seen from Figure 6. that ENN detects their accuracy less than others, such as the *domain topic* and the *domain region*, by applying the strategies, the accuracy of each has increased about 7%. Also, the relations chart shows that the amount of heterogeneity of the relations causes the strategies to have an effect on the accuracy of each relation. For instance, the *synset domain topic* relation that the ENN estimates its accuracy more than *domain topic* and *domain region*, with applying the strategies the results show less improvement compared the two mentioned. Even in some relations, there is no increase in accuracy. In *member holonym* and *member meronym* relations, the accuracy of the ENN is greater than applying strategies (these relations have the highest accuracy among them). The difference is about 0.5%. This phenomenon shows the decreasing effect of original samples or existence noise in applying strategies. However, it is worth noting that such decreasing is negligible in comparison with the increase of accuracy in other relations.

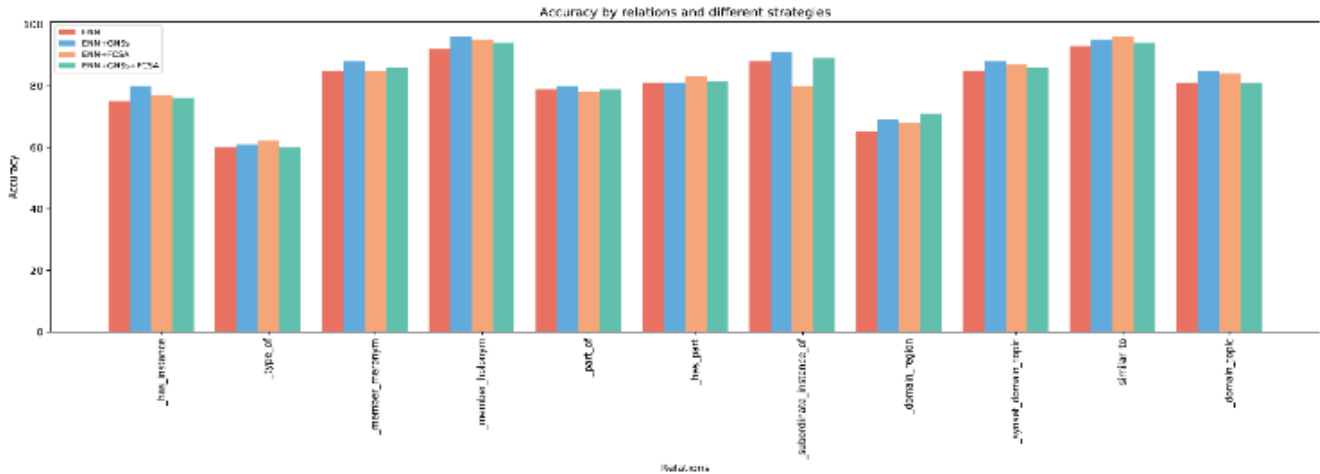


Figure 6. Accuracy of each relation with 4 different tests

The results demonstrate that ENN associated with GNSs is more accurate in comparison with ENN + FCSA. Not only ENN + GNSs consider the training set at the end but also a new training set has been given to the triple-based embedding neural network. Comparing the superiority of the applying FCSA strategy is to reduce the error in high-frequency relations such as *member holonym* and *member meronym*. In this regard, it can be ensured that the accuracy of applying FCSA is not as good a less adverse effect, and is relatively more stable. The ENN has an accuracy of 87.3% and by applying strategies GNSs and FCSA, the accuracy increases about 1 and 2 percent, and this demonstrates that the proposed strategies have a positive effect on the performance of the method. Applying strategies at the same time performs inversely and does not increase accuracy. The cause of this deterioration is related to various aspects. First, by performing FCSA, choosing new samples occurs more when the model is close to the end of the training. If applying FCSA occurs with GNSs, it is probable that some of the new positive samples generated by GNSs will be changed again with FCSA and will be reduced the effect of the GNSs strategy. Also, simultaneously applying these two strategies will cause the original samples at the end of training more faded, and the actual samples do not have their effect [24, 25].

In table 2, all four our distinct examination accuracy with the previously reported results on Wordnet11 are compared. Besides their accuracy, the optimization function that they use for pre-training is shown. Some models have used optimization functions to avoid overfitting. For instance, the NTN method achieves the accuracy of 70.6 without any pre-training, while initializing the embeddings with an unsupervised semantic word vector the accuracy increases to 86.6. Table 2 shows the same result for TransE. Pre-training is used to prevent overfitting, mainly on simple relations. Each model uses distinct methodologies, which makes the comparison not reasonably fair. However, as pointed out by [10] and [34], averaging the pre-trained word vectors for initializing entity vectors is an open problem, and it is not always beneficial since entity names in many domain-specific knowledge bases are not lexically meaningful. However, a comparison has not been made on their performance independently.

According to Table 2, ENN has a high accuracy compared

to methods with the same conditions (without any optimization). It shows that the triple-based embedding neural network is robust to overfitting. Also, applying the GNSs strategy has the highest efficiency among all previous states of the arts. It does not only increase the performance but also it is not domain-specific and does not need external data.

Table 1. Link prediction results on Wordnet11

Methods	Acc%	Opt
NTN [16]	70.06	None
NTN [16]	86.2	Initiate with unsupervised semantic word vectors
TransE(unif) [10]	75.85	None
TransE(bern) [10]	75.82	None
TransE [8]	85.2	Initiate embedding with word2vec
TransH(unif) [10]	77.7	None
TransH (bern) [10]	78.8	None
TranSparse-DT [26]	87.1	None
TransD [9]	86.4	Initiate embedding with the result of TransE
TransR [14]	85.9	Initiate embedding with the result of TransE
CTransR (bern) [14]	85.7	Initiate embedding with the result of TransE
TransG [27]	87.4	Initiate embedding by [28]
ENN	87.3	None
ENN+GNSs	89.4	None
ENN + FCSA	88.2	None
ENN + GNSs + FCSA	87.4	None

**Analysis of Generate Positive triples.** In this section, the effectiveness of the generated positive examples is analyzed. In this regard, some of the positive samples generated in procedure GNSs are given in Table 3. As shown in table 3, the bold tails are also in the test data set. Adding these positive samples and fine-tuning the triple based neural

network with the new training dataset will increase the accuracy and improve the ranks of the test samples.

Table 3. Samples of Generated New Samples

Generated Positive triples in GNSs
(__chromatic_color_1 __has_instance, __pink_4)
(__chromatic_color_1 __has_instance, __red_1)
(__period_1, __has_instance, __bronze_age_1)
(__period_1, __has_instance, __civilisation_2)
(__period_1, __has_instance, __june_1)
(__astronomy_1, __domain_region, __apex_2)
(__astronomy_1, __domain_region, __zenith_1)
(__astronomy_1, __domain_region, __outer_planet_1)
(__family_lobeliaceae_1, member_meronym, __dicot_family_1)
(__japan_2, __has_part, __hondo_1)

Although triples like (*\_\_period\_1, \_\_has\_instance, \_\_june\_1*) and (*\_\_chromatic\_color\_1 \_\_has\_instance, \_\_pink\_4*) not in the test data set, their tails are in the same community with examples like (*\_\_period\_1, \_\_has\_instance, \_\_season\_5*) and (*\_\_chromatic\_color\_1, \_\_has\_instance, \_\_yellow\_2*) respectively, as a result, according to Figure 6, they have affected the performance of the relationship.

#### 4.5. Results on Wordnet18

This section evaluates and represents results on Wordnet18 in two levels. First, results from the four examinations are presented, then a comprehensive analysis of the results of a variety of evaluation criteria with the other state-of-the-art methods is provided. Table 3 shows the result of four different examinations. In this table, the results are displayed in two raw and filtered modes with evaluation criteria.

*MR* is quite sensitive to the outliers. From Table 3, we see that different strategies do not have much effect on the outliers and make significant changes. Unlike Wordnet11, applying both of the strategies has decreased the value of *MR*, which indicates it has advantages in some ways. The lower value of *MR*, the more desirable. One of the matters is to reduce the rank of the outliers. Although the effect is not striking, cannot ignore. The *Hit@k* criterion is a significant benchmark, due to it helps to understand the capability of assigning better ranks to each triple. It is essential to be assured, how many potential triples in the *K* first choices are predicted. Hence, the examinations have been evaluated by  $K = 1, 3, 10$  [18]. As illustrated in Table 3, over more than 90% of samples are predicted with  $k = 10$ . Even in the strictest mode, which  $k = 1$ , more than half of the samples predict as the first prediction option. An assessment with  $k = 3$  is the balance between a flexible and yet rigorous one. However, more than two-thirds of the test cases have

been predicted. The combination of ENN and the GNSs strategy has achieved the best value in all evaluation criteria except *MR* compared to other examinations. It seems that the model has a better performance in increasing the volume of the knowledge base. Although applying the FCSA has a positive effect, does not has a significant performance due to the constant size of the due to the regularization is robust to overfitting and does not need any pre-training and extra optimization functions. It achieves state-of-the-art results on benchmark datasets. Besides, we propose two strategies, GNSs and FCSA, to augment datasets to overcome the heterogeneity of the dataset. In our analysis, we show the performance of applying the knowledge base, which the original triple replaces with the new one. Regarding the application of both strategies on the ENN, the same argument applies to the Wordnet11 dataset. As a conclusion from the experiments in Wordnet18, the number of added triples must be controlled. Obviously, by combining both strategies with the embedded neural network, it cannot allocate very low ranks to triples. On the other hand, it assigns the lower ranks to the outliers [5, 30]. It shows that generated positive triples may be helpful to bring information from other aspects.

In contrast to *MR*, *MRR* is insensitive to outliers. The results also show that increasing the size of the knowledge leads to better *MRR* results. This supports our hypothesis. Table 4 compares the proposed method with other states of arts. In this table, the types of optimizations used are specified to make better comparisons. The HolE and ComplEx implement each of the comparison methods individually and have performed different optimization functions, which have the results reported for TransE being different from one another and the original article. So, it is difficult to determine precisely how much models with pre-training gain over the other ones [12, 18].

ENN has been able to independently handle the structure of the triple, without any pre-training and additional information to perform better. On the *MRR* metric, ENN cannot achieve as good performance as the model with pre-training. There are two noticeable phenomena in the result. First, ENN cannot assign a lower rank to the triples. We believe that this phenomenon is caused by the regularization of the models, even though the principle of it has the potential to represent real knowledge and to achieve knowledge graph completion. Second, it shows that an augmented knowledge base affects weaker but consistent improvement on all metrics.

The proposed method has a significant performance compared to non-pre-trained methods, and its results reflect the evaluation criteria of *MR*, *Hit@10*, and *MRR*. ENN with GNSs and FCSA largely outperforms on *MR* and yields a score of 109 among all methods. Since ENN's Regularization cannot assign a lower rank to most of the triples, it can compete with the state-of-the-art model [31, 32].

Table 4. The comparison of results on Wordnet18 with previous work

Methods	Raw					Filtered				
	MR	MRR	Hit@1	Hit@3	hit@10	MR	MRR	Hit@1	hit@3	Hit@10
ENN	120	0.65	37.42	70.9	85.18	115	0.696	46.24	86.64	93.29
ENN+GNSs	116	0.664	42.98	82.02	91.08	113	0.703	50.54	90.1	94.92
ENN + FCSA	117	0.659	39.92	75.22	90.6	111	0.68	46.8	87.2	93.67
ENN + GNSs + FCSA	114	0.643	38.96	73.66	89.21	109	0.679	47.22	86.12	93.34

Table 5. The comparison of results on Wordnet18 with previous work

Methods	Raw		Filtered			Opt
	MR	hit@10	MR	hit@10	mrr	
TansE [13]	263	75.4	251	89.4	-	None
TransE	-	-	-	94.3	0.495	Using Optimize function [12]
TransH [10]	401	73.0	303	86.7	-	None
NTN [16]	-	-	-	66.1	0.53	None
ManifoldE Sphere [29]	-	81.1	-	94.4	-	Initiate embedding by [28]
ManifoldE Hyperplane [29]	-	81.4	-	93.7	-	Initiate embedding by [28]
TransR [14]	238	79.8	225	92.0	-	Initiate embedding with the result of TransE
TransR [14]	-	-	-	94.9	0.605	using optimize function [8]
CTransR (bern) [14]	231	79.4	218	92.3	-	Initiate embedding with the result of TransE
TransD [9]	224	79.6	212	92.2	-	Initiate embedding with the result of TransE
TransG [27]	483	81.4	470	93.3	-	Initiate embedding by [28]
TranSparse-DT [26]	234	81.4	211	94.3	-	None
HolE [18]	-	-	-	94.9	0.938	using optimize function
ComplEx [12]	-	-	-	94.7	0.941	using optimize function
ConvE [31]	-	-	504	94.2	0.955	Use dropout on the embeddings
R-GCN[32]	-	-	-	96.4	0.819	None
TorusE [8]	-	-	-	95.4	0.947	using optimize function
KE-GCN[32]	-	-	-	-	-	
ENN	120	85.18	115	93.29	0.796	None
ENN+GNSs	116	91.08	113	94.92	0.803	None
ENN + FCSA	117	90.6	111	93.67	0.78	None
ENN + GNSs + FCSA	114	89.21	109	93.34	0.679	None

## 5. Conclusion and future studies

This paper describes a model based on a triple structure for embedding entities and relations via an embedding neural network (ENN). We found that previous methods failed to overfit on infrequent relations. ENN strategies are consistent and reliable. In particular, GNSs and FCSA aren't model dependent, and they can be applied to any models. We believe this observation is essential to assess and prioritize directions for further research on the topic.

In our future work, we will focus on improving the ENN, which needs to utilize loss function. Due to the significant results of the proposed strategies, we will consider other methods for generating new samples and employ them.

## 6. References

- [1] Miller, G. A., "WordNet: a lexical database for English," *Communications of the ACM*, pp. 39-41, 1995.
- [2] Suchanek, F. M., Kasneci, G., and Weikum, G., "Yago: a core of semantic knowledge," 2007.
- [3] Nickel, M., Murphy, K., Tresp, V., and Gabrilovich, E., "A review of relational machine learning for knowledge graphs," *Proceedings of the IEEE*, Vol. 104, No. 1, pp. 11-33, 2015.
- [4] Sadeghi, A., Graux, D., and Lehmann, J., "MDE: Multi Distance Embeddings for Link Prediction in Knowledge Graphs," *arXiv preprint arXiv:1905.10702*, 2019.
- [5] Cai, H., Zheng, V. W., and Chang, K., "A comprehensive survey of graph embedding: Problems, techniques and applications," *IEEE Transactions on Knowledge and Data Engineering*, p. IEEE, 2018.
- [6] Ebisu, T., and Ichise, R., "Toruse: Knowledge graph embedding on a lie group," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [7] Mighan, Sima Naderi, Mohsen Kahani, and Fateme Pourgholamali. "POI Recommendation Based on Heterogeneous Graph Embedding." *2019 9th International Conference on Computer and Knowledge Engineering (ICCCKE)*. IEEE, 2019.
- [8] Nguyen, D. Q., Sirts, K., Qu, L., and Johnson, M., "STransE: a novel embedding model of entities and relationships in knowledge bases," in *Proceedings of NAACL-HLT*, 2016.
- [9] Ji, G., Liu, K., He, S., and Zhao, J., "Knowledge graph completion with adaptive sparse transfer matrix," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.
- [10] Wang, Z., Zhang, J., Feng, J., and Chen, Z., "Knowledge Graph Embedding by Translating on Hyperplanes," in *AAAI*, 2014.
- [11] Li, Zhifei, Hai Liu, Zhaoli Zhang, Tingting Liu, and Neal N. Xiong. "Learning knowledge graph embedding with heterogeneous relation attention networks." *IEEE Transactions on Neural Networks and Learning Systems*, 2021.
- [12] Trouillon, T., Welbl, J., Riedel, S., Gaussier, E., and Bouchard, G., "Complex embeddings for simple link prediction," in *International Conference on Machine Learning*, 2016.
- [13] Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O., "Translating embeddings for modeling multi-relational data," in *Advances in neural information processing systems*, 2013.
- [14] Lin, Y., Liu, Z., Sun, M., Liu, Y., and Zhu, X., "Learning Entity and Relation Embeddings for Knowledge Graph Completion," in *AAAI*, 2015.
- [15] Yang, B., Yih, W.-t., He, X., Gao, J., and Deng, L., "Embedding entities and relations for learning and inference in knowledge bases," *arXiv preprint arXiv:1412.6575*, 2014.


- [16] Socher, R., Chen, D., Manning, C. D., and Ng, A., "Reasoning with neural tensor networks for knowledge base completion," in *Advances in neural information processing systems*, 2013.
- [17] Abedini, F., Menhaj, M. B., and Keyvanpour, M. R., "Neuron Mathematical Model Representation of Neural Tensor Network for RDF Knowledge Base Completion," *Journal of Computer & Robotics*, Vol. 10, No. 1, pp. 1-10, 2017.
- [18] Nickel, M., Rosasco, L., Poggio, T. A., and others, "Holographic Embeddings of Knowledge Graphs," *AAAI*, pp. 1955-1961, 2016.
- [19] Haghani, S., and Keyvanpour, M. R., "moLink: Modeling link representation of knowledge base," in *Information and Knowledge Technology (IKT), 2017 9th International Conference on*, 2018.
- [20] Cao, X., Wei, Y., Wen, F., and Sun, J., "Face alignment by explicit shape regression," *International Journal of Computer Vision*, Vol. 107, No. 2, pp. 177-190, 2014.
- [21] Cui, X., Goel, V., and Kingsbury, B., "Data augmentation for deep neural network acoustic modeling," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, Vol. 23, No. 9, pp. 1469-1477, 2015.
- [22] Bengio, Y., Courville, A., and Vincent, P., "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, Vol. 35, No. 8, pp. 1798-1828, 2013.
- [23] Haghani, S., and Keyvanpour, M. R., "A systemic analysis of link prediction in social network," *Artificial Intelligence Review*, Vol. 52, pp. 1961-1995, 2019.
- [24] Keyvanpour, M., Kholghi, M., and Haghani, S., "Hybrid of Active Learning and Dynamic Self-Training for Data Stream Classification," *International Journal of Information & Communication Technology Research*, Vol. 9, No. 4, 2017.
- [25] Zhu, J., Jia, Y., Xu, J., and others, "Modeling the Correlations of Relations for Knowledge Graph Embedding," *J. Comput. Sci. & Technol*, Vol. 33, No. 2, 2018.
- [26] Chang, L., Zhu, M., Gu, T., Bin, C., Qian, J., and Zhang, J., "Knowledge Graph Embedding by Dynamic Translation," *IEEE Access*, Vol. 5, pp. 20898-20907, 2017.
- [27] Xiao, H., Huang, M., and Zhu, X., "TransG: A generative model for knowledge graph embedding," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 2016.
- [28] Glorot, X., and Bengio, Y., "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010.
- [29] Xiao, H., Huang, M., and Zhu, X., "From one point to a manifold: Knowledge graph embedding for precise link prediction," *arXiv preprint arXiv:1512.04792*, 2015.
- [30] Rosso, Paolo, Dingqi Yang, and Philippe Cudré-Mauroux. "Beyond triplets: hyper-relational knowledge graph embedding for link prediction." *Proceedings of The Web Conference* 2020.
- [31] Dettmers, T., Minervini, P., Stenetorp, P., and Riedel, S., "Convolutional 2d knowledge graph embeddings," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [32] Schlichtkrull, M. a. K. T. N., Bloem, P., van den Berg, R., Titov, I., and Welling, M., "Modeling relational data with graph convolutional networks," in *European Semantic Web Conference*, 2018.
- [33] Yu, Donghan, et al. "Knowledge embedding based graph convolutional network." *Proceedings of the Web Conference 2021*. 2021
- [34] Kazemi, Seyed Mehran, and David Poole. "Simple embedding for link prediction in knowledge graphs." *Advances in neural information processing systems* 31, 2018.
- [35] Sun, Zhiqing, et al. "Rotate: Knowledge graph embedding by relational rotation in complex space." *arXiv preprint arXiv:1902.10197*, 2019.
- [36] Nguyen, Dat Quoc. "An overview of embedding models of entities and relationships for knowledge base completion." *arXiv preprint arXiv:1703.08098*, 2017



# Efficient and Deception Resilient Rumor Detection in Twitter\*

Research Article

Milad Radnejad<sup>1</sup>

Zahra Zojaji<sup>2</sup> 

Behrouz Tork Ladani<sup>3</sup>

**Abstract:** Social networks have become a central part of our lives these days and have real effects on the world's events. However, social networks greatly boost spreading misinformation and rumors that are becoming more and more dangerous each day. As fighting rumors first requires detecting them, several researchers tried to propose novel approaches for automatic early detection of rumors. However, most of them rely on handcrafted content features which makes them prone to deception and threats the adaptability of the model. Furthermore, a great deal of work have concentrated on event-level rumor detection while it faces early detection with serious challenges. There are also deficiencies in proposed methods in terms of time and resource complexity. This study proposes a deep learning approach to automate the detection of rumors on Twitter. The proposed method relies on automatically extracted features through word and sentence embeddings along with profile and network-based features. It then uses Recurrent Neural Networks (RNN) leveraging Gated Recurrent Units (GRU) for detecting the veracity of a tweet. The proposed method also improves time efficiency. The achieved experimental evaluation results on RumorEval2019 dataset demonstrate that the proposed method outperforms other rival models on the same dataset in terms of both performance and time complexity. By the way, the proposed method is more resilient to deception by avoiding the use of handcrafted content features and leveraging features that are out of the control of the user.

**Keywords:** Deception, Deep Learning, Rumor Detection, Social Network, Twitter

## 1. Introduction

The explosive growth of online social media is an evidence for their crucial role in spreading news in the modern society. Nowadays, a large number of users actively engage in producing or propagating news about different trending topics. The convenience of publishing news in online social networks causes also the spreading of misinformation and rumors.

There have been numerous definitions for rumor in the literature, each offering its interpretation. However, the definition provided in [1] seems to be more popular which defines rumor as "a story or statement in general circulation without confirmation or certainty." Another essential research on rumor has been undertaken by [2], which defines three characteristics for rumors: 1) Rumors have a distinct mode of transmission, 2) Rumors always provide information about some particular person, happening, or condition, and 3) Rumors satisfy audiences. The second and the third characteristics refer to the fact that people feel

unsafe in the absence of information, and rumors satisfy them by providing information. Spreading rumors imposes potentially harmful effects on public perception and behavior. One can point to the alleged Russian interference in the 2016 US Presidential Election with the spread of rumors and misinformation through social media [3–5]. Online social networks facilitate rapid propagation of fake news and rumors which thereby greatly amplify the impact of harmful effects.

The most effective and operational approach in rumor detection and debunking today is manual detection, which is done by authoritative centers and websites like Snopes ([www.snopes.com](http://www.snopes.com)) and Politifact ([www.politifact.com](http://www.politifact.com)). However, although this approach seems to be very accurate, it is slow and ineffective with the nature of fast-spreading rumors in social networks. Another approach used these days is automatic detection using artificial intelligence, which leverages machine learning techniques to detect social network rumors. Although the performances of the proposed systems are lower than manual detection, the upside is the constant innovations that are making this approach a likely candidate to replace manual detection. Automatic rumor detection facilitates detecting and preventing rumors in early stages of spreading prior to affecting the public opinion.

Although several researches have been conducted for detecting rumors, the previous methods mostly rely on handcrafted content features. Along with dynamic changing of social network conversations, the content of rumors and the signs of fake or verified news also change. Therefore, feature extraction process should be also dynamic in order to reflex the specifications of the rumors, which is not achieved in the case of developing detection model based on the handcrafted features. Furthermore, handcrafted features make the rumor detection system more susceptible to deception. Employing these features provides more chance to design fake news with appearance similar to verified news. In addition, handcrafted features could bias the prediction model without any explicit improvement in the performance. Again, most of the prior works operate at the event-level, meaning that it can only detect whether a general topic is a rumor or not and cannot decide about the veracity of a single post. Moreover, event-level rumor detection requires an extensive set of messages in each topic which is not available in the first stages of rumor propagation. Hence event-level models are hardly applicable for early detection. Moreover, the scalability of previously presented rumor detection systems is low due to extensive computational complexity. In this work, due to mentioned shortcomings of the previous works, we propose a deep learning approach for detecting rumors on Twitter. The proposed method operates at the

\* Manuscript Received: 06 March 2022; Revised, 22 September 2022. Accepted, 26 September 2022.

<sup>1</sup> MSc. Of Information Technology, Faculty of Computer Engineering, University of Isfahan, Isfahan, Iran,

<sup>2</sup> Corresponding author, Assistant Professor, Faculty of Computer Engineering, University of Isfahan, Isfahan, Iran.

Email: z.zojaji@eng.ui.ac.ir.

<sup>3</sup> Professor, Faculty of Computer Engineering, University of Isfahan, Isfahan, Iran.

tweet level i.e., detecting rumor using a tweet and its responses. While handcrafted features can improve a deep learning system in doing its task, we do not directly use handcrafted content features and let the system extract them by itself through feeding it the raw text of tweets. Instead, word and sentence level embedding for content feature extraction are utilized which makes the model more resilient to user deception, more scalable to social network dynamics, and less susceptible to model biasing. Our insight also is that analyzing social and profile features is a rich information source for developing deception resilient models. This is because these features are independent from the content of the claims and are out of the direct control of the user that was neglected in most of researches. Therefore, we use social and profile features in our model. Lastly, we emphasize on the system's performance, taking the time efficiency and scalability of the system into account.

The main contributions of this work can be summarized as:

1. Avoiding the use of handcrafted content features for better dealing with the dynamic nature of social networks and providing more resiliency against the deception;
2. Using profile and network features that provide the system with valuable information about the users and propagation state;
3. Proposing a new deep learning model with RNN architecture leveraging GRU cells for automatic rumor tweet detection;
4. Detection of rumors at the tweet level in order to facilitate early detection;
5. Emphasis on performance, especially in the training phase, that make the system more scalable in comparison to the similar works.

The proposed method is evaluated and compared using the RumorEval 2019 dataset. The overall performance of the system is first compared to the state of the art methods in terms of Macro F-score. The achieved results show that the proposed method outperforms nearly all similar methods. The experimental results also show the superiority of the proposed method in terms of time efficiency comparing the baseline. Furthermore, some experiments are conducted in order to prove the resiliency of the proposed method against the intended content alteration with the aim of deception. We believe that the proposed rumor detection model has enough capabilities to be applied efficiently in early, tweet-level rumor detection task with remarkable tolerance to deception.

Section 2 of the paper will briefly introduce the concepts that were used in our research. Section 3 will discuss the researches that are similar and related to our work. The problem statement will be described in Section 4. Section 5 is devoted to the proposed method description and its details. The evaluation process and the experimental results and comparisons are presented in Section 6, and finally Section 7 will conclude the paper.

## 2. Background

In this section some preliminary concepts about RNNs and text embedding methods are provided.

### 2.1. Recurrent neural network

As described in [6], recurrent neural networks are a family of neural networks for processing sequential data. These

networks arise from the idea of sharing parameters across different parts of a sequence, making them very efficient and effective in processing sequential data as well as in extraction and learning of sequential features. One of the most exciting features of RNNs is that they can process data of different length as an ability not seen in other types of neural networks which require fixed-size inputs. Another feature of these networks is the concept of memory, which arises from the fact that by sharing parameters and processing the data in sequence, each input will contribute to the model's output in a later stage, which acts as a memory. As also described in [7], one significant shortcoming of conventional RNN cells is that by applying Backpropagation through time, they cannot learn or extract dependencies in a long sequence due to the problem of vanishing or exploding gradients; This can be described as a sort of memory loss, which means conventional RNNs have a very short term memory.

### 2.2. Bidirectional RNNs

RNNs usually process data in a feed-forward approach, meaning at each timestep, the output is calculated using the information from the past, which is the hidden state and the current input [6]. However, in some cases giving the network information about the whole sequence (past and future timesteps) will help solving the problem. Bidirectional RNNs are the combination of two RNNs, one moving forward through time from the start of the sequence and the other moving backward through the time from the end of the sequence. In this way, the output at each timestep is calculated using the information from the past and the future, but more dependent on the data nearest to the current timestep.

### 2.3. Long Short-Term Memory (LSTM)

To combat the memory loss in conventional RNN cells, LSTM was proposed, which defines a pathway for long dependencies, which acts as long-term memory [6]. This pathway can be seen as a cell inside the LSTM cell with its parameters which can add data to the cell and remove unnecessary data when needed. By giving the network, the option of adding data to and removing it from this pathway, the network can memorize essential data in the sequence and forget unnecessary information, which has made LSTM cells a very successful architecture for solving problems, where the input is a sequence.

### 2.4. Gated recurrent unit

Although LSTMs are considered the go-to architecture when dealing with sequential data, they have a crucial shortcoming that arises from its too many parameters. These parameters burden the model, which has to do the standard calculations and with learning the parameters of the LSTM. Moreover, due to the high number of parameters, LSTMs are more susceptible to overfitting, which is a prevalent problem in neural networks and deep learning tasks.

To combat the mentioned shortcomings, the GRU cell was proposed in [8], which is very similar to the LSTM cell but differs in that it combines some parts of the LSTM cell into a unified part and causes a reduction in the number of parameters compared to the LSTM cell. This modification has two benefits: 1) it gives the model less space for overfitting compared to LSTM, and 2) it puts less burden on

the model in terms of calculations that in turn makes it run faster.

### 2.5. Text embedding

In the field of natural language processing, which deals with human language, there is no straightforward way of using words in a neural network. One possible solution might be using one-hot encoding on a lengthy dictionary of words, but this approach has two problems: 1) it wastes memory and processing resources, which can be used elsewhere; (2) by giving the one-hot code based on the alphabetic order, this approach might give close codes to words with different meaning and remote codes to words with similar meaning (the distances are in terms of points in a hyperspace), which might give the model the wrong impression about the similarity of the words.

Word embeddings were proposed to overcome the enumerated challenges by using dense vectors for word representation, reducing memory, and processing the needed resources. The proposed embeddings also put similar words in respect to their concepts and meaning in close vectors in terms of distance, giving the model the ability to understand those words' meanings. Google's word2vec [9] and Stanford's Glove [10] are examples of word embeddings. Although word embeddings are very useful, since the words in a language get their meaning in a sentence, they are not an optimal solution for sentence-level embeddings. One trivial solution for sentence-level embedding is using arithmetic operations to combine the word vectors of a sentence, but this approach can alter the sentence's meaning. In response to these challenges, sentence-level embeddings were proposed to turn a sentence into a dense vector preserving the meaning of sentences in the terms that similar sentences will be given close vectors. Universal sentence encoder (USE) [11] and Fast Sentence Embedding (FSE) are examples of popular sentence-level embeddings.

### 3. Review of related works

There are two different objectives in the automatic rumor detection literature, including *event-level* and *tweet-level* rumor detections. The purpose of event-level approach is to identify the veracity of a general topic related to an event represented by a set of conversations with similar topic. Formally, given an event  $E$  containing conversations  $C_1$  to  $C_n$  (i.e.  $E = \{C_1, \dots, C_n\}$ ) the label  $L(E)$  indicates whether the whole event is rumor or not. In contrast, in tweet-level approach, given a source tweet of each conversation, its responses, and some metadata about the tweets and users, the model should be able to decide whether the source tweet is rumor, non-rumor, or unverified. In fact, in tweet level view, for each conversation  $C$ ,  $L(C)$  specifies the veracity of its single source tweet.

One of the earliest attempts to automate rumor detection was undertaken by [12], in which the effectiveness of different feature categories was studied for identifying rumors. The proposed system can track known rumors but cannot detect new rumors on Twitter.

The first attempt to detect new rumors has been performed in [13], which uses the fact that users respond to rumors by asking questions about them, which was also reported by [14]. The proposed system utilizes conventional machine learning for detection of rumors. However, their system

relies on handcrafted content features. Moreover, it operates at the event-level mode.

Another interesting work on automatic rumor detection based on conventional machine learning on Twitter is [15], which states that although users' stances used in [13] offers an indicator for detecting rumors, but detecting these stances itself is a big challenge. The proposed system leverages a few interesting and lesser-used features; however, it detects unverified stories and does not detect rumors in the context of false information. Moreover, the proposed system operates at the event-level, which was discussed before.

One of the first works in detecting rumors leveraging deep learning techniques is [7], which proposes to use RNN architecture, containing LSTM and GRU cells in detection of rumors. The proposed system considers content data, but it operates at the event level.

Yu et al. [16] have also employed deep learning techniques for rumor detection. They used Convolutional Neural Network (CNN) architecture, reasoning that the proposed system will be more suited due to the fact that RNN architecture is more biased towards the last elements of input while the indicators of rumor are not necessarily in the last elements of the input. They also point out that RNN architecture requires a lengthy input for reliable detection, while many microblog posts are short. Although their work is innovative in that few works are using CNN to detect rumors, but their system, like the ones before, works at the event level.

The closest research to our work is [17], which was later refined and presented as the baseline for RumourEval 2019 [18], and we also consider this research as the baseline for our work. Furthermore, working on this base code and the RumourEval framework makes evaluation of the work more straightforward and clear. In their work, the authors propose a system based on RNN architecture leveraging LSTM cells. One significant contribution of their work is that the proposed system uses some novel features as the feature vector, and it also detects rumors at the tweet level. However, their proposed system uses many handcrafted content features which makes it more susceptible to deception. In addition, it neglects social and profile features which can be potentially used for efficient rumor detection.

Another crucial work similar to our work is [19], which is the winner of SemEval-2019 and the state of the art system. This system is trained with Twitter data and has an exciting innovation that is using fine-tuned word-level embedding specific for the task of rumor detection. Unfortunately, due to the unavailability of the source code and development details of this system, few comments on this work can be made. The proposed system again uses many handcrafted features that makes their system more susceptible to deception. Furthermore, their proposed system relies on some machine learning systems that are still in R&D phase and are considered as open problems in the field of machine learning and natural language processing. To be more specific, this work relies on: 1) a system for detection of parts of sentences like named entities, verbs, etc.; 2) a system for detection of user stances; 3) a system to detect the topic area of the rumor. Hence the overall performance of the proposed method significantly depends on the performance of these underlying systems.

Again, a research related to our work is [15] in which we used some of the features that they have proposed for the task of detection. However, our work is different from [15] in some issues. We are trying to detect false rumors contrary to their work, which can only detect unverified stories. Also we work at the tweet level while their research detects rumors at the event-level.

Another interesting works in automatic rumor detection is [20], which takes a novel approach to detect rumors by leveraging spatial-temporal rumor aspects in social media. Other work is [21] that leverages multi-loss bidirectional RNNs for rumor detection. The work reported in [22] is also another exciting work in rumor detection that has utilized ensemble method for rumor detection.

Table 1 summarizes some of the most important aspects of the more relevant researches to our work. As it can be inferred from Table 1, most of the previous important researches developed rumor detection models in event level, hence they cannot judge about the veracity of each individual tweet. Many papers employed handcrafted features which results in low generalization in detecting new rumor forms and make these systems more prone to deception. Some systems used user's stance as a representative of the crowd wisdom about the specified tweet. Just one research has used the network and user profile features in detecting rumors. As a consequence, the proposed method is designed so that it operates in tweet level, it uses nearly all information sources including content, profile and network features along with users' stances. This is while the proposed method does not inherit the weaknesses of using handcrafted features.

#### 4. Statement of the problem

In this research we aim to automatically detect rumors in Twitter. We attempt to develop a rumor detection system which is resilient to deception. Moreover, the system should detect rumors at tweet-level. The metadata includes profile and network features. Profile features are used to determine the user's credibility, while network features are used to show the state of the rumor propagation in the social network.

While automatic rumor detection has attracted the attention of many researchers over the past few years, a huge

bulk of studies rely on handcrafted features which leads the developed models susceptible to deception. In psychological studies, deception is defined as an intentional and knowing attempt of the writer of a message to create a false deduction or belief in the reader's mind [23, 24]. Humans often do not detect fake contents, in most situations. It has been proved that people can distinguish a truthful statement from a lie with the accuracy of 54% which is just a bit above the random decision [25]. This fact highlights the role of automatic rumor detection under the intended deception process. When a statement is created with the aim of deceiving people, its content appearance should mimic a legitimate statement.

Thus a rumor detection system should detect the veracity of a message regardless of its appearance in order to have resiliency to deception. The appearance of the message can be defined in terms of punctuations, letters cases, image inclusion, and so on. Since most of the handcrafted features used in rumor detection task are describing the message appearance, the resulting models are prone to deception. In this study we attempt to propose a model for rumor detection which can detect rumors efficiently, while neglecting the appearance based features.

Furthermore, event-level approaches require large volume of messages in each topic which is not available in the first stages of rumor propagation. Thus, the aim of this research is developing a tweet-level rumor detection system that will be applicable for early detection. In Twitter, after a user posts a tweet, others can reply to it, and it is also possible to post a reply to a previous reply, and so on. This results in a tree structure of tweets and replies that is called a conversation. Each conversation can be broken up into several branches, each starting from the source tweet and ending at a tree leaf. It is possible to break the conversation into its branches by running a depth-first search on the tree, and each time the algorithm reaches a leaf, the current branch can be extracted by backtracking the steps.

To better understand the concepts of branch and conversation, Figure 1 shows an example, in which a conversation is represented in two branches. One branch containing the source, User1 and User2 posts and another including the source plus User3 and User4 posts.

Table 1. Important aspects of related researches

Research	Operation level	Using handcrafted content features	Using user's stance	Using profile features	Using network features
Yu et al. [1]	Event	Low	Not used	Not used	Not used
Zhao et al. [2]	Event	Medium	Low(only inquires)	Not used	Not used
Ma et al. [3]	Event	Low	Not used	Not used	Not used
Li et al. [4]	Tweet	High	High (all possible stances)	Yes	Yes
Kochkina et al. [5]	Tweet	High	High (all possible stances)	Not used	Not used
Huang et al. [6]	Event	None	Not used	Not used	Yes
Sujana et al. [7]	Event	None	Not used	Not used	Not used
Mouli Madhav Kotteti et al. [8]	Event	None	Not used	Not used	only tweet time stamps
Proposed method	Tweet	None	High (all possible stances)	Yes	Yes

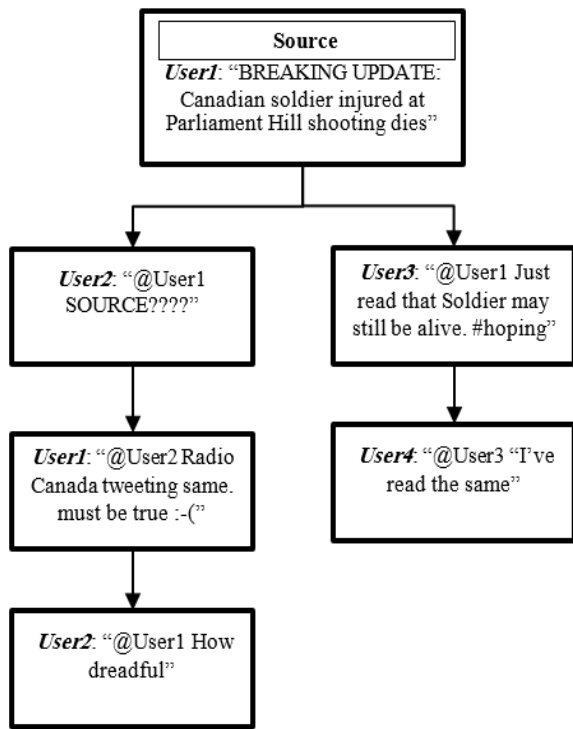


Figure 1. Branches of a conversation

## 5. The proposed method

In this section, we introduce the proposed method in detail. The section begins with the explanation of the system architecture and continues with the detail description of each individual phase.

### 5.1. System architecture

The proposed method is composed of three main phases: preprocessing, feature extraction, and modeling. The overall architecture of the proposed system is depicted in Figure 2. As it is shown in this figure, having a row conversation  $C$ , it is decomposed initially into the set of branches  $\{b_1, b_2, \dots, b_k\}$  where each branch  $b_i$  is composed of the sequence of tweets  $t_{i,1}, t_{i,2}, \dots, t_{i,|b_i|}$  in which  $t_{i,j}$  is the  $i$ th tweet of the  $j$ th branch. Then different features are extracted from the tweet contents and metadata. After that feature vectors can be used to train the model in the training phase and predicting the veracity as label  $L(C)$  in the testing phase.

### 5.2. Preprocessing

Due to the tree's nature, the conversation data processing with a neural network is particularly challenging. To combat this challenge, some researches, such as [5, 9], suggested representation of conversation in terms of its branches. Therefore, the conversation is fed to the network branch by branch. Figure 3 shows the preprocessing phase, in which branches  $b_1$  to  $b_k$  are first extracted and tweets of each branches are then extracted in terms of  $t_{i,1}, \dots, t_{i,|b_i|}$  for each  $b_i$  where  $1 \leq i \leq k$ . In this notation,  $k$  is the number of different branches in  $C$  and  $|b_i|$  is the number of tweets in

branch  $b_i$ .

### 5.3. Feature extraction

Feature extraction phase is illustrated in Figure 4. For each tweet  $t_{i,j}$ , the corresponding network, profile and content features are extracted respectively and concatenated to form the overall tweet feature vector  $\bar{t}_{i,j}$ . The network and profile features are characterizing the social context of the tweet and content features are representing the text of the tweet itself. The feature vector associated with tweets of a branch are then concatenated to form the branch feature vector  $\bar{b}_i = \{\bar{t}_{i,1}, \dots, \bar{t}_{i,|b_i|}\}$  and a conversation is finally represented as a set of branch feature vectors (i.e.  $C = \{\bar{b}_1, \dots, \bar{b}_k\}$ ). One of the innovations of our work is the novel feature set proposed for detection of rumors. Although many of these features have been used before in rumor detection, we have not seen them used together in other previous works. The proposed method also relies on user's stances based on the fact that the users' reactions to rumors are different from non-rumors, which was first pointed out in [10].

An important aspect of the proposed feature set is that we use word and sentence level embeddings for content feature extraction which makes the model more resilient to user deception.

The feature set we use can be described in the following three categories:

1. Profile features (the features of the user who posted the tweet):
  - Number of followers
  - Number of followings
  - Whether the account is verified or not
  - Number of total tweets
2. Network features (the features related to state of the propagation):
  - Number of retweets of the tweet
  - Number of likes of the tweet
  - Whether the tweet is the source or response
  - The stance of each tweet towards the source tweet with values of Supporting, Denying, Querying, and Commenting
3. Content features (the features of the tweet itself by dense vectors leveraging word and sentence level embeddings)
  - Avg2Vec, used in [5, 9], uses Google's word2vec to create a sentence level embedding for tweets by averaging between the word level embedding vectors of the words in the tweet.
  - Universal Sentence Encoder (USE), which is a sentence level embedding also introduced by Google.

It is also worth mentioning that the features used in our work can be categorized in two set:

1. Manually extracted features containing profile and network features;
2. Automatically extracted features containing content features.

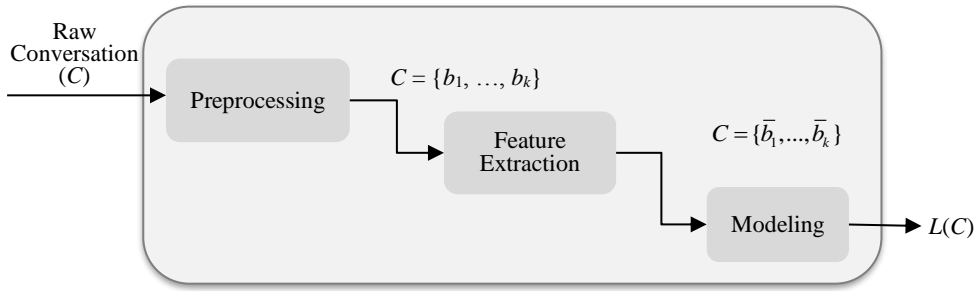


Figure 2. System architecture

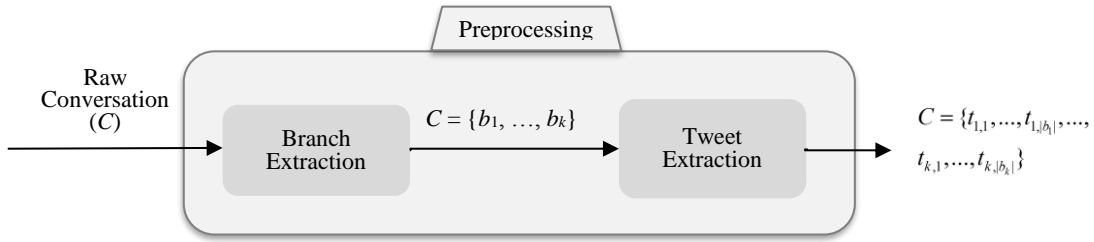


Figure 3. Preprocessing phase

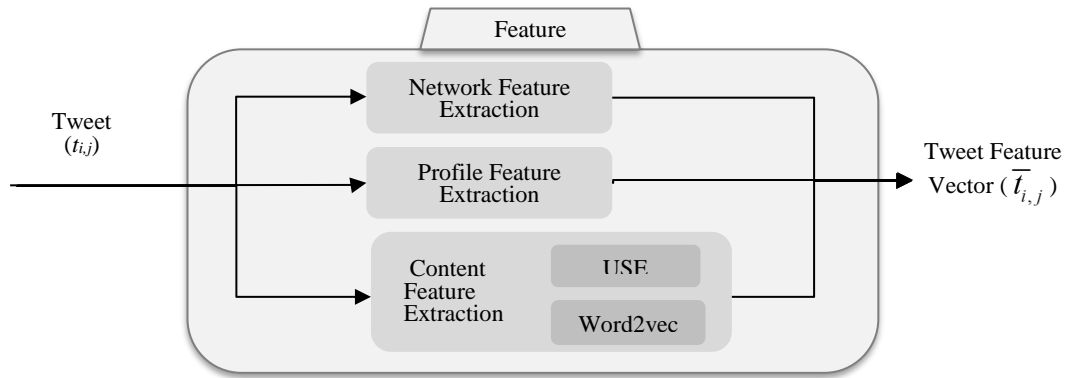


Figure 4. Feature extraction phase

## 6. Modeling

The core of the proposed system is the modeling phase that is comprised from training and testing phases. In training phase, the branch feature vectors along with the source tweet label are feed to RNN for learning. After training, the learnt model can be deployed in automatic rumor detection system as it is demonstrated in Figure 5. In this phase, given a conversation feature vector, the feature vector of each branch is extracted and inputted to RNN model. After predicting the corresponding labels for individual branches, a voting module is used to determine the final label for the conversation as the majority label predicted by its branches. We employed RNN deep learning architecture because the nature of the input data in the underlying system is a sequence. It means that we want the learning system to recognize the patterns and relations between consecutive words, sentences and tweets in processing a conversation. Since RNN is memory-based architecture and learns sequences well, it is appropriate for our purpose. Furthermore, RNNs support learning sequences with variant lengths which is the case in rumor detection systems for branches. Since the tweet branches may form as long sequences, memory based unites such as LSTM and GRU are needed for learning these sequences. Using GRUs is

more preferable because of their speed and efficiency and also to give the model less space for overfitting, which contributes to the overall model performance. We also leverage bidirectional GRUs for two reasons: 1) giving the model more information at each time step; 2) reducing the model's bias towards the end of sequence by processing the sequence from both directions.

The detail architecture of RNN units are revealed in Figure 6. The model is comprised of one bidirectional GRU Layer, and the output of this layer is passed to two dense layers with ELU (exponential linear units) function as their activation [26]. It is worth mentioning that before each layer, the input of that layer is normalized with the batch normalization layer. This has two effects:

1. The data is scaled and the training phase's noise is reduced, where in turn makes the training phase faster and more stable;
2. By feeding data in different batches, it has a slight regularization effect on the model, which reduces the chance of overfitting.

For improving the generalization of the network and avoid overfitting, L2-regularization mechanisms are adopted. The complexity of the network and subsequently the overfitting issue are controlled in this way.



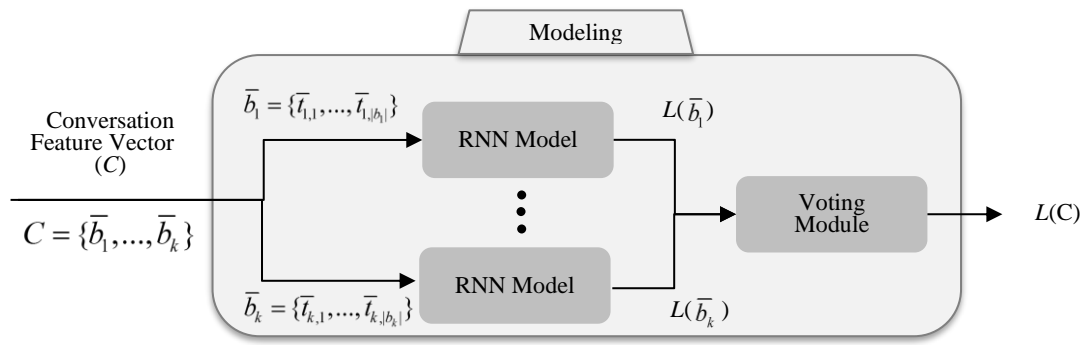


Figure 5. Modeling phase

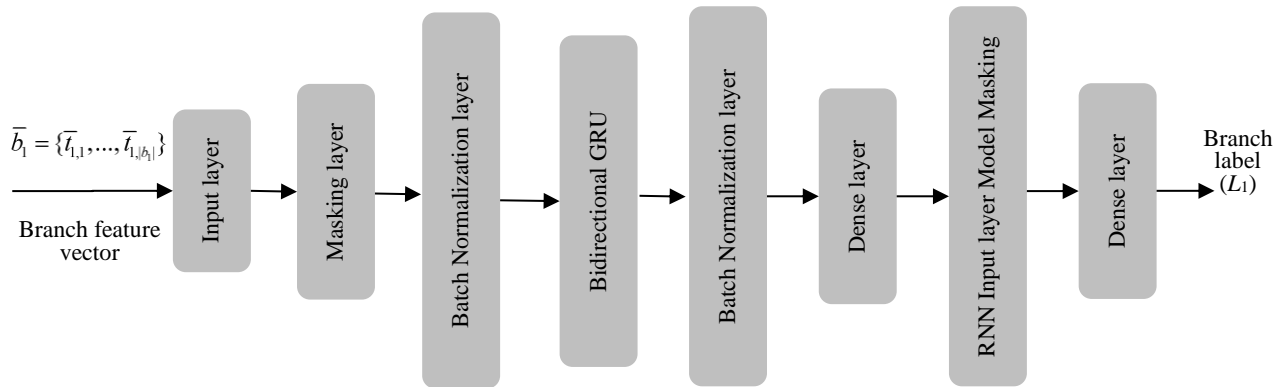


Figure 6. The architecture of the RNN unit

Due to social networks' dynamic nature, the rumor detection model should be adapted and retrained many times after deployment. To achieve this goal, we have leveraged CuDNN libraries in designing the model in order to improve the time complexity in the training phase. The use of CuDNN has one big downside: the loss of some layer features like dropout in GRU layers, but it helps a lot in the model's generalization. Therefore, there is a tradeoff between the time complexity and the generalization and hence we took the middle way through utilizing CuDNN.

## 7. Experimental evaluations and results

In this section after describing the evaluation configurations including the dataset specification and the system setup, the proposed rumor detection system is evaluated in terms of macro F-Score, time efficiency and deception resiliency. The experimental results are also compared to the results of the state of the art methods.

### 7.1. Dataset

The dataset used in this research is associated with RumorEval 2019 competitions which is a refined and updated version of the PHEME dataset [27]. This dataset is comprised of conversations categorized into topics, each topic containing conversations with one of the below labels:

1. True: Conversations that are spreading verified information;
2. False: Conversations that are spreading rumors;
3. Unverified: Conversations that are spreading unverified information that was neither verified nor denied up to the time of their retrieval.

Table 2 shows the distribution of conversations and

branches between the training, development, and testing sets, while Table 3 and Table 4 show the distribution of labels in conversations and branches in different sets. Note that the development dataset in RumorEval context is equivalent to validation dataset known in machine learning literature.

Table 2. Distribution of conversations and branches

	Train	Development	Test
Conversation	297	28	56
Branch	3245	768	1010

Table 3. Label distribution in conversations

	Train	Development	Test
True	137	8	22
False	62	12	30
Unverified	98	8	4

Table 4. Label distribution in branches

	Train	Development	Test
True	1470	124	341
False	549	514	558
Unverified	1226	130	111

### 7.2. Setup

All of the experiments were run on a single system with the same hardware setting for all of them. Table 5 shows the system details.

Table 5. System setup

Processor	Intel Core i7 6700HQ
RAM	16GB DDR4 2133MHz
GPU	Nvidia GeForce GTX 980M

One of the most important parts of every deep learning model is the hyper parameters setting presented in Table 6. Hyper parameters include parameters that define the model and cannot be learnt by the model during the training phase. In order to find the optimal hyper parameters, we leveraged Tree-structured Parzen Estimator (TPE) algorithm [11] implemented in python library called Hyperopt which helps automating some of the tasks in hyper parameter search and model tuning. TPE is a sequential model-based optimization approach, which sequentially estimates the conditional probability density function of the objective function based on hyperparameters. In each iteration, the next set of hyperparameters are configured based on their evaluation on the estimated probability model and the model is refined accordingly. Sequential model-based optimization is a formalization of Bayesian optimization which is more efficient than random or grid search in finding the best set of hyperparameters [11]. Table 6 shows the best hyper parameters found by TPE for the proposed system.

Table 6. Hyper parameter setting

Hyper Parameter	Value
Number of GRU Layers	1
Number of GRU Units	400
Number of hidden dense layers	2
Number of dense units in 1 <sup>st</sup> dense layer	600
Number of dense units in 2 <sup>nd</sup> dense layer	400
Training Steps	50
L2 Regularization Parameter in 1 <sup>st</sup> dense layer	1e-4
L2 Regularization Parameter in 2 <sup>nd</sup> dense layer	1e-4
L2 Regularization Parameter in output layer	1e-6
L2 Regularization Parameter in GRU layer	1e-6
Minibatch size	64
Optimization Algorithm	Adam

### 7.3. Overall performance

Table 7 shows the results of evaluation metrics of the proposed method as well as those in [5, 9] as the baseline. The performance is measured in terms of precision, recall, and F1-score. It can be deduced from the table that our model outperforms the baseline in the overall metric used by the RumourEval 2019 competitions (i.e., Macro-F1 Avg.). A more detailed look shows that the proposed method outperforms the baseline in the rue class but slightly lags behind it in the other classes. It is due to low false negative rate of the proposed method which is a critical necessity of a rumor detection system. The performance of the baseline can be attributed to many features, but as we will show later, this gives their model a significant disadvantage regarding resilience to deception.

The results of all RumorEval 2019 participants can be found in [9]. As it can be inferred from the table, the overall performance of the proposed method is better than other models. There are also some works like [4] that uses some auxiliary data for training. Utilizing auxiliary dataset gives the model some advantages and not only makes the comparison a little unfair, but also we believe it threatens the scalability of the method. When the model is trained and evaluated based on the auxiliary datasets, its performance is not guaranteed for rumor detection in other environments in which this data volume is not available.

Table 7. Comparison to the baseline

	Class	Precision	Recall	F1
Baseline [5, 9]	True	-	-	0.31
	False	-	-	0.53
	Unverified	-	-	0.17
	Macro Avg.	-	-	0.33
Proposed method	True	0.85	0.37	0.51
	False	0.48	0.45	0.47
	Unverified	0.05	0.25	0.08
	Macro Avg.	0.46	0.36	0.35

Table 8 shows the performance comparison of the proposed method with the most successful related models, which operate on RumorEval 2019 dataset.

Table 8. Comparison to other models

Model Name	Macro-F1 score
Baseline [5, 9]	0.33
VANTA and Aono[12]	0.32
WeST (CLEARumor) [13]	0.28
GWU NLP LAB [14]	0.26
BLCU NLP [15]	0.25
FINKI NLP (reported in [9])	0.33
EventAI [4]	0.58
Proposed method	<b>0.35</b>

### 7.4. Resilience to deception

Since many rumors are created with the aim of user deception, the appearance of the claim is designed to mimic a legitimated news post. A successful rumor detection system should not be sensitive to simple apparent signs. In the proposed method, we tried to develop a model that is resilient to these changes. To evaluate the models' resilience to deception, we propose changes to the tweet text, keeping in mind that it is entirely in the user's control and can be changed easily without changing the tweet's overall meaning. The applied changes, enumerated below, are minimal and do not affect the meaning of the text:

1. Removing the periods or adding one if does not exists any;
2. Removing question marks or adding one if does not exists any;
3. Removing exclamation mark or adding one if does not

exists any;

4. Removing pictures or adding one if does not exists any;
5. Changing the capital ratio of the characters to a random value.

We assume that these changes can simulate the changes that a rumor creator made intentionally in the rumor content so that it looks like a normal verified claim. These changes are chosen regarding the experiences reported in [5, 9], which is the only model that shared its details and code. After applying the changes on the test set, both models trained on the original training set (the proposed model and the baseline model) were rerun on the modified test set, and the results were compared to the original run. Since our model ignores all the mentioned handcrafted content features due to the use of embeddings for extracting features from the text, these changes do not affect the model's performance. In contrast, the predicted labels in works reported in [5, 9] were changed in 34% of the conversations after applying the modifications. It can be inferred from the experiment that even simple text changes can easily mislead the baseline model.

Since the described issue arises due to the employment of handcrafted features, it seems that other researches that model the rumor based on these features (e.g., [4]) also suffer from the similar weaknesses. In fact, current experiment compares the resiliency of two categories of approaches to deception, the models based on automatic feature extracted and the models based on handcrafted features. To this end, the proposed method and the baseline are selected as representatives for these two categories of approaches, in the absence of source code of other related methods.

This test shows the downside of handcrafted features, especially for content features, since they are in the control of the user and can be changed easily. For that reason, all the content features used in the proposed method are 1) the ones that are determined by the network and are not controlled by the user, or 2) the ones that are extracted using methods like embedding that focus on the meaning instead of the looks which minimizes the user's influence on the model.

### 7.5. Time efficiency

Table 9 shows the training time of the proposed method and the baseline model. It can be seen that our model outperforms the baseline significantly in training time, giving it a valuable advantage for deployment. This means it saves much time in training, which leads to savings in resources and capital making it more suitable for deployment.

Although, as discussed before, our model cannot use dropout in the GRU layer, which can give it a significant advantage in generalization, we showed that it outperforms the baseline while being much faster in training.

Table 9. Time efficiency

Model	Training Time (seconds)
Baseline [5, 9]	2406.13
Proposed method	40.25

### 7.6. The role of profile and network features

Another exciting aspect that needs discussing is the proposed feature set. Regarding content features, we have already shown that using embeddings instead of handcrafted features

gives our model a significant advantage regarding resilience to deception. Regarding other features, it can be deduced that all of the profile and network features are a part of the social network and they are out of the control of the user, especially for trending topics, and one or a group of users cannot meaningfully change them to mislead the model.

We can also show that the proposed non-content features are needed to achieve the results shown in Table 7. Table 10 compares the model's performance using the proposed feature set to the model using only the content features. It shows that the full feature set outperforms the content features, which in turn shows that network and profile features provide essential information for rumor detection.

Another important aspect of our model is the use of GRU cells instead of LSTM. Although LSTMs are more common in RNN architecture, as discussed before, the use of GRU leads to reducing the training time and increasing the generalization.

Table 10. The role of different feature sets

Feature Set	Macro-F1 Avg.
Content Features only	0.31
Full Feature Set	0.35

Table 11 compares the proposed method with the same model with LSTMs instead of GRUs. It can be seen that the GRU model slightly outperforms the LSTM with fewer parameters and much faster run time.

Table 11. The comparison of GRU and LSTM unites

Model	Macro-F1
LSTM based model	0.34
Proposed method (GRU)	0.35

## 8. Conclusion

While a considerable research effort has been done recently to develop automatic rumor detection models, most of prior approaches have had the problem of relying on handcrafted features. Using these features make the model more susceptible to deception and reduces the scalability of the system. Moreover, a great deal of work is devoted to event level rumor detection which is not applicable for early detection and prevention in real world. This research proposed a rumor detection system based on RNN model and GRU cells for specifying the veracity of tweets in Twitter network. One of the most important innovations of this research is a novel feature set that avoids the extraction of handcrafted content features and uses network and profile features that are out of users' control. Considering these features makes the model more resilient against deception. We focused on efficiency and scalability, especially in the training phase, keeping in mind that social networks' dynamic nature requires the model to be retrained many times to adapt to the users and network behavioral changes, making our model more suitable for deployment.

A number of experiments were conducted to analyze the effectiveness of the proposed rumor detection system. Experimental results show that the proposed method outperforms most similar research in terms of macro F-score.

It also revealed that the proposed system is less prone to deception. Furthermore, the results indicate the superiority of the proposed method comparing the baseline in terms of time efficiency. Consequently, the proposed rumor detection system is suitable for being applied efficiently in early tweet-level rumor detection task with remarkable tolerance to deception.

As the future work in our research direction, we tend to use pertained pre-trained contextual deep neural networks for both content embedding and tweet classification tasks in order to improve the overall performance.

## 9. References

- [1] Yu, F., Liu, Q., Wu, S., et al., "A Convolutional Approach for Misinformation Identification", In: *IJCAI International Joint Conference on Artificial Intelligence*, pp. 3901-3907, 2017.
- [2] Zhao, Z., Resnick, P., Mei, Q., "Enquiring minds: Early detection of rumors in social media from enquiry posts", In: *WWW 2015 - Proceedings of the 24th International Conference on World Wide Web*, pp. 1395-1405, 2015.
- [3] Ma, J., Gao, W., Mitra, P., et al., "Detecting rumors from microblogs with recurrent neural networks", In: *IJCAI International Joint Conference on Artificial Intelligence*, pp. 3818-3824, 2016.
- [4] Li, Q., Zhang, Q., Si, L., "eventAI at SemEval-2019 Task 7: Rumor Detection on Social Media by Exploiting Content", User Credibility and Propagation Information, 2019.
- [5] Kochkina, E., Liakata, M., Augenstein, I., Turing at SemEval-2017 Task 8: Sequential Approach to Rumour Stance Classification with Branch-LSTM, 2018.
- [6] Huang, Q., Zhou, C., Wu, J., et al., "Deep spatial-temporal structure learning for rumor detection on Twitter. *Neural Comput Appl*", <https://doi.org/10.1007/s00521-020-05236-4>, 2020.
- [7] Sujana, Y., Li, J., Kao, H-Y., "Rumor Detection on {T}witter Using Multiloss Hierarchical {B}i{LSTM} with an Attenuation Factor", *Aacl*, 2020.
- [8] Kotteti, C. M. M., Dong, X., Qian, L., "Ensemble deep learning on time-series representation of tweets for rumor detection in social media", *Appl Sci* 10: <https://doi.org/10.3390/app10217541>, 2020.
- [9] Gorrell, G., Kochkina, E., Liakata, M., et al., "SemEval-2019 Task 7: RumourEval", Determining Rumour Veracity and Support for Rumours, 2019.
- [10] Mendoza, M., Poblete, B., Castillo, C., Twitter under crisis: Can we trust what we RT? In: *SOMA 2010 - Proceedings of the 1st Workshop on Social Media Analytics*, 2010.
- [11] Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B., "Algorithms for hyper-parameter optimization", In: *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011*, NIPS, 2011.
- [12] Vanta, T., Aono, M., "Stance Classification and Rumor Analysis: Using New Dialog-Act Features and Augmenting Input Tweets", In: *2020 7th International Conference on Advance Informatics: Concepts, Theory and Applications (ICAICTA)*. IEEE, pp 1–6, 2020.
- [13] Baris, I., Schmelzeisen, L., Staab, S., CLEARumor at SemEval-2019 Task 7: ConvoLving ELMo Against Rumors, 2019.
- [14] Hamidian, S., Diab, M., GWU NLP at SemEval-2019 Task 7: Hybrid Pipeline for Rumour Veracity and Stance Classification on Social Media, 2019.
- [15] Yang, R., Xie, W., Liu, C., Yu, D., BLCU\_NLP at SemEval-2019 Task 7: An Inference Chain-based GPT Model for Rumour Evaluation, 2019.

# A Deep Neural Network Architecture for Intrusion Detection in Software-Defined Networks\*

Research Article

Somayeh Jafari Horestani<sup>1</sup>

Somayeh Soltani<sup>2</sup> 

Seyed Amin Hosseini Seno<sup>3</sup>

**Abstract:** For more comprehensive security of a computer network as well as the use of firewall and anti-virus security equipment, intrusion detection systems (IDSs) are needed to detect the malicious activity of intruders. Therefore, the introduction of a high-precision intrusion detection system is critical for the network. Generally, the general framework of the proposed intrusion detection models is the use of text classification, and today deep neural networks (DNNs) are one of the top classifiers. A variety of DNN-based intrusion detection models have been proposed for software-defined networks (SDNs); however, these methods often report performance metrics solely on one well-known dataset. In this paper, we present a DNN-based IDS model with a 12-layer arrangement which works well on three datasets, namely, NSL-KDD, KDD99, and UNSW-NB15. The layered layout of the proposed model is considered the same for all the three datasets, which is one of the strengths of the proposed model. To evaluate the proposed solution, six other DNN-based IDS models have been designed. The values of the evaluation metrics, including accuracy, precision, recall, F-measure, and loss function, show the superiority of the proposed model over these six models. In addition, the proposed model is compared with several recent articles in this field, and the superiority of the proposed solution is shown.

**Keywords:** Intrusion Detection, Software-defined Network, Deep Learning, Network Security

## 1. Introduction

In computer systems and networks, the attackers exploit security vulnerabilities to attack the network; therefore, there is a need for some methods to detect intrusions into a computer system or network. An intrusion detection system (IDS) is the software or hardware that detects and reacts to intrusions. An IDS prevents illegal access and tampering with the resources of a computer system or network [1-3]. Generally, the IDS monitors the activities of the host computer or the entire network and reports the violations of management and security policies to the network administrator [4-6].

With the growing use of the Internet, network traffic is becoming increasingly complex, and the challenge is becoming more difficult for IDS to detect attacks or anomalies more accurately and quickly. Therefore, researchers leverage machine learning techniques to improve the capability of IDSs.

In the category of machine learning, artificial neural network (ANN) is one of the most widely used models. It is

a computational technique widely used in data processing, pattern recognition, and information classification. Deep learning, a subset of machine learning, attempts to extract high level features from the raw input using several hidden layers. Deep neural networks are used in the design of IDSs for software-defined networks (SDNs).

In recent years, several approaches for intrusion detection have been proposed using machine learning techniques; however, each of the methods has its challenges and problems. For example, most studies have reported good accuracy rates, while they have not reported other metrics such as precision or recall. Some methods have reported relatively low values for these performance measures. Another weakness of these methods is that they work only on one dataset and do not evaluate their methods on larger and newer datasets. On the other hand, some studies have compared their methods with only simple classifiers. However, it is clear that this kind of comparison does not have the necessary quality. In this paper, we offer an intrusion detection method for software-based networks using deep neural networks; the proposed method achieves high performance on several datasets.

The contributions of this work can be summarized as follows:

- It provides a comprehensive and complete classification (Research Tree) in the field of intrusion detection systems.
- It follows a deep learning approach to IDS using deep neural networks in software-defined networks.
- It provides seven neural network-based IDS models and evaluates them on three datasets, namely NSL-KDD, KDD99, and UNSW-NB15. The best model, which has the best accuracy, precision, recall, and F-measure values on all datasets, is then introduced.
- One of the strengths of this solution is that the layered layouts of the proposed models are the same for all three datasets.

The paper then presents the theoretical background and research motivation, discusses the proposed model, evaluates the proposed model, and finally concludes the work.

## 2. Research background

In general, intrusion detection systems can be categorized in terms of various aspects, such as detection method (or analysis technique), type of architecture, how to respond and react to intrusion, information source, and many others [7-12]. For example, intrusion detection systems can be divided into two types of continuous monitoring and periodic

\* Manuscript received: 13 March 2022; Revised, 01 July 2022, Accepted, 01 August 2022.

<sup>1</sup> MSc, Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran.

<sup>2</sup> Corresponding Author, PhD, Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran.

Email: somayeh.soltani@mail.um.ac.ir

<sup>3</sup> Associate Professor, Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran.

analysis in terms of continuity [13-15]. They can also be divided into active and passive responses [16-19].

Chalapathy and Chawla [7] categorized the deep learning-based anomaly detection techniques using three criteria: application, type of anomaly, and type of model. Then they defined nine applications, that is, fraud detection, cyber intrusion detection, medical anomaly detection, sensor network anomaly detection, video surveillance, IoT big data anomaly detection, log anomaly detection, and industrial damage detection. They defined three types of anomalies: collective, contextual, and point. Moreover, they considered four types of detection models: unsupervised, semi-supervised, hybrid, and one-class neural networks.

Kwon *et al.* [9] classified the anomaly-based IDSs into two groups: programmed and self-learning. Then they classified the programmed IDSs into two categories of simple-rule and statistical-based, and they categorized the self-learning IDSs into four categories: cognition-based, computation-intelligence, data mining, and machine learning. In the next step, they classified the machine learning-based IDSs into six groups: Bayesian network, genetic algorithm, fuzzy logic, artificial neural network (ANN), supervised vector machine (SVM), and outlier detection. Furthermore, they defined two types of ANNs: supervised and unsupervised. The supervised ANN IDSs can be free-forward ANN or recurrent ANN. The unsupervised methods include deep learning, adaptive resonance theory, and self-organizing maps. Finally, the deep learning methods include AutoEncoder, sum-product network, recurrent neural network (RNN), Boltzmann machine (BM), convolutional neural network (CNN), and deep neural network (DNN).

Lee *et al.* [18] categorized deep learning-based IDS schemes into nine classes: AutoEncoder-based, RBM-based, DBN-based, DNN-based, CNN-based, GAN-based, LSTM-based, RNN-based, and hybrid. They then classified the AutoEncoder-based schemes into six groups: Stacked

AutoEncoder, Denoising AutoEncoder, NonSymmetric AutoEncoder, Sparse AutoEncoder, Variational AutoEncoder, and Convolutional AutoEncoder. They also defined several hybrid schemes: AE+CNN, AE+DBN, AE+DNN, AE+GAN, AE+LSTM, CNN+LSTM, CNN+RNN, and DNN+RNN.

Having reviewed various articles in the field of intrusion detection systems, we categorized these systems in different ways. In terms of continuity, we classified intrusion detection systems into two categories: continuous monitoring and periodic analysis. Concerning reaction to influence, we divided these systems into two groups: active response and passive response. Regarding the architecture, we divided the IDSs into two groups, centralized and distributed. In addition, we defined two types of real-time or offline forecasting.

In terms of the knowledge base, we considered three classes: Boltzmann machine, descriptive languages, and expert systems. We classified the IDSs into three groups: one variable, multivariate, and time series model. Moreover, the IDS systems are categorized into two classes: anomaly-based and signature-based. We considered three signature-based techniques: data mining, state transition, and expert systems.

Anomaly-based techniques are divided into two groups: self-learning, and programming. The self-learning techniques are cognition-based and relate to computation intelligence, data mining, or machine learning. The machine learning techniques can be semi-supervised, supervised, unsupervised, or reinforcement learning. Each of these techniques has so many subcategories.

We summarize various categorizations in a tree named Research Tree in the field of intrusion detection systems. Figure 1. shows the comprehensive classification tree.

In the following, we categorize previous research works into two main groups in terms of the model architecture: 1) works done on shallow architectures, 2) works done on deep architectures.

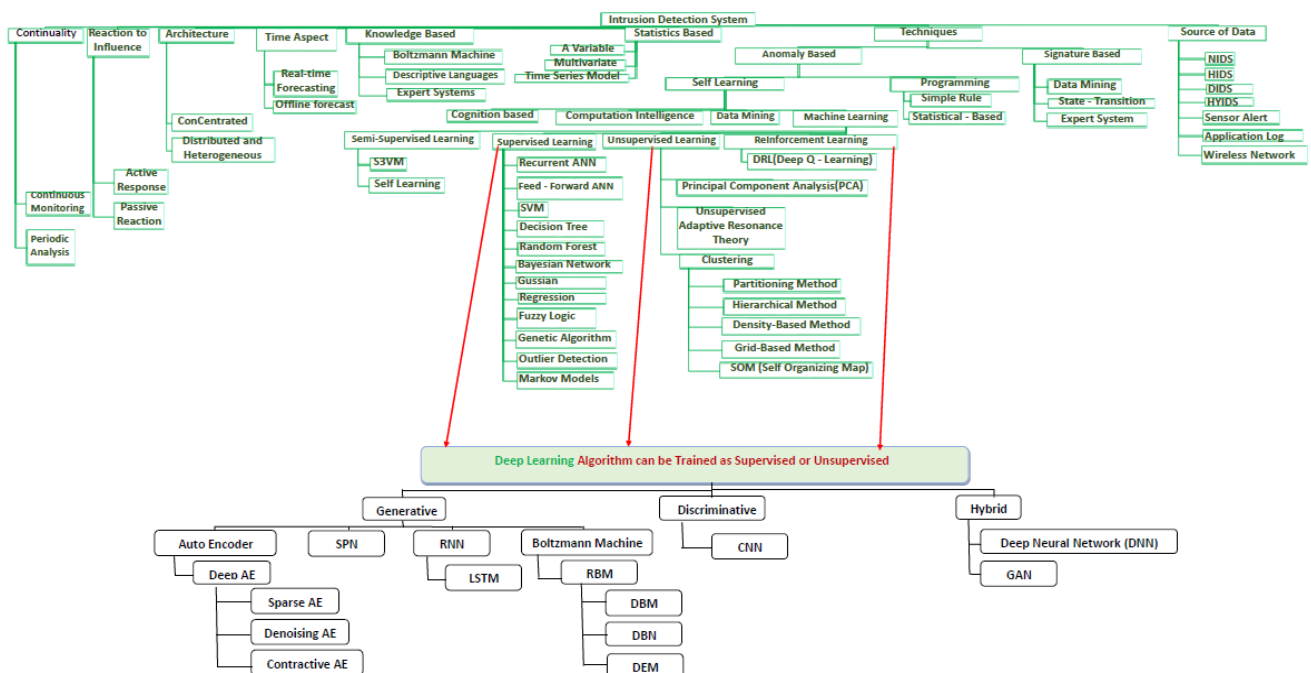


Figure 1. Research Tree in the field of intrusion detection systems



### 2.1. Shallow Learning IDSs

Some intrusion detection methods use shallow architectures, such as support vector machine (SVM), decision tree (DT), random forests (RF), clustering, K nearest neighbor (KNN), particle swarm optimization (PSO), simulated annealing (SA), ANN, and ensemble methods [4, 20-27].

Lin et al. [4] used the SVM, the decision tree, and the simulated annealing and reached 99.96% accuracy. Wang et al. [23] used the SVM algorithm and reached 99.31% accuracy. Baek et al. [22] achieved an 88% accuracy rate using several simple classifications.

These methods take advantage of the mentioned algorithms and use KDD99 or NSL-KDD datasets to evaluate their solutions and report good accuracy or precision rates. However, these methods report only one metric of accuracy or precision and no other metrics. They use only one dataset for evaluation, and they compare the results with only ordinary classifications.

### 2.2. Deep Learning IDSs

In this section, we describe intrusion detection models based on deep learning methods.

#### A. Convolutional Neural Network (CNN)

This category includes research works which have based their intrusion detection techniques on convolutional neural networks [28-32]. Zhu et al. [28] considered 6 layers of the neural networks and used the pooling layer among them. Moreover, they used a learning rate of 0.5 and achieved 80.34% of Accuracy. Li et al. [29] used convolution architecture and data-to-image conversion techniques to detect intrusion but provided a relatively low accuracy rate (about 80%). Nguyen et al. [32] used a deep convolutional network and used 4 main layers of CNN networks. They reported 99.87% accuracy on the KDD99 dataset.

#### B. Recursive Neural Network (RNN) or Gated Recurrent Unit RNN (GRU-RNN)

Research works in this category have used recursive neural network techniques [33-37]. For example, Yin et al. [33] proposed a binary classification method based on a deep recursive neural network to detect intrusions. They first performed pre-processing (such as normalization) on the input dataset and then attempted to weigh the deep network layers using a recursive neural network with forward propagation, reporting 99.81% accuracy. Tang et al. [34] proposed a gated recurrent unit (GRU) over SDN-based networks. They compared their method with DNN classifiers having different layouts, support vector machines, and simple Bayesian, and reported 89% accuracy and 87% precision. Zhong et al. [37] presented an IDS for IoT servers using text-CNN and GRU methods. They reported the F-score criterion on the KDD99 and ADFA-LD datasets.

#### C. Long Short-Term Memory (LSTM)

Ponkarthika and Saraswathy [38] developed an intrusion detection system based on the RNN and its specific type, and LSTM networks. They achieved 82% accuracy for the RNN and 83% accuracy for the LSTM on the KDD99 dataset with a learning rate of 200.

#### D. CNN-RNN

Vinayakumar et al. [39] proposed an intrusion detection technique using the convolutional network for feature

extraction and the RNN network for classification. They proposed a CNN-based model and showed that the CNN network would perform better than MLP, CNN-LSTM, and CNN-GRU in extracting and presenting features from network traffic. Their model could report the highest accuracy and recall on single-layer CNN and the highest precision on almost all CNN combinations with other networks on the KDD99 dataset at 99.9%. Chawla et al. [40] proposed a technique using a combined convolutional network and GRU RNN; they also could achieve 81% accuracy on the ADFA-LD dataset with a learning rate of 0.0001.

#### E. CNN-LSTM

The intrusion detection method proposed by Wang et al. [41] uses the convolution filter to extract the feature and the LSTM network for classification. That is, it uses CNN deep networks to learn low-level features and LSTM networks to learn high-level features. This method reported 99.89% accuracy on the ISCX2012 dataset. Furthermore, Hsu et al. [42] used a hybrid method based on LSTM and convolution network to detect intrusion and reported 94.12% accuracy on a larger dataset. Lee et al. [43] designed an intrusion detection system to prevent SSH and DDOS attacks in software-defined networks, which used four deep learning models, including MLP, CNN, LSTM, and SAE. Malik et al. [44] designed an Efficient Reconnaissance and Surveillance Detection in SDN using CNN and LSTM; however, they evaluated their model using only one dataset, namely CICIDS 2017.

#### F. RNN-LSTM

Jiang et al. [6] developed a multi-channel intelligent attack detection technique based on a combination of LSTM and RNN networks. In this LSTM-RNN architecture, multiple feature channels are given to the network input layer. Then, the LSTM layer, the Mean Pooling layer and finally the logistic regression layer are used. Finally, a majority vote is taken on the results obtained. Jiang et al. reported a detection rate of 99.23 and an accuracy of 98.94% on the NSL-KDD dataset.

#### G. Auto Encoder

The articles in this category [45-50] use deep Auto Encoder neural networks. Mohammadi and Namadchian [45] first performed normalization and then used a deep Auto Encoder method to reduce the error rate. Finally, on the NSL-KDD dataset, they achieved 92.72% accuracy and 98.11% detection rate in the classification of R2L attacks.

Papamartzivanos et al. [49] provided a comprehensive framework based on self-taught learning and MAPE-K methodology. The framework included plan, monitor, analyze, and execute activities that are applied to a knowledge base. Their model was a Sparse Auto Encoder and a Feedforward Auto Encoder. Their tests on the KDD99 and NSL-KDD datasets reported 99.8% and 99.6% accuracy.

#### H. Deep Neural Network (DNN)

The models in this category [51-58] use deep neural networks to detect network intrusions. In 2019, Vinayakumar et al. [5] proposed an IDS based on deep neural network and tested it on six datasets. The model layers

included the fully connected layer, the normalization layer, and the drop-out layer with a coefficient of 0.01. They used 16 consecutive layers, several ReLU activation functions, and learning rates between 0.01 and 0.5. This model reported good accuracy on six well-known datasets (e.g. 96.3% on the CICIDS dataset or 93% on the KDD99 dataset).

Tang et al. [51] also proposed a DNN-based intrusion detection method which was performed on the SDN environment and the NSL-KDD data set, and its experiments with a learning rate of 0.0001 reported an accuracy of 75.75. Using the Boltzmann neural network on the KDD99 dataset, Roy et al. [52] were able to report a very high 99.99% accuracy for two-class mode (attack or normal).

Ustebay et al. [53] used both the deep neural network and the shallow neural network (SNN) to detect abnormalities. They used these two models to reduce the feature set. They also trained the models on the CICIDS2017 dataset and reported a 98.45% accuracy rate on the deep models. They showed that deep models would achieve higher accuracy, precision, and recall than shallow models.

Vigneswaran et al. [54] evaluated DNN and SNN models on NIDS. They performed experiments on the DNN architecture with 1 to 5 layers at a learning rate of 0.1, considered 1000 Epochs on the KDD99 dataset, and finally compared the results with shallow machine learning algorithms. The results showed that the three-layer DNN had the best accuracy of 93% and precision of 99% among all these algorithms.

Duy et al. [57] designed a framework called DIGFuPAS which creates attack examples and acts like deep learning-based IDS in SDN in a Black-Box manner. They used Wasserstein Generative Adversarial (WGAN) Model, a generative model based on deep learning. Bouria and Guerroumi [58] presented an IDS based on a deep learning approach to strengthen SDN network security. The communication channel between the control layer and the infrastructure layer of the SDN is protected against various attacks. Moreover, they evaluated their model only on the CICIDS 2017 dataset.

### 2.3. Software-Defined Networks

Software-defined network is a new type of network architecture in which one or more central servers are responsible for controlling all network elements, whereas the rest of the elements only direct network traffic [59, 60]. Traditional networks were suitable for a static client-server structure. But today's modern networks, including data centers, cloud services, mobiles, and IoT devices, demand new requirements.

As you know, in traditional networks, each network device calculates routes and makes decisions on network policies. However, in SDN networks, the network operating system (the controller) is responsible for deciding how to route packets and applying network policies. The most essential concept in SDN networks is to separate the control plane and data plane. While the control plane decides how to route the packets, the switches and routers merely forward packets and are not involved in decision-making.

Apart from the controller and the network devices, some other components constitute the SDN architecture. For example, the SDN applications express their desired network behavior to the controller using some interfaces. Moreover,

the OpenFlow protocol communicates between the control and the data planes.

While SDN provides easy, flexible, and integrated management, it imposes several security issues. As the control logic in SDN is centralized, it is more vulnerable to cyber-attacks such as DDoS; therefore, the design of security appliances for SDN networks is crucial [61, 62].

### 3. The proposed model

The proposed model consists of three phases: 1) preprocessing phase, 2) neural network design phase, and 3) intrusion detection phase. In the first phase, the necessary pre-processing is performed on the raw data collected from the SDN network traffic. In the second phase, the neural network is designed with the appropriate layer arrangement and the proper activating function. The model is trained on the training dataset with the required number of repetitions. In the third phase, the trained model is tested on a test dataset, and the performance of the model is evaluated using various metrics such as accuracy, precision, and recall.

In deep learning, the goal of training is to increase the performance of the model using the defined training set. To measure the performance, we defined a loss function and reduced it in the hope that it would improve the overall performance of the model. While there are many loss functions to compute the distance between the true value and the estimated one, Cross entropy [3] is the most popular. In this research, we used the Cross entropy and the Adam optimizer for all three datasets. Cross entropy for a classification problem with  $n$  classes is defined as (1):

$$CE = -\sum_{i=1}^n t_i \log(p_i), \quad (1)$$

where  $t_i$  is the true value and  $p_i$  is the probability for the  $i^{th}$  class.

In this research, we considered seven different configurations for the neural network and evaluated all these seven models on three datasets, NSL-KDD, KDD99, and UNSW-NB15. It should be noted that the architecture and layered layout of the proposed models are the same for all three datasets, which is one of the strengths of our solution. To achieve proper performance, many previous models [5,24,28,59] have offered different layout layers for each data set, but our proposed architecture achieved good performance for all three datasets without manipulation.

Each of these seven models had a unique layout consisting of several layers, such as embedding, Dense, Drop out, and activation layers. The first model was a model based on dense layers and had nine layers. The second model was based on the convolutional neural network (CNN) and had the largest number of layers (22 layers). The third model was a 10-layer LSTM-CNN hybrid network. The fourth model was based on the dense network with the least number of layers. The fifth model, like the second one, was a CNN-based model with a relatively large number of layers. The sixth model was based on the LSTM-CNN hybrid network and had 12 layers. Finally, the seventh model was based on the dense network and had 12 layers. The designs of these seven models are described in Table 1. The number of neurons in each layer is represented in parentheses.

Table 1. The layout of the seven proposed ANN-based IDSs

Layers	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7
Layer 1	Dense(128)	Embedding	Embedding	Dense(128)	Embedding	Embedding	Dense(64)
Layer 2	Dense(64)	Dropout	Dropout	Dense(64)	Dropout	Dropout	Dense(32)
Layer 3	Dense(128)	Conv(256)	Conv(32)	Dense(128)	Conv(256)	Conv(64)	Activation
Layer 4	Dropout	Dense(100)	Conv(64)	Dense(64)	Dense(100)	Dropout	Dropout
Layer 5	Dense(64)	Dropout	Conv(128)	Dense(6)	Dropout	Dropout	Dense(32)
Layer 6	Dropout	Conv(128)	LSTM(128)	Dense(128)	Conv(128)	Conv(64)	Activation
Layer 7	Dense(6)	Dense(100)	Dense(100)	FC	Dense(100)	Dropout	Dropout
Layer 8	Dropout	Dropout	Dropout	---	Dropout	LSTM(300)	Dense(32)
Layer 9	FC	Conv(256)	Dense(100)	---	Conv(128)	Dense(100)	Activation
Layer 10	---	Dense(100)	FC	---	Dense(100)	Dropout	Dropout
Layer 11	---	Dropout	---	---	Dense(100)	Dense(10)	Dense(32)
Layer 12	---	Dense(200)	---	---	Dropout	FC	FC
Layer 13	---	Dropout	---	---	Dense(200)	---	---
Layer 14	---	Dense(100)	---	---	Conv(32)	---	---
Layer 15	---	Conv(32)	---	---	Conv(64)	---	---
Layer 16	---	Max-pool	---	---	Conv(128)	---	---
Layer 17	---	Dropout	---	---	Dense(100)	---	---
Layer 18	---	Dense(256)	---	---	Dropout	---	---
Layer 19	---	Dropout	---	---	Dense(256)	---	---
Layer 20	---	Dense(100)	---	---	Dropout	---	---
Layer 21	---	Dropout	---	---	FC	---	---
Layer 22	---	FC	---	---	---	---	---

We tested all seven proposed models against three datasets NSL-KDD, KDD99, and UNSW-NB15 and selected the best one (i.e., Model 7).

The best-proposed model (Model 7) was a unique 12-layer deep neural network with the following layer topology: dense, dense, activation, drop out, dense, activation, drop out, dense, activation, drop out, dense, and finally activation or fully connected (FC) layer which is used to select the appropriate class using SoftMax or Tanh functions. From now on, we will call Model 7 the proposed model. The proposed model improves the evaluation metrics without changing the number and layout of layers on the three datasets. This is the superiority of our solution over other works, which provides a different network architecture for each dataset.

However, it should be noted that since these three datasets are different in terms of the number of parameters and the number of output classes, our model also considers different parameters and final activation functions for selecting output classes. Also, the initial values for the dense layers are slightly different for each dataset. In the following, we will examine the layers of the proposed model.

**Dense Layer:** The values of the dense layers in the proposed model are different for each dataset and depend on the number of dataset properties. For example, for the UNSW-NB15 dataset, 64 values are provided for the first

layer. The activation function is also one of the best-tested functions for the neural network. The non-linear ReLU function is defined in the following (2):

$$ReLU(x) = \max(0, x), \quad (2)$$

where  $x$  is the input.

**Drop out layer:** The drop out layer accidentally removes and releases some neurons, preventing the network overfit. Therefore, it does not allow the network to retain data and to be disturbed in predicting the testing data. In the proposed model, a drop out layer with values of 0.15 to 0.5 is considered after each dense layer.

**Fully connected (FC) layer:** A fully connected layer (unlike a dense layer) is a layer that connects to all the neurons in the previous layer. It considers the trained inputs in the previous layers and assigns them to the appropriate class using an activation function such as SoftMax or Tanh, as defined in (3) and (4). Figure 2 illustrates the proposed model diagram:

$$SoftMax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^n e^{x_j}}, \quad (3)$$

$$Tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}, \quad (4)$$

where  $x$  is the input.

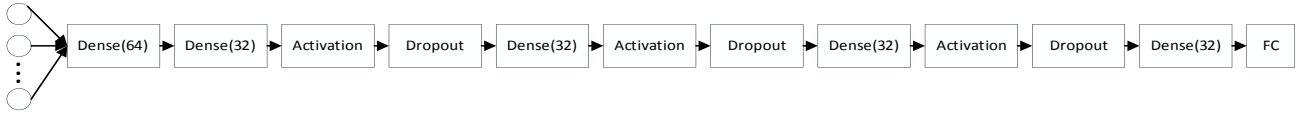


Figure 2. The layer layout of the proposed solution

The implementation of the proposed solution is described below. The data collected from the data sets of SDN networks were prepared and modeled. The layout, number, and type of network layers were set, the activation functions were selected, and the specified attack classes were converted into vectors. Then, the dataset was tagged with attack and non-attack labels. Finally, the input data, the output classes, layers, and weights of the data set in CSV format would make the DNN neural networks. We used TensorFlow and Keras deep learning package and Python programming language. To train the model faster and use powerful GPUs, we used the Google Colab service. We trained the model and the network with a suitable number of iterations and made sure of avoiding overfitting in each epoch. Finally, we tested the model using the test data set and evaluated the improvement of the model's performance in terms of accuracy, precision, recall, and cost function.

#### 4. The evaluation of the proposed model

##### 4.1. Datasets

In this study, three datasets, including NSL-KDD, KDD99, and UNSW-NB15, were used as benchmarks to select the best model among seven models and to compare the proposed model with other methods.

##### A. KDD99 dataset

The KDD99 dataset is an old dataset containing 41 features and five different classes: normal, DoS, remote-to-local (R2L), user-to-root (U2R), and Prob. It includes 494,021 records for training and 311,029 for testing sets. Some of the derived features include duration, protocol\_type, service, src\_bytes, dst\_bytes, flag, urgent, and so on. One drawback of KDD99 is that the sets of classes in the training and testing sets are imbalanced. Moreover, there are many duplicates in the dataset.

##### B. NSL-KDD dataset

The NSL-KDD dataset is one of the most widely used datasets for intrusion detection research; it is a subset of the original KDD99 and is designed to solve some of the drawbacks of the KDD99 dataset. This dataset does not have duplicate records in the training and testing sets, and the number of records is considered more reasonable and appropriate. The feature set and the type of classes are the same as the original KDD99.

##### C. UNSW-NB15 dataset

UNSW-NB15 is a relatively new dataset with a hybrid of real normal activities and synthetic contemporary attacks. It has 175,341 records in the train set and 82,332 records in the test set. The dataset has ten classes (normal and nine types of attacks). The attack types are DoS, backdoors, fuzzers, analysis, exploits, generic, shellcode, reconnaissance, and Worms. Moreover, there are 49 derived features.

##### 4.2. Evaluation Metrics

The most important and widely used metrics to evaluate the

quality of the results of intrusion detection methods are: 1) accuracy, 2) precision, 3) recall, 4) F-measure, and 5) loss function [49, 63-65]. At first, it is necessary to define the four basic terms used in the mentioned metrics [66]:

- True Positive (TP) indicates the number of records in the dataset that our method correctly classified in the attack class.
- True Negative (TN) is the number of records in the dataset that our method rightly classified in the normal category.
- False Positive (FP) indicates the number of records in the dataset that our method incorrectly classified in the attack class.
- False Negative (FN) is the number of records in the dataset that our method mistakenly classified in the normal category.

In the following, we will explain the application of these basic terms in the mentioned evaluation metrics.

**Precision:** This metric estimates the ratio of correctly identified attack records to the total number of detected attack records (5):

$$Precision = \frac{TP}{TP+FP} \quad (5)$$

**Recall:** This metric estimates the ratio of correctly classified attack records to the total number of attack records (6):

$$Recall = \frac{TP}{TP+FN} \quad (6)$$

**Accuracy:** This metric estimates the ratio of correctly classified records to the total records. In other words, the accuracy metric shows the percentage of the data that are correctly categorized in (7):

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

**F-measure:** This metric establishes a tradeoff between precision and recall. It is the harmonic mean of precision and recall (8):

$$F = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (8)$$

**Loss function:** This metric indicates the amount of output error, and we can get good results by optimizing it [36, 51, 67].

##### 4.3. The evaluation of the Proposed Solution

To implement the proposed models, we used the Jupyter Notebook in the free Google Collaboratory service. In particular, we used the Tensorflow 1.0 deep learning package [68] along with Keras Backend and the Adam optimizer with different learning rates. Moreover, we used the most popular cost function, that is, Cross Entropy. The seven neural network models were examined and evaluated on the three datasets with the same conditions.

Table 2. The performance of the seven proposed models on the KDD99 dataset

Metrics	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7
Accuracy	53.96	98.93	50.87	61.1	98.9	88.22	99.02
Precision	83.79	88.67	78.54	98.27	88.38	63.17	99.14
Recall	76.19	86.34	79	41.24	85.94	58.77	98.93
F-measure	78.78	87.49	78.74	58.08	87.14	60	99.04
Loss function	0.046	0.03	0.054	0.415	0.031	0.077	0.107

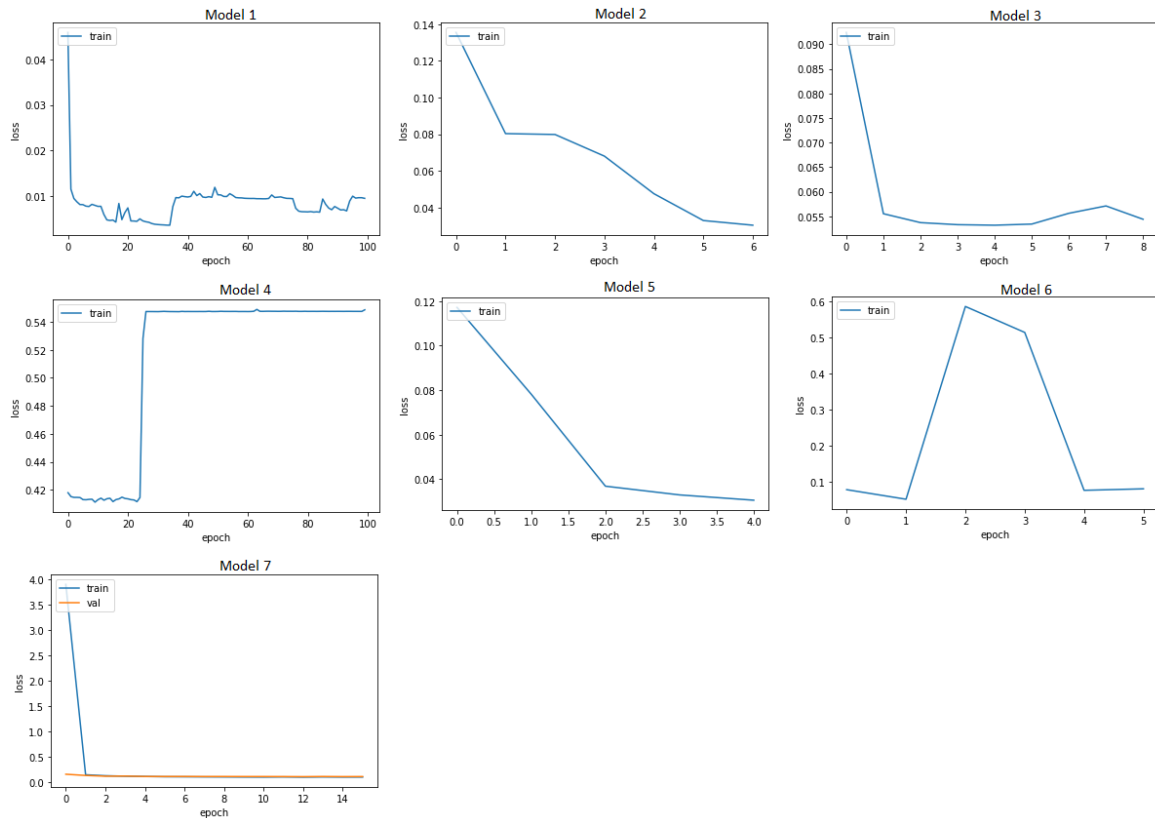


Figure 3. The evaluation of the loss function of the seven proposed models on the KDD99 dataset

#### A. The evaluation of the seven models on the KDD99 dataset

We examined all seven proposed models with the KDD99 dataset. Table 2 shows the accuracy, precision, recall, F-measure, and loss functions of all the models. The performance of Model 7 was better than the other models in terms of accuracy, precision, recall, and F-measure; however, the loss function of Model 7 was higher than most models. Model 2, with a value of 0.03, had the lowest loss function.

Figure 3. shows the evaluation of the loss function on our seven models for the KDD99 dataset. In Model 7, the cost function for validation data (orange line) was approximately tangent to the cost function for training data (blue line). For other models, only training data were examined due to the imbalanced training and validation data.

In Model 1, with increasing epochs, the error decreased but with fluctuations, which can be attributed to the lack of use of drop out layer for optimal control of overfitting. Model 2 had a stepped decrease. In the first repetitions, it had good learning from training data, however, in the subsequent repetitions, the learning rate decreased. It should be noted that this model had the best value of the loss function.

Model 3 did not reach the minimum value of the loss

function but had a good decreasing slope. The loss function of Model 4 not only decreased after a while, but also it showed an increase due to the high learning rate and inappropriate layer arrangement. The lower the learning rate, the greater the possibility of improving the loss function. The loss function of this model was the worst loss function among the design models.

Model 5 initially had a sudden decrease and then reached a slow and relatively uniform decrease. The learning rate gradually improved better in this model. In Model 6, the overfitting fluctuations were uncontrolled, and the error increased and decreased abruptly. The sixth model worked well on the training data. However, after feeding new data, the loss function increased due to not using the drop out layer correctly.

Model 7 (i.e., the best proposed model) did not have the least loss function among all the models, but it was able to control overfitting with the correct arrangement of layers. This model also performed well in the validation dataset.

#### B. The evaluation of the seven models on the NSL-KDD dataset

Table 3. shows the performance values of the seven models on the NSL-KDD dataset. It is quite clear that the proposed

solution (Model 7) performed much better than other models in all evaluation criteria, even in the loss function. As compared to other models, this showed the excellent performance of the proposed solution and the proper arrangement of the neural network layers in it.

Considering Figure 4, it is clear that Model 7 greatly reduced the loss function. In fact, the loss function had a slight difference in both the training and validation datasets. If the loss function of the training data were close to the loss function of the validation data, it would be safe to say that the over-fitting is well controlled. The loss function in Model 7 reached the lowest possible value among the seven models. One of the reasons for this smooth reduction of the loss function was the correct use of drop out layers between the dense layers.

Model 1 had the lowest loss after model 7. The value of the loss function could be well reduced due to its good learning rate. Of course, the loss function fluctuated with the arrival of some new data, and the network controlled the fluctuations using the appropriate learning rate. The cost function of the second model initially decreased but

remained constant after a few iterations. To solve this problem, the learning rate should be adjusted and reduced during the training steps.

The layout of Model 3 was not able to reduce the loss function well. Model 4, like the third model, was subject to fluctuations in new train data. This indicated that the model had learned well from previous data; however, the error fluctuated with new data, which was not very acceptable.

The loss function of Model 5 remained constant very soon and could not reduce the loss function more than this amount. Adjusting the input weights of the next layers was very important. In the fifth model, the input weights of the layers were not well adjusted and had been updated with a constant value, producing a fixed loss function.

Model 6 did not perform well in this dataset. Increasing the learning rate initially reduced the loss function, but the error rate then increased. It seems that by reducing the learning rate in these models, we can solve this problem and improve the model performance. Thus, Model 7 in the NSL-KDD dataset is undoubtedly the best-designed model according to the evaluation criteria under consideration.

Table 3. The performance of the seven proposed models on the NSL-KDD dataset

Metrics	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7
Accuracy	71.75	95.95	96.53	67.08	95.95	61.32	99.39
Precision	83.51	53.26	93.25	91.17	53.48	80.33	99.49
Recall	69.79	53.23	70.23	78.4	53.44	84.11	99.33
F-measure	75.63	53.24	79.98	83.89	53.46	81.74	99.41
Loss function	0.068	0.088	0.084	0.094	0.088	0.076	0.022

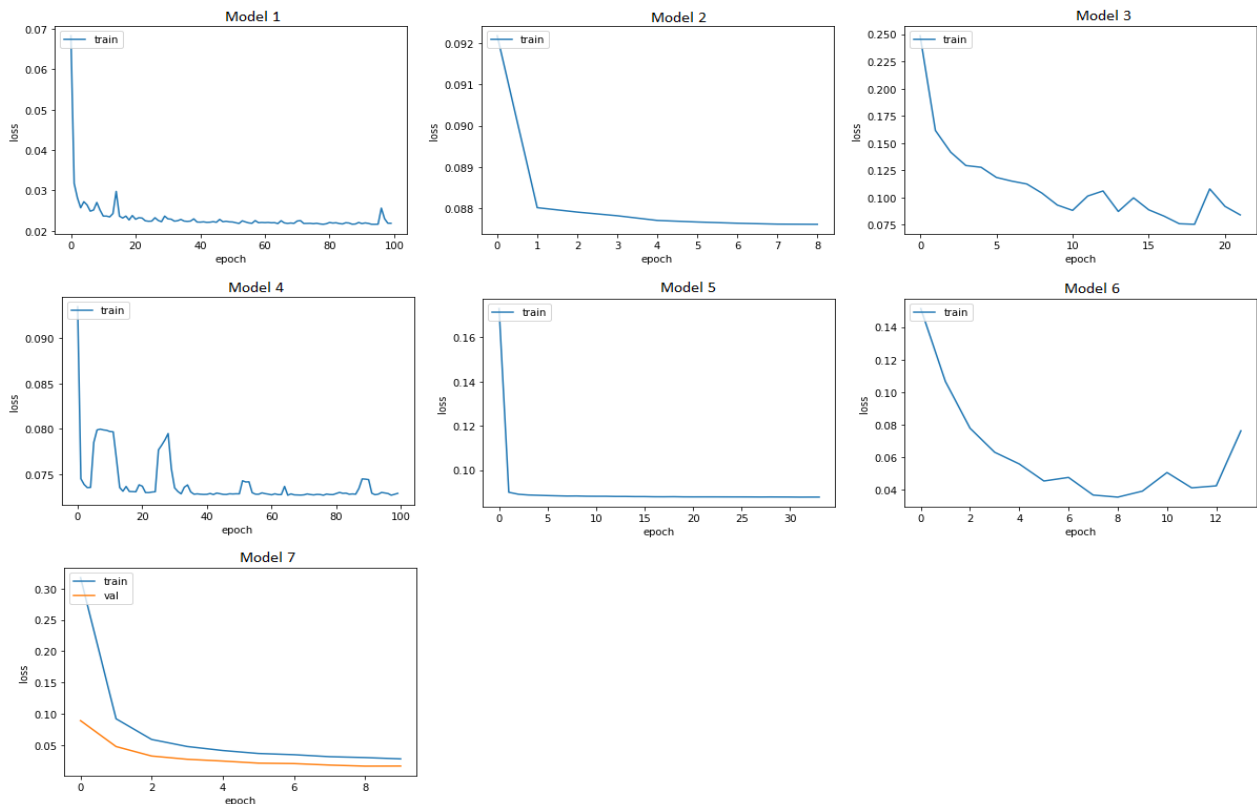


Figure 4. The evaluation of the loss function of the seven proposed models on the NSL-KDD dataset



### C. The evaluation of the seven models on the UNSW-NB15 dataset

Table 4. shows the evaluation of our models on the UNSW-NB15 dataset. Model 7 had the highest value in terms of accuracy and F-measure. In addition, this model achieved a recall of 99.98% (approximately one). Although the third and fourth models achieved 100% recall, other performance measures of these two models were lower than Model 7. However, the loss function of Model 7 was higher than Models 2, 4, 5, and 6. It is possible to reduce the error rate by changing the activation function for this dataset. However, changing the activation function is not acceptable, and we consider fixed activation functions for three datasets.

Referring to Table 2, Table 3, and Table 4, it is clear that the highest value for accuracy metric (one of the most important evaluation metrics in intrusion detection systems) on all three datasets of KDD99, NSL-KDD, and UNSW-NB15 belonged to the proposed model (Model 7).

Figure 5 shows the loss functions of the seven models on the UNSW-NB15 dataset. It is clear that the first model

increased the error instead of decreasing it and thus had the worst loss function among the seven models. It can be inferred that the final activation function of Model 1 failed to predict the correct class. Given that changing the activation may reduce the error, we did not change it in this study; in fact, we considered fixed activation functions for all three datasets.

The loss function of the second model had a decreasing trend, which implies that the layer arrangement and the final activation function were chosen properly. The third model is almost the same as the second model and has the least loss value. The loss function in the fourth model continuously decreased; however, it did not reach the lowest level and was fixed at approximately 0.6881%.

While the loss functions of the fifth and the sixth models performed similarly, the fifth model acted slightly better. The sixth model had a higher learning rate than the fifth model, but it controlled the overfitting better. The loss function of the proposed model also had a decreasing trend, but its error rate was higher than the other models.

Table 4. The performance of the seven proposed models on the UNSW-NB15 dataset

Metrics	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7
Accuracy	4.3	64.04	55.05	55.04	64.02	66.45	68.11
Precision	3.3	68.5	55.05	55.04	69.15	74.54	68.13
Recall	2.2	65.56	100	100	64.55	59.36	99.98
F-measure	2.2	66.59	71	70.99	66.11	66.06	80.97
Loss function	8.86	0.632	0.34	0.688	0.635	0.739	7.16

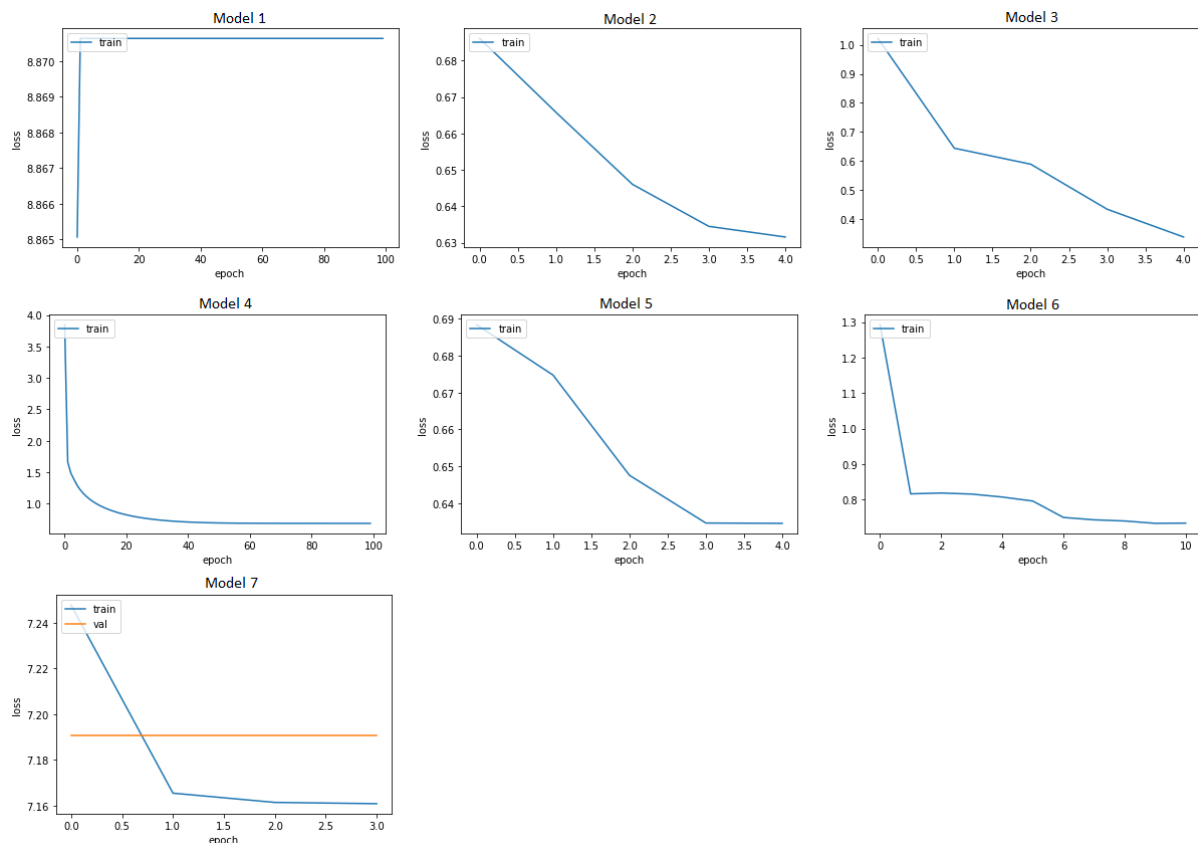


Figure 5. The evaluation of the loss function of the seven proposed models on the UNSW-NB15 dataset

#### D. Performance Comparison

In this section, we will compare the results of our model with the findings offered by VinayaKumar et al. [5] in which excellent evaluation metrics values are obtained on six datasets.

As shown in Figure 6, the proposed model performs better than VinayaKumar et al.'s work on the KDD99 dataset in terms of all evaluation metrics (accuracy, precision, recall, and F-measure). Figure 7 shows a comparison of the proposed method with VinayaKumar et al.'s method on the NSL-KDD dataset. The proposed method works much better than VinayaKumar et al.'s work. All four criteria in the proposed method are close to 100%, while in VinayaKumar et al.'s method they are about 80%.

The proposed solution and the method offered by VinayaKumar et al. on the UNSW-NB15 dataset were also compared. Figure 8 shows that the accuracy of the proposed model is 68.11%, and the accuracy of the method of VinayaKumar et al. is 65.1%, and therefore the proposed solution works better. Regarding the Precision metric, VinayaKumar et al.'s method is 59.7%, and the proposed method is 68.13%. Also, in the recall metric, the proposed method performs much better than the method of VinayaKumar et al.

It should be noted that the UNSW-NB15 dataset is one of the largest intrusion detection datasets, and the improvement obtained by the proposed method on this dataset is valuable.

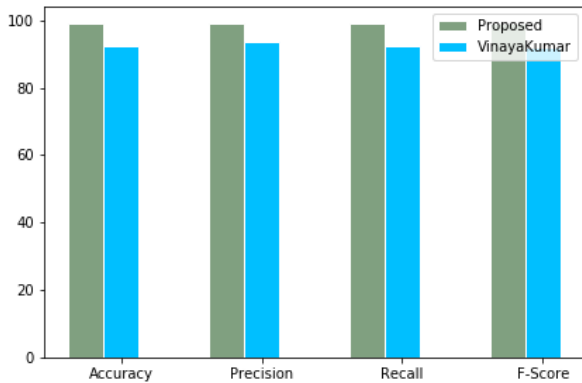


Figure 6. Comparison between the proposed model and VinayaKumar et al.'s work on the KDD99 dataset

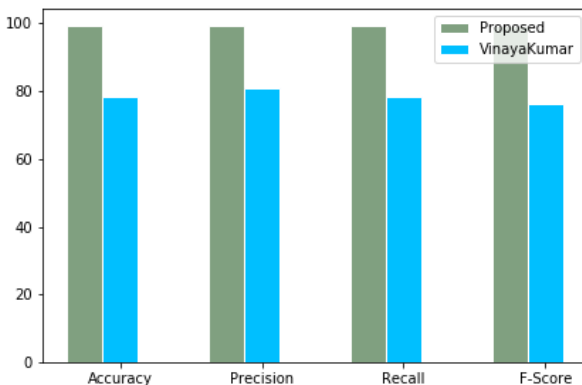


Figure 7. Comparison between the proposed model and VinayaKumar et al.'s work on the NSL-KDD dataset

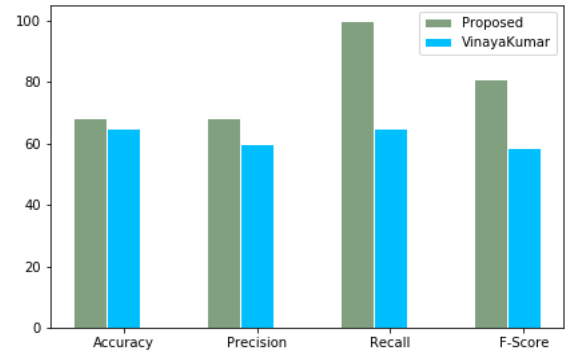


Figure 8. Comparison between the proposed model and VinayaKumar et al.'s work on the UNSW-NB15 dataset

In the following, we will compare the performance of the proposed model with several other models. The models in [69-71] are evaluated using NSL-KDD. As can be seen in Table 5, the accuracy, precision, recall, and F-score of our model are better than these models in this dataset. The model in [37] reports the F-measure on the KDD99 dataset. However, as Table 5 shows, the F-measure of our model is greater than the F-measure of [37]. Finally, the model in [72] reports the precision equal to 93.41 on UNSW-NB15, which is greater than our precision score on this dataset. However, we should mention that our model, unlike [72], has acceptable performance on each of these three datasets.

Table 5. Comparison between the proposed model and other state-of-the-art models on the KDD99, NSL-KDD, and UNSW-NB15 datasets

DataSet		KDD99	NSL-KDD	UNSW-NB15
References / Metrics				
Proposed Model	Accuracy	99.02	99.39	68.11
	Precision	99.14	99.49	68.13
	Recall	98.93	99.33	99.98
	F-measure	99.04	99.41	80.97
[69]	Accuracy	-	79.08	-
	Precision	-	87.27	-
	Recall	-	94.60	-
	F-measure	-	91.47	-
[72]	Accuracy	-	-	-
	Precision	-	-	93.41
	Recall	-	-	-
	F-measure	-	-	-
[37]	Accuracy	-	-	-
	Precision	-	-	-
	Recall	-	-	-
	F-measure	94.50	-	-
[70]	Accuracy	-	90.73	-
	Precision	-	86.38	-
	Recall	-	93.17	-
	F-measure	-	89.65	-
[71]	Accuracy	-	86.70	-
	Precision	-	89.36	-
	Recall	-	86.70	-
	F-measure	-	87.22	-

## 5. Conclusion and future work

One of the challenges of SDN networks is to design an intrusion detection system that can prevent various types of attacks. While several methods have provided IDSs for SDNs, none of them has been able to achieve suitable performance values on different available datasets.

In this study, to improve the security level of the network and prevent various attacks, we proposed an intrusion detection system based on a 12-layer deep neural network. This intrusion detection system was trained and tested on three SDN-specific datasets, namely NSL-KDD, KDD99, and UNSW-NB15. We evaluated our model over these datasets. The accuracy, precision, recall, and F-measure of the model on KDD99 were 99.02, 99.14, 98.93, and 99.04, respectively. These measures on the NSL-KDD dataset were 99.39, 99.49, 99.33, and 99.41, respectively. Furthermore, the model on the UNSW-NB15 dataset reached good results. The results on the three datasets show that our model can reduce the loss function significantly. Moreover, we compared our model with six recent works. The experiment results showed the supremacy of the proposed model over these models.

For future work, the authors plan to work on the following:

- Working on different datasets. While only three widely used datasets are examined in this study, we can work on more than 20 publicly available SDN-specific datasets.
- Implementing other neural network architectures, including CNNs, such as MobileNet, AlexNet, or LeNet. Another possible work is to ensemble the proposed model with other deep architectures or meta-heuristic algorithms such as particle swarm optimization (PSO) algorithm.

## 6. Reference

- [1] Heady, R., Luger, G., Maccabe, A., and Servilla, M., "The architecture of a network level intrusion detection system", *Los Alamos National Lab.*, 1990.
- [2] Panda, M., Abraham, A., and Patra, M. R., "A hybrid intelligent approach for network intrusion detection", *Procedia Engineering*, Vol. 30, pp. 1-9, 2012.
- [3] Sheikhan, M., and Bostani, H., "A hybrid intrusion detection architecture for internet of things", in *2016 8th International Symposium on Telecommunications (IST)*, IEEE, pp. 601-606, 2016.
- [4] Lin, S.-W., Ying, K.-C., Lee, C.-Y., and Lee, Z.-J., "An intelligent algorithm with feature selection and decision rules applied to anomaly intrusion detection", *Applied Soft Computing*, Vol. 12, No. 10, pp. 3285-3290, 2012.
- [5] Vinayakumar, R., Alazab, M., Soman, K., Poornachandran, P., Al-Nemrat, A., and Venkatraman, S., "Deep learning approach for intelligent intrusion detection system", *IEEE Access*, Vol. 7, pp. 41525-41550, 2019.
- [6] Jiang, F., et al., "Deep learning based multi-channel intelligent attack detection for data security", *IEEE transactions on Sustainable Computing*, Vol. 5, No. 2, pp. 204-212, 2018.
- [7] Chalapathy, R., and Chawla, S., "Deep learning for anomaly detection: A survey", *arXiv preprint arXiv:1901.03407*, 2019.
- [8] Sultana, N., Chilamkurti, N., Peng, W., and Alhadad, R., "Survey on SDN based network intrusion detection system using machine learning approaches", *Peer-to-Peer Networking and Applications*, Vol. 12, No. 2, pp. 493-501, 2019.
- [9] Kwon, D., Kim, H., Kim, J., Suh, S. C., Kim, I., and Kim, K. J., "A survey of deep learning-based network anomaly detection", *Cluster Computing*, Vol. 22, No. 1, pp. 949-961, 2019.
- [10] Chaabouni, N., Mosbah, M., Zemmari, A., Sauvignac, C., and Faruki, P., "Network intrusion detection for IoT security based on learning techniques", *IEEE Communications Surveys & Tutorials*, Vol. 21, No. 3, pp. 2671-2701, 2019.
- [11] Zhong, G., Ling, X., and Wang, L. N., "From shallow feature learning to deep learning: Benefits from the width and depth of deep architectures", *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, Vol. 9, No. 1, pp. e1255, 2019.
- [12] Hodo, E., Bellekens, X., Hamilton, A., Tachtatzis, C., and Atkinson, R., "Shallow and deep networks intrusion detection system: A taxonomy and survey", *arXiv preprint arXiv:1701.02145*, 2017.
- [13] Lansky, J., Ali, S., Mohammadi, M., Majeed, M. K., Karim, S.H., Rashidi, S., Hosseinzadeh, M., Rahmani, A.M., "Deep learning-based intrusion detection systems: a systematic review", *IEEE Access*, No. 14, Vol. 9, pp. 101574-99, Jul, 2021.
- [14] Ahmad Z., Shahid Khan, A., Wai Shiang, C., Abdullah, J., Ahmad, F., "Network intrusion detection system: A systematic study of machine learning and deep learning approaches", *Transactions on Emerging Telecommunications Technologies*, Vol. 32(1), pp. e4150, Jan, 2021.
- [15] Kocher, G., Kumar, G., "Machine learning and deep learning methods for intrusion detection systems: recent developments and challenges", *Soft Computing*, Vol. 25(15), pp. 9731-63, Aug, 2021.
- [16] Jasim, A. D., "A survey of intrusion detection using deep learning in internet of things", *Iraqi Journal For Computer Science and Mathematics*, Vol. 3(1), pp. 83-93, Jan 30, 2022.
- [17] Alsoufi, M. A., Razak, S., Siraj, M. M., Nafea, I., Ghaleb, F. A., Saeed, F., Nasser, M., "Anomaly-based intrusion detection systems in iot using deep learning: A systematic literature review", *Applied Sciences*, No. 9, Vol. 11(18), pp. 8383, sep, 2021.
- [18] Lee, S. W., Mohammadi, M., Rashidi, S., Rahmani, A. M., Masdari, M., Hosseinzadeh, M., "Towards secure intrusion detection systems using deep learning techniques: Comprehensive analysis and review", *Journal of Network and Computer Applications*, Aug No. 1, Vol. 187, pp. 103111, Aug, 2021.
- [19] Ahmed, M., Shatabda, S., Islam, A. K., Robin, M., Islam, T., "Intrusion detection system in software-defined networks using machine learning and deep learning techniques—a comprehensive survey", 2021.
- [20] Wang, G., Hao, J., Ma, J., and Huang, L., "A new approach to intrusion detection using Artificial Neural Networks and fuzzy clustering", *Expert systems with applications*, Vol. 37, No. 9, pp. 6225-6232, 2010.
- [21] Aburomman, A. A., and Reaz, M. B. I., "A novel SVM-kNN-PSO ensemble method for intrusion detection system", *Applied Soft Computing*, Vol. 38, pp. 360-372,

- 2016.
- [22] Baek, S., Kwon, D., Kim, J., Suh, S. C., Kim, H., and Kim, I., "Unsupervised labeling for supervised anomaly detection in enterprise and cloud networks", in *2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)*, IEEE, pp. 205-210, 2017.
  - [23] Wang, H., Gu, J., and Wang, S., "An effective intrusion detection framework based on SVM with feature augmentation", *Knowledge-Based Systems*, Vol. 136, pp. 130-139, 2017.
  - [24] Aljawarneh, S., Aldwairi, M., and Yassein, M. B., "Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model", *Journal of Computational Science*, Vol. 25, pp. 152-160, 2018.
  - [25] Pham, N. T., Foo, E., Suriadi, S., Jeffrey, H., and Lahza, H. F. M., "Improving performance of intrusion detection system using ensemble methods and feature selection", in *Proceedings of the Australasian Computer Science Week Multiconference*, pp. 1-6, 2018.
  - [26] He, D., Chen, X., Zou, D., Pei, L., and Jiang, L., "An improved kernel clustering algorithm used in computer network intrusion detection", in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, IEEE, pp. 1-5, 2018.
  - [27] Song, J., Takakura, H., Okabe, Y., and Kwon, Y., "Unsupervised anomaly detection based on clustering and multiple one-class SVM", *IEICE transactions on communications*, Vol. 92, No. 6, pp. 1981-1990, 2009.
  - [28] Zhu, M., Ye, K., and Xu, C.-Z., "Network anomaly detection and identification based on deep learning methods", in *International Conference on Cloud Computing* Springer, pp. 219-234, 2018.
  - [29] Li, Z., Qin, Z., Huang, K., Yang, X., and Ye, S., "Intrusion detection using convolutional neural networks for representation learning", in *International conference on neural information processing*, Springer, Vol. ???, pp. 858-866, 2017.
  - [30] Liu, Y., Liu, S., and Zhao, X., "Intrusion detection algorithm based on convolutional neural network", *DEStech Transactions on Engineering and Technology Research*, No. iceta, 2017.
  - [31] Potluri, S., Ahmed, S., and Diedrich, C., "Convolutional neural networks for multi-class intrusion detection system", in *International Conference on Mining Intelligence and Knowledge Exploration*, Springer, pp. 225-238, 2018.
  - [32] Nguyen, S. -N., Nguyen, V.-Q., Choi, J., and Kim, K., "Design and implementation of intrusion detection system using convolutional neural network for DoS detection", in *Proceedings of the 2nd international conference on machine learning and soft computing*, pp. 34-38, 2018.
  - [33] Yin, C., Zhu, Y., Fei, J., and He, X., "A deep learning approach for intrusion detection using recurrent neural networks", *Ieee Access*, Vol. 5, pp. 21954-21961, 2017.
  - [34] Tang, T. A., Mhamdi, L., McLernon, D., Zaidi, S. A. R., and Ghogho, M., "Deep recurrent neural network for intrusion detection in sdn-based networks", in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, IEEE, pp. 202-206, 2018.
  - [35] Vinayakumar, R., Soman, K., and Poornachandran, P., "A comparative analysis of deep learning approaches for network intrusion detection systems (N-IDSs): deep learning for N-IDSs", *International Journal of Digital Crime and Forensics (IJDCF)*, Vol. 11, No. 3, pp. 65-89, 2019.
  - [36] Chockwanich, N., and Visoottiviseth, V., "Intrusion detection by deep learning with tensorflow", in *2019 21st International Conference on Advanced Communication Technology (ICACT)*, IEEE, pp. 654-659, 2019.
  - [37] Zhong, M., Zhou, Y., Chen, G., "Sequential model based intrusion detection system for IoT servers using deep learning methods. Sensors", No. 5, Vol. 21(4), pp. 1113, Feb, 2021.
  - [38] Ponkarthika, M., and Saraswathy, V., "Network intrusion detection using deep neural networks", *Asian Journal of Science and Technology*, Vol. 2, No. 2, pp. 665-673, 2018.
  - [39] Vinayakumar, R., Soman, K., and Poornachandran, P., "Applying convolutional neural network for network intrusion detection", in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, IEEE, pp. 1222-1228, 2017.
  - [40] Chawla, A., Lee, B., Fallon, S., and Jacob, P., "Host based intrusion detection system with combined CNN/RNN model", in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, pp. 149-158, 2018.
  - [41] Wang, W., et al., "HAST-IDS: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection", *IEEE access*, Vol. 6, pp. 1792-1806, 2017.
  - [42] Hsu, C.-M., Hsieh, H.-Y., Prakosa, S. W., Azhari, M. Z., and Leu, J.-S., "Using long-short-term memory based convolutional neural networks for network intrusion detection", in *International wireless internet conference*, Springer, pp. 86-94, 2018.
  - [43] Lee T. H., Chang, L. H., Syu, C. W., "Deep learning enabled intrusion detection and prevention system over SDN networks", in *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, Jun 7, pp. 1-6, IEEE, 2020.
  - [44] Malik, J., Akhunzada, A., Bibi, I., Imran, M., Musaddiq, A., Kim, S. W., "Hybrid deep learning: An efficient reconnaissance and surveillance detection mechanism in SDN", *IEEE Access*. No. 16, Vol. 8, pp. 134695-706, Jul, 2020.
  - [45] Mohammadi, S., and Namadchian, A., "A new deep learning approach for anomaly base IDS using memetic classifier", *International Journal of Computers Communications & Control*, Vol. 12, No. 5, pp. 677-688, 2017.
  - [46] Niyaz, Q., "Design and Implementation of a Deep Learning based Intrusion Detection System in Software-Defined Networking Environment", University of Toledo, 2017.
  - [47] Shone, N., Ngoc, T. N., Phai, V. D., and Shi, Q., "A deep learning approach to network intrusion detection", *IEEE transactions on emerging topics in computational*

- intelligence*, Vol. 2, No. 1, pp. 41-50, 2018.
- [48] Farahnakian, F., and Heikkonen, J., "A deep auto-encoder based approach for intrusion detection system", in *2018 20th International Conference on Advanced Communication Technology (ICACT)*, IEEE, pp. 178-183, 2018.
- [49] Papamartzivanos, D., Mármol, F. G., and Kambourakis, G., "Introducing deep learning self-adaptive misuse network intrusion detection systems", *IEEE Access*, Vol. 7, pp. 13546-13560, 2019.
- [50] A. Abusitta, M. Bellaiche, M. Dagenais, and T. Halabi, "A deep learning approach for proactive multi-cloud cooperative intrusion detection system", *Future Generation Computer Systems*, Vol. 98, pp. 308-318, 2019.
- [51] Tang, T. A., Mhamdi, L., McLernon, D., Zaidi, S. A. R., and Ghogho, M., "Deep learning approach for network intrusion detection in software defined networking", in *2016 international conference on wireless networks and mobile communications (WINCOM)*, IEEE, pp. 258-263, 2016.
- [52] Roy, S. S., Mallik, A., Gulati, R., Obaidat, M. S., and Krishna, P. V., "A deep learning based artificial neural network approach for intrusion detection", in *International Conference on Mathematics and Computing*, Springer, pp. 44-53, 2017.
- [53] Ustebay, S., Turgut, Z., and Aydin, M. A., "Cyber attack detection by using neural network approaches: shallow neural network, deep neural network and autoencoder," in *International conference on computer networks*, Springer, pp. 144-155, 2019.
- [54] Vigneswaran, R. K., Vinayakumar, R., Soman, K., and Poornachandran, P., "Evaluating shallow and deep neural networks for network intrusion detection systems in cyber security", in *2018 9th International conference on computing, communication and networking technologies (ICCCNT)*, IEEE, pp. 1-6, 2018.
- [55] Javeed, D., Gao, T., Khan, M.T., Ahmad, I., "A hybrid deep learning-driven SDN enabled mechanism for secure communication in Internet of Things (IoT)", *Sensors*, No. 18, Vol. 21(14), pp. 4884, Jul, 2021.
- [56] Hande, Y., Muddana, A., "Intrusion detection system using deep learning for software defined networks (SDN)", in *2019 International Conference on Smart Systems and Inventive Technology (ICSSIT) 2019 Nov 27*, pp. 1014-1018, IEEE, 2019.
- [57] Duy, P. T., Khoa, N. H., Nguyen, A. G., Pham, V. H., "DIGFuPAS: Deceive IDS with GAN and Function-Preserving on Adversarial Samples in SDN-enabled networks", *Computers & Security*, No. 1, Vol. 109, pp. 102367, Oct, 2021.
- [58] Boukria, S., Guerroumi, M., "Intrusion detection system for SDN network using deep learning approach", in *2019 International Conference on Theoretical and Applicative Aspects of Computer Science (ICTAACS)*, Dec 15, Vol. 1, pp. 1-6, IEEE, 2019.
- [59] Seyedkolaei, A. A., Seno, S. A. H., Moradi, A., and Budiarto, R., "Cost-Effective Survivable Controller Placement in Software-Defined Networks", *IEEE Access*, Vol. 9, pp. 129130-129140, 2021.
- [60] Prajapati, A., Sakadasariya, A., and Patel, J., "Software defined network: Future of networking", in *2018 2nd International Conference on Inventive Systems and Control (ICISC)*, IEEE, pp. 1351-1354, 2018.
- [61] Khairi, M. H., Ariffin, S. H., Latiff, N. A., Abdullah, A., and Hassan, M., "A review of anomaly detection techniques and distributed denial of service (DDoS) on software defined network (SDN)", *Engineering, Technology & Applied Science Research*, Vol. 8, No. 2, pp. 2724-2730, 2018.
- [62] Shaghaghi, A., Kaafar, M. A., Buyya, R., and Jha, S., "Software-defined network (SDN) data plane security: issues, solutions, and future directions", *Handbook of Computer Networks and Cyber Security*, pp. 341-387, 2020.
- [63] Ludwig, S. A., "Intrusion detection of multiple attack classes using a deep neural net ensemble", in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, IEEE, pp. 1-7, 2017.
- [64] Faker, O., and Dogdu, E., "Intrusion detection using big data and deep learning techniques", in *Proceedings of the 2019 ACM Southeast Conference*, pp. 86-93, 2019.
- [65] Rawat, S., Srinivasan, A., Ravi, V., Ghosh, U., "Intrusion detection systems using classical machine learning techniques vs integrated unsupervised feature learning and deep neural network", *Internet Technology Letters*, Jan, Vol. 5(1), pp. e232, 2022.
- [66] Powers, D. M., "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation", *arXiv preprint arXiv*, 2010.16061, 2020.
- [67] Revathi, S., and Malathi, A., "A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection", *International Journal of Engineering Research & Technology (IJERT)*, Vol. 2, No. 12, pp. 1848-1853, 2013.
- [68] Abadi, M., et al., "Tensorflow: A system for large-scale machine learning", in *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pp. 265-283, 2016.
- [69] Saritha Reddy, A., Ramasubba Reddy, B., Suresh Babu, A., "An Improved Intrusion Detection System for SDN using Multi-Stage Optimized Deep Forest Classifier", *International Journal of Computer Science & Network Security*, Vol. 22(4), pp. 374-86, 2022.
- [70] Fu, Y., Du, Y., Cao, Z., Li, Q., Xiang, W. A., "Deep Learning Model for Network Intrusion Detection with Imbalanced Data", *Electronics*, Mar 14, Vol. 11(6), pp. 898, 2022.
- [71] Imrana, Y., Xiang, Y., Ali, L., Abdul-Rauf, Z., "A bidirectional LSTM deep learning approach for intrusion detection", *Expert Systems with Applications*, Dec 15, Vol. 185, pp. 115524, 2021.
- [72] Toldinas, J., Venčkauskas, A., Damaševičius, R., Grigaliūnas, Š., Morkevičius, N., Baranauskas, E., "A novel approach for network intrusion detection using multistage deep learning image recognition", *Electronics*, Aug 1, Vol. 10(15), pp. 1854, 2021.





# Meta-Learning for Medium-Shot Sparse Learning via Deep Kernels<sup>\*</sup>

Research Article

Zohreh Adabi-Firuzjaee<sup>1</sup>

Sayed Kamaledin Ghiasi-Shirazi<sup>2</sup> 

**Abstract:** Few-shot learning assumes that we have a very small dataset for each task and trains a model on the set of tasks. For real-world problems, however, the amount of available data is substantially much more; we call this a medium-shot setting, where the dataset often has several hundreds of data. Despite their high accuracy, deep neural networks have a drawback as they are black-box. Learning interpretable models has become more important over time. This study aims to obtain sample-based interpretability using the attention mechanism. The main idea is reducing the task training data into a small number of support vectors using sparse kernel methods, and the model then predicts the test data of the task based on these support vectors. We propose a sparse medium-shot learning algorithm based on a metric-based Bayesian meta-learning algorithm whose output is probabilistic. Sparsity, along with uncertainty, effectively plays a key role in interpreting the model's behavior. In our experiments, we show that the proposed method provides significant interpretability by selecting a small number of support vectors and, at the same time, has a competitive accuracy compared to other less interpretable methods.

**Keywords:** Bayesian Meta-learning, Medium-shot Learning, Sample-based Interpretability, Sparse Kernel, Attention

## 1. Introduction

So far, two approaches for deep learning have received more attention. The first approach is deep learning on a large dataset, which has been more successful than other machine learning methods in image, language, and signal processing [1]. In deep learning, as it is difficult for humans to analyze a huge amount of data, one tries to train deep neural networks with it so that the information in the data could be exploited through interaction with the model. We need a massive amount of data to use deep learning, but in most real-world problems the amount of labeled data is not enough to train a deep model. The second approach is known as few-shot learning [2]. It aims to make deep learning models like humans and learn new concepts well by seeing a few examples [3].

In few-shot learning, the assumption is that the number of training data is very small. For example, in few-shot classification, the number of data for each class ranges between one and five. This assumption is contrary to the fact that in real-world problems, we easily have more data for each task, or it is even possible for the user to provide a few hundred samples. Therefore, many practical problems such as classification of medical images [4] and time series prediction are naturally in the medium-shot setting. Medium-shot learning is an extension of few-shot learning in terms of

the number of data. In recent years, meta-learning methods have shown remarkable performance in solving few-shot learning problems [5]. In this paper, we consider meta-learning methods for the case of medium-shot setting.

Deep neural networks have attracted widespread attention due to their ability to obtain high accuracies in various problems. However, there is a serious debate about them related to interpretability [6]: to what extent and on what basis can we trust the response of neural networks? Because the nature of deep networks is black-box, many methods have been proposed to interpret neural networks and their decision-making [7]. In problems where the model has to make a decision, the user wants to know why the model has made this decision. The decision of the model can be described in different ways. One of these methods is that the model determines based on the data it has made its decision. Therefore, the user can determine the quality of a decision by examining the samples that the model has selected.

The medium size of the data in the medium-shot setting provides us with the possibility and opportunity of interpretation based on the evaluation of the entire training data of the task. Our goal is to train a model in such a way that it determines which data have a more important role in its decision-making, and we consider these data as support vectors. Our idea to achieve this kind of interpretability is to follow the perspective of attention in deep learning. We want to learn which data to pay more attention to. For this purpose, we present an interpretable meta-learning algorithm. We start our work with Deep Kernel Transfer (DKT), a metric-based meta-learning algorithm [8]. DKT is a Gaussian process with a deep kernel, so it combines the representational power of neural networks and the reliable uncertainty of Gaussian processes simultaneously. To implement the attention mechanism, we use sparse kernel methods and extend the DKT algorithm to the medium-shot setting. By sparsifying the expansion of the decision function, we can have sample-based interpretability with the selected data as support vectors. The resulting algorithm, Sparse DKT, reduces the data to a small number of support vectors for each task. In the Sparse DKT algorithm, only the support vectors at the test time directly influence the prediction of the test data label. The experimental results show that Sparse DKT, in addition to interpretability, has comparable accuracy to other state-of-the-art meta-learning methods, including the DKT algorithm.

The main contributions of this article are:

1. Introducing learning with the medium-shot setting and utilizing deep meta-learning algorithms for it;
2. Learning a sample-based interpretable model using the attention mechanism;
3. Applying sparse kernel methods for determining a small

<sup>\*</sup> Manuscript received; 05 July 2022, Revised, 06 October 2022, Accepted, 09 October 2022.

<sup>1</sup> PhD student, Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran.

<sup>2</sup> Corresponding author, Assistant Professor, Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran.

Email: k.ghiasi@um.ac.ir

subset of training data as support vectors.

The remaining structure of this article is as follows: in section 2, the basic concepts about meta-learning, interpretability, attention perspective, sparse kernel, and related works are described. In section 3, the proposed algorithm is presented. The evaluation of the presented algorithm in classification will be in the section 4. In section 5, conclusion and future works are presented.

## 2. Preliminaries

### 2.1. Meta-learning

Meta-learning is one of the areas that has received attention in recent years [9-34]. In classic learning, in order to learn a task, the model is trained on the task data in such a way that it has a good generalization of the new data. The objective of meta-learning, also known as learning to learn, is to go to a higher level and understand how to solve tasks rather than just learning a single task (Figure 1). Humans face with different issues over time and develop better ways to deal with new ones by drawing on their experiences. Similar to humans, we should train the model on a set of tasks from the same distribution sequentially in meta-learning. By completing each task, we acquire metadata that the model can use to learn a new, unseen task more effectively and quickly.

### A. Meta-learning setup

In meta-learning, as shown in Figure 2, instead of one task, we have a set of tasks,  $\mathcal{M} = \{\mathcal{D}_\tau\}_{\tau=1}^T$ , which are from the same distribution. According to Figure 2, for each task, indexed by  $\tau$ , we have the data  $\mathcal{D}_\tau = \{X, y\}$ , which can be divided into two parts, the train/support set,  $\mathcal{D}_\tau^{tr}$ , and the test/query set,  $\mathcal{D}_\tau^{ts}$ . The test data that is used for meta-test is denoted by the asterisk symbol as  $\mathcal{D}_* = \{\mathcal{D}_*^{tr}, \mathcal{D}_*^{ts}\}$ .

### B. Few-shot learning

Few-shot learning refers to tasks with a few training data. For example, in the few-shot classification represented as N way - K shot, N is the number of classes in the task, and K (usually considered 1 or 5) training samples are available for each class (Figure 2 shows 3 way- 2 shot classification). Few-shot learning aims to make deep neural networks capable of learning a new concept by observing a small number of training samples. The small amount of training data makes it infeasible to train the deep neural network, but the meta-learning approach has achieved significant improvements in few-shot learning. Deep meta-learning learns a model that can solve a new task despite the small training data. Medium-shot learning is a generalization of few-shot learning, so we employ the meta-learning framework.

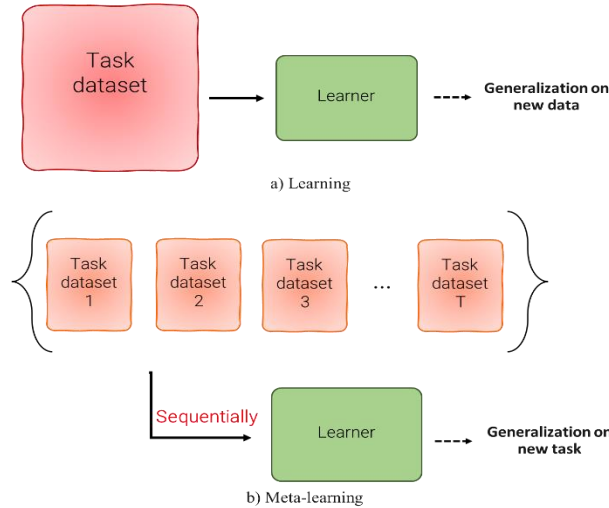


Figure 1. Difference between a) learning and b) meta-learning. In learning, training on a task data is done to generalize new data from the same dataset. In meta-learning, we train the model on a set of tasks sequentially. By learning to learn, we can solve the new task more efficiently and quickly.

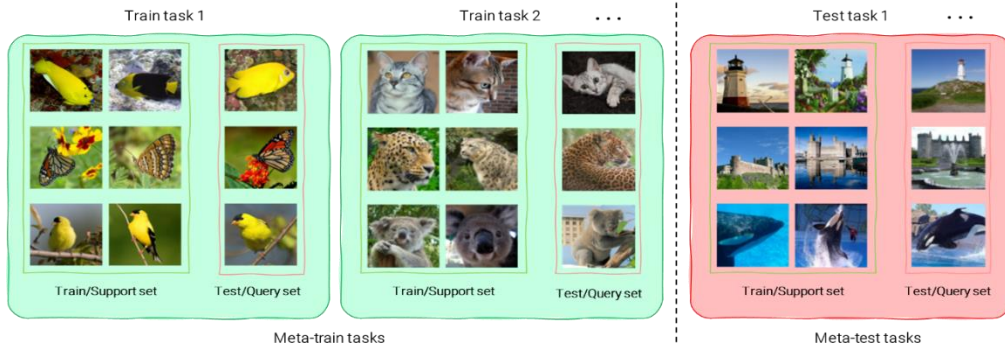


Figure 2. An example of a meta-learning setup for few-shot learning. The set of tasks  $\mathcal{M} = \{\mathcal{D}_\tau\}_{\tau=1}^T$  is divided into two parts, meta-train and meta-test. The data of each task has train and test sets,  $\mathcal{D}_\tau^{tr}$  and  $\mathcal{D}_\tau^{ts}$  respectively

## 2.2. Interpretability in Deep Neural Networks

In deep learning, there are two main classes of approaches to explain the prediction of a model: feature-based and sample-based. In the feature-based approach, features from the input image that have a greater impact on the model's prediction are identified [35, 36]. The idea of [36] in few-shot learning has been applied in [37] to provide interpretable feature-based meta-learning.

In the sample-based approach, the data that have the most impact on the network's decision-making for test data are identified as samples to interpret its prediction (Figure 3) [38, 39]. ProtoAttend [40] trains a network that compares the input data with training data to predict it based on the attention mechanism and learn an attention weight that demonstrates the degree of similarity between them. To interpret the model's decision for the input data, the data whose weight is not zero affect the model's prediction and are selected as prototypes. Because there are a lot of data in deep learning and it is difficult to compare them all, a subset of the data is typically chosen as a candidate set, and attention weight is only learned for the candidate set. In contrast, the number of data is not large in medium-shot setting, and since we can evaluate all the training data, sample-based interpretability is possible. In order to achieve this, we proceed according to the attention point of view.

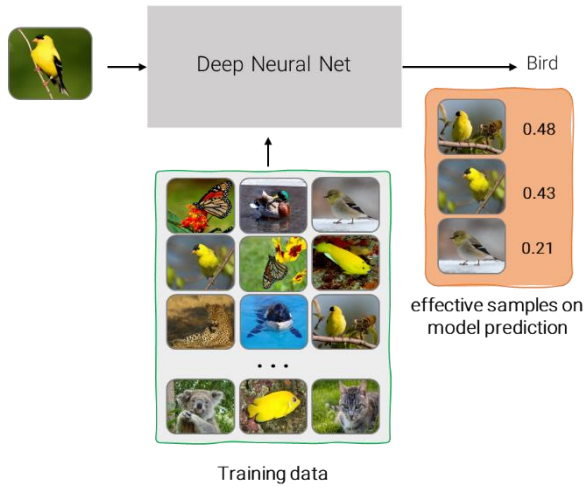


Figure 3. In sample-based interpretability, the training data that the model used to determine the label for the input data are specified.

### A. Sample-based interpretability through Attention

Using the attention perspective, we can learn a model with sample-based interpretability [41]. Simply it means to compare the input data with the training data and give greater weight to the training data that is more similar to the input data when determining its label. To compare the data properly, we need to learn a metric space in which similar data are placed close together, and dissimilar data are far apart. This method is used in the metric-based meta-learning algorithms presented for few-shot learning [14–16]. In these papers, since the number of training samples is small, there is no need for sample-based interpretability, and the main objective is to increase accuracy. Since we have more data in medium-shot learning, sample-based interpretability becomes important; in some applications, explaining the

model's behavior with a small number of samples makes it easier for humans to understand and evaluate the model.

### B. Attention and kernel methods

The attention mechanism and kernel methods are closely related [42–45]. It can be said that the idea of attention in deep learning is derived from kernel methods [42]. Kernel methods have a kernel function  $k(\mathbf{x}, \mathbf{x}')$  that determines the degree of similarity [46]. Linear kernel, polynomial, RBF (Radial Basis Function), and exponential are the well-known kernel functions. Learning the kernel function corresponds to learning its parameters, e.g., in the RBF kernel

$$k(\mathbf{x}, \mathbf{x}') = s * \exp\{-\frac{1}{l} \|\mathbf{x} - \mathbf{x}'\|^2\} \quad (1)$$

the parameters  $\phi = \{l, s\}$  are learned during training.

In deep kernel learning or DKL [47–50], we first use a deep neural network to obtain data representations, then apply a kernel function to them. The new deep kernel is

$$k(\mathbf{x}, \mathbf{x}') = \tilde{k}_{\phi}(f_{\theta}(\mathbf{x}), f_{\theta}(\mathbf{x}')) \quad (2)$$

where  $\tilde{k}_{\phi}(\mathbf{x}, \mathbf{x}')$  is the kernel function with parameter  $\phi$  and  $f_{\theta}$  is a deep neural network. DKL involves jointly learning kernel and network parameters. For example, optimization of the parameters in the regression of  $\{X, \mathbf{y}\}_{n=1}^N$  with noise variance  $\sigma^2$  is based on the log marginal likelihood,

$$\log p(\mathbf{y}|X) = \frac{1}{2} \{-\mathbf{y}^T [K + \sigma^2 I]^{-1} \mathbf{y} - \log |K + \sigma^2 I| + N \log(2\pi)\} \quad (3)$$

where  $K$  is the kernel matrix on the training data.

### 2.3. Deep Kernel Transfer

Deep Kernel Transfer or DKT falls into the category of metric-based meta-learning [8]. This class of algorithms tries to learn a metric space to compare representations based on a distance measure [14–16]. DKT is a combination of MAML (Model-Agnostic Meta-Learning) and DKL for few-shot learning. MAML [21] is based on the idea of [13] without using an additional model as a meta-learner, learns a meta-parameter as an initialization for the parameters of the network. The meta-parameter adapts quickly to the data of the new task without overfitting due to a few training data.

The computational graph of the MAML is shown in Figure 4a Using SGD (Stochastic Gradient Descent) optimization on the task training data, the MAML algorithm obtains task-specific parameter  $\phi_{\tau}$  from the meta-parameter  $\theta$ . The inner loop (adaptation loop) of the MAML has a parametric form, so in the outer loop, we encounter the second gradient of  $\theta$  with respect to the optimization path in the inner loop.

The idea of DKT is to replace the inner loop computation with a Gaussian process, which has a non-parametric form. Therefore, as shown in Figure 4b, adaptation to the task is eliminated. Similar to the DKL, a Gaussian process is applied to the representations. DKT computes the marginal likelihood (3) on the data of each task and optimizes the parameters  $\theta$  and  $\phi$ . By meta-learning a deep kernel on a set of tasks, we have a kernel that can be transferred to a new



task without needing adaptation. By replacing the inner loop with the Gaussian process, the DKT algorithm provides a computational simplification for the MAML. Furthermore, it is regarded as a Bayesian meta-learning. In the regression and image classification in few-shot settings, DKT has achieved higher accuracy than MAML and other few-shot learning methods.

#### 2.4. Sparse kernel methods

SVM (Support Vector Machine) is a popular sparse kernel method [51]. The Sparsity of SVM results from zeroing coefficient  $\alpha$  for part of the data during the quadratic optimization, which determines a subset of data as support vectors. In few-shot learning, the MetaOptNet [30] has used SVM to simplify the inner loop of MAML to obtain the task-specific parameter without SGD optimization and not to encounter the second derivative in meta-parameter optimization (Figure 4c).

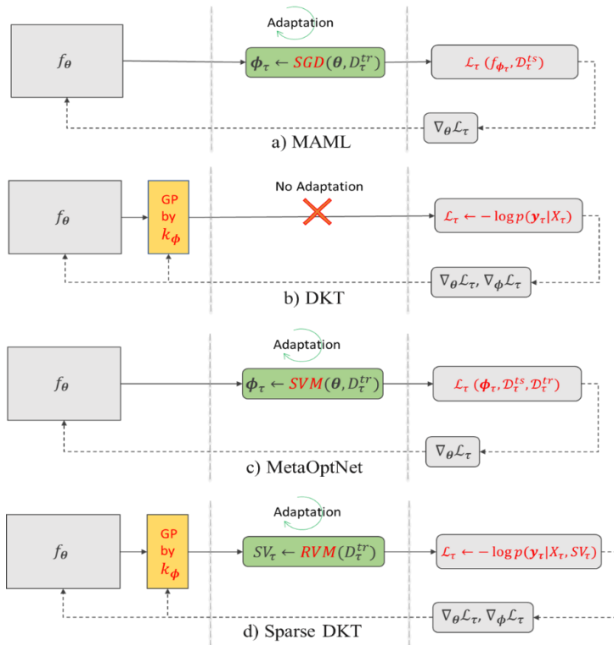


Figure 4. Computational graph of a) MAML, b) DKT, c) MetaOptNet, and d) Sparse DKT (ours). In a), adapting to the task is equivalent to obtaining the task-specific parameter  $\phi_\tau$ . In b), meta-parameters  $\theta$  and  $\phi$  without adapting to the task are updated based on the marginal likelihood of the Gaussian process on the entire data. In c), the task-specific parameter  $\phi_\tau$  is computed by applying SVM to the training data of the task. In d), adapting to the task is equivalent to specifying the support vectors, a small subset of the task training data

The disadvantage of SVM in the MetaOptNet algorithm is that it becomes less effective in sparsifying as the data increases. Another disadvantage of SVM compared to the Gaussian process [52] is that it is not probabilistic. In contrast, the Gaussian process is not inherently sparse; the kernel matrix is calculated between the test data and all  $n$  training data at test time. Several sparse approximations have been proposed to overcome the computational and memory complexity in the Gaussian process [53, 54]. Almost all of these approximation methods specify a criterion to determine the significance of the data and greedily select a subset of the data of size  $m \ll n$  to be used in the kernel matrix

approximation. The main goal of methods in [55–59] is to reduce the computational complexity of the Gaussian process by assuming that there is a set of support vectors. The criteria to determine the support vectors in these methods are usually considered for adding data to this set, so the number of support vectors is defined as a fixed hyperparameter. However, since these vectors are supposed to have the most impact on the model's prediction, we are looking for support vectors to be automatically selected with a small number and high accuracy. Additionally, in the medium-shot learning, the number of data selected as support vectors should depend on the task. Therefore, in the proposed algorithm, intending to achieve sample-based interpretability using Gaussian processes, we leverage the sparse Bayesian approach, which we will explain in the following section.

### 3. Sparse DKT for medium-shot learning

This section presents our meta-learning algorithm, Sparse DKT, for medium-shot learning. To achieve sample-based interpretability, we need to determine the importance of data in data modeling and prediction. We measure the degree of importance with the kernel function, so we use DKT. We modify this algorithm to attain sample-based interpretability and apply attention to it in two ways: attention in adaptation and attention in prediction. Attention in adaptation is independent of the test data and is performed only on the training data. The Sparse Gaussian process is trained on the task data; In other words, it adapts to it, and the result of this adaptation is the identification of support vectors.

In contrast, attention in prediction depends on the test data but uses only support vectors from the entire training data. Due to the usage of Gaussian processes, we already have attention in prediction; that is, support vectors affect test label prediction based on how similar they are to it. We discuss the proposed algorithm for regression, but it can be easily generalized for classification.

#### 3.1. Sparse Gaussian process as Adaptation

In the sparse Bayesian learning framework, Tipping introduces the RVM algorithm (Relevance Vector Machine) [60]. The advantage of this algorithm we adopted for our proposed algorithm is that it automatically selects the data that play the main role in data modeling when adapting to the task.

This algorithm is essentially a Gaussian process. Assume that we have data  $\{X, \mathbf{y}\}$ , including the inputs  $X = \{\mathbf{x}_j\}_{j=1}^n$  and the labels  $\mathbf{y} = \{y_j\}_{j=1}^n$ . Labels have Gaussian noise  $\epsilon_j \sim \mathcal{N}(0, \sigma^2)$  added to latent function  $f(\mathbf{x})$  according to  $y(\mathbf{x}_j) = f(\mathbf{x}_j) + \epsilon_j$ . The prior knowledge on the function  $f(\mathbf{x})$  is a Gaussian process  $\mathcal{GP}(\mu, k_\phi)$  with mean  $\mu$  and kernel function  $k_\phi$ . The mean is usually considered zero.

We can rewrite the latent function  $\mathbf{f}$  in the parametric form  $\mathbf{f} = \mathbf{K}\mathbf{w}$  in the equation  $\mathbf{y} = \mathbf{f} + \boldsymbol{\epsilon}$ .  $\mathbf{K}$  is the covariance matrix based on the kernel function  $k_\phi(\mathbf{x}, \mathbf{x}')$ . In the Gaussian process, the weight  $\mathbf{w}$  has a Gaussian distribution  $\mathcal{N}(0, \alpha_0^{-1}I)$ , where  $\alpha_0$  is a hyperparameter. In RVM, Gaussian distribution  $p(\mathbf{w}|\boldsymbol{\alpha}) = \mathcal{N}(0, A^{-1})$  is considered for weights, where  $A = \text{diag}(\boldsymbol{\alpha})$  is a diagonal covariance matrix. As a result, RVM is a Gaussian process with kernel

function:

$$c(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^n \frac{1}{\alpha_i} k_{\phi}(\mathbf{x}, \mathbf{x}_i) k_{\phi}(\mathbf{x}', \mathbf{x}_i) \quad (4)$$

where  $k_{\phi}(\mathbf{x}, \mathbf{x}_i)$  is equal to the kernel function that is defined based on the training data  $\mathbf{x}_i$ .  $c(\mathbf{x}, \mathbf{x}')$  is an expansion of the product of values of the kernel function  $k_{\phi}(\mathbf{x}, \mathbf{x}_i)$  in which all data contribute. The kernel function of the data that will be included in the expansion is determined by coefficient  $\alpha_i$ . When  $\alpha_i$  goes to infinity, the kernel function corresponding to  $\mathbf{x}_i$  data is removed; as a result, the expansion  $c(\mathbf{x}, \mathbf{x}')$  becomes sparse. The covariance matrix of RVM can be expressed as  $C = KA^{-1}K^T$ .

The next step is that based on Bayes Equation 5, and having likelihood  $p(\mathbf{y}|\mathbf{w}) = \mathcal{N}(\mathbf{f}, \sigma^2 I)$ ,

$$p(\mathbf{w}|\alpha, \mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{w})p(\mathbf{w}|\alpha)}{p(\mathbf{y})} \quad (5)$$

obtain the posterior distribution of the weight,

$$p(\mathbf{w}|\alpha, \mathbf{y}) = \mathcal{N}(\boldsymbol{\mu}, \Sigma)$$

$$\boldsymbol{\mu} = \sigma^{-2} \Sigma K^T \mathbf{y}$$

$$\Sigma = (A + \sigma^{-2} K^T K)^{-1}. \quad (6)$$

RVM training is similar to Gaussian process training; We optimize the logarithm of the marginal likelihood (7) with respect to the hyperparameters  $\alpha$  and  $\sigma^2$ .

$$p(\mathbf{y}) = \mathcal{N}(0, C + \sigma^2 I)$$

$$\log p(\mathbf{y}) =$$

$$-1/2\{\mathbf{y}^T [C + \sigma^2 I]^{-1} \mathbf{y} + \log |C + \sigma^2 I| + n \log 2\pi\} \quad (7)$$

By deriving the Equation 7 with respect to  $\alpha$  and  $\sigma^2$  and setting them equal to zero, optimization equations are obtained as follows:

$$\alpha_i^{new} = \frac{\gamma_i}{\mu_i^2}$$

$$\gamma_i = 1 - \alpha_i \Sigma_{ii}$$

$$(\sigma^{-2})^{new} = \frac{\|\mathbf{y} - K\boldsymbol{\mu}\|^2}{n - \sum_j \gamma_j} \quad (8)$$

where  $\Sigma_{ii}$  is the  $i$ -th diagonal component of the covariance matrix  $\Sigma$  in (6).  $\gamma_i \in [0,1]$  indicates how much the data contributed to the determination of  $w_i$ . To get  $\alpha$  and  $\sigma^2$ , we can use an iterative algorithm. During training, many  $\alpha_i$  become infinite, which causes variance and mean corresponding to their weights to be zero. When weight  $w_i$  becomes zero, the kernel function at  $\mathbf{x}_i$  does not contribute to describing the data so that it can be removed from the model. The data that have non-zero weight are considered as support vectors. Another method to train RVM is to use the Expectation-Maximization algorithm [61]. In this study, we use the sequential algorithm proposed in the [62] (The authors of [62] published their code in MATLAB, and we re-implemented it with Python. <http://www.miketipping.com/sparsebayes.htm>). In this algorithm, the set of support vectors is initially empty, and

important data are added to this set sequentially. The computational cost of RVM is significantly decreased by using this addition method, which is better for learning in the medium-shot setting.

### 3.2. Sparse DKT algorithm

The Sparse DKT algorithm using RVM as the inner loop, on the one hand, is a simplification for the MAML; on the other hand, it adds interpretability to the DKT. According to Figure 4, the difference between DKT and Sparse DKT is the addition of the adaptation loop. Unlike MetaOptNet, in Sparse DKT, the parameters of the kernel function are part of the meta-parameters and are updated by loss of each task. Sparse DKT Pseudocode is given in Algorithm 1. In meta-training, what is important for us from utilizing the RVM algorithm as the inner loop of Sparse DKT is to obtain  $\alpha$ . We are interested in learning which data are most important in describing the whole data and consequently in the model's prediction. The Sparse DKT algorithm selects the data whose  $\alpha$  coefficient is not infinite as task support vectors. In the outer loop, they are used in the optimization with RVM marginal likelihood (7).

Algorithm 1. Sparse Deep Kernel Transfer (Sparse DKT)

**Require:**  $\mathcal{M} = \{\mathcal{D}_\tau\}_{\tau=1}^T$  meta-train tasks

**Require:**  $\phi$  kernel hyperparameters,  $\theta$  neural network weights

**Require:**  $\beta_1, \beta_2$  step size

```

1: while not done do
2:   Sample  $\mathcal{D}_\tau$  from  $\mathcal{M}$ 
3:    $\text{SV} = \text{RVM}(\mathcal{D}_\tau)$  //Obtain support vectors of  $\mathcal{D}_\tau$  with RVM
4:   //Use marginal likelihood to update parameters
5:    $\mathcal{L}_\tau = -\log p(\mathbf{y}|\mathbf{X}, \phi, \theta)$  //Eq (7)
6:    $\phi \leftarrow \phi - \beta_1 \nabla_\phi \mathcal{L}_\tau, \theta \leftarrow \theta - \beta_2 \nabla_\theta \mathcal{L}_\tau$ 
7: end while
8: function RVM( $\mathcal{D}$ )
9:   //Automatically select support vectors
   //of the dataset  $\mathcal{D}$ 
10:  Initialize  $\alpha$  and  $\sigma^2$ 
11:  while not converged:
12:    Update  $\boldsymbol{\mu}$  and  $\Sigma$  //Eq (6)
13:    Update  $\alpha$  and  $\sigma^2$  // (8)
14:  return support vectors from  $\mathcal{D}$  for finite  $\alpha_i$  values
15: end function
```

At the meta-test time, for the test task with data  $\mathcal{D}_*^{tr} = \{\mathbf{X}, \mathbf{y}\}$  and  $\mathcal{D}_*^{ts}$ , the support vectors of the task are first selected from the training data  $\mathcal{D}_*^{tr}$  by running RVM. In addition to the support vectors, the mean and covariance of the posterior weight distribution are also obtained, which we use in the RVM prediction distribution,

$$p(\mathbf{y}_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*) = \mathcal{N}(\boldsymbol{\mu}_*, \sigma_*^2)$$

$$\boldsymbol{\mu}_* = \mathbf{k}_* \boldsymbol{\mu}, \quad \boldsymbol{\mu} = \sigma^{-2} \Sigma K_{mn} \mathbf{y}$$

$$\sigma_*^2 = \sigma^2 + \mathbf{k}_* \Sigma \mathbf{k}_*, \quad \Sigma = (A + \sigma^{-2} K_{mn} K_{nm})^{-1} \quad (9)$$

where  $\mathbf{k}_*$  is the covariance between  $\mathbf{x}_* \in \mathcal{D}_*^{ts}$  and  $m$  support vectors.  $K_{mn}$  is the covariance between support vectors and training data.

#### 4. Experiments

We run classification tests using common datasets in few-shot learning in a medium-shot setting to evaluate the Sparse DKT algorithm. The number of samples has been chosen in such a way that we get out of the few-shot mode. We used PyTorch and GPyTorch [63] for the implementation of the Sparse DKT.

To compare Sparse DKT with DKT, Feature Transfer, MAML, and MetaOptNet, we have considered Omniglot, CUB-200, and miniImageNet dataset for image classification (Figure 5).

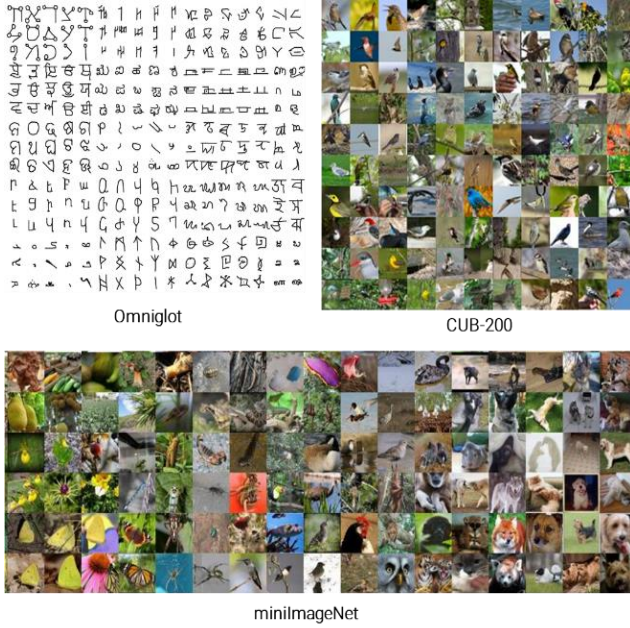


Figure 5. Images from datasets used in classification

Omniglot consists alphabet of 50 languages and has 20 hand-written samples for each character. CUB-200 contains 200 classes of different bird species. MiniImageNet has 100 classes which is a subset of ImageNet classes. Each class has 600 images. We run 2-way and 5-way classifications test. As in the DKT paper, classification is done one-versus-rest (Figure 6), i.e., for each class, we consider a binary Gaussian process model with labels  $\{-1, 1\}$  and apply the sigmoid function to its output in order to have a probabilistic interpretation (for MetaOptNet, we also used binary SVMs for multi-class classification in experiments). The model whose output has the highest probability determines the class of the test data. We used a linear kernel in experiments and a deep neural network that has a similar architecture to the network used in the DKT paper (Figure 7).

In Feature Transfer, a network and classifier are first trained on samples for the training classes. When fine-tuning, the network parameters are fixed, and a new classifier is trained on the test classes. MAML depends on the number of gradient steps in the inner loop and has low accuracy at a few steps. Increasing the gradient steps also leads to an increase in computation and memory consumption. In order to be able to test MAML in 10 steps adaptation, we used its first order approximation [28]. Table 1 shows the result of Omniglot 5 way- 15 shot classification.

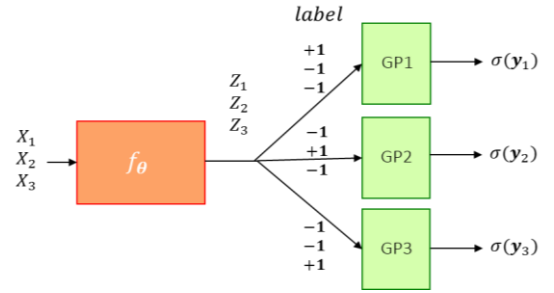


Figure 6. One-versus-rest scheme. Each model is a binary classifier for input data with labels  $\{-1, 1\}$ . For a probabilistic output, a sigmoid function  $\sigma$  is applied to it.

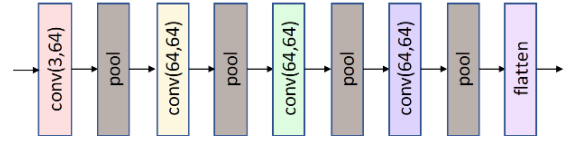


Figure 7. The CNN used as a backbone for classification. It consists of 4 convolutional layers, each consisting of a 2D convolution, a batch-norm layer, and a ReLU non-linearity.

Table 1. Average accuracy and standard deviation on Omniglot classification with average number of support vectors

Method	Omniglot 5 way - 15 shot	SVs
Feature Transfer	99.36 $\pm$ 0.08	-
MAML	95.80 $\pm$ 0.312	-
DKT	99.52 $\pm$ 0.211	75
MetaOptNet	99.46 $\pm$ 0.141	13
Sparse DKT	99.33 $\pm$ 0.1	6

Sparse DKT is more accurate than MetaOptNet and close to DKT, while DKT uses all training data of 5 classes as support vectors for its prediction. MAML can achieve more accuracy at the cost of more adaptation steps. Table 2 shows the classification results of CUB and miniImageNet. Due to the limited resources in this section, we had to run 2-way classification. The number of task training data in CUB and miniImageNet is 50 and 125, respectively. Feature transfer overfits in the few-shot setting. However, it was able to get higher accuracy than other methods in our experiments. We believe that the accuracy of Feature Transfer decreases when the new task's classes diverge more from the training classes. We leave further investigations to future works.

Sparse DKT is more interpretable and has higher accuracy than MetaOptNet, with a smaller number of support vectors. The efficiency of MetaOptNet in sparsity decreases with the increase of training data due to the weakness of SVM. In miniImageNet classification, the proposed method has selected 14 support vectors on average from 250 data, while MetaOptNet has selected 76 support vectors. Additionally, the experiments on these different datasets show that the number of support vectors for each application depends on intra-class and inter-class similarity. The metric space learned by the Sparse DKT to separate classes affects the number of support vectors.

In Figure 8, we have given an example of testing the trained model with the Sparse DKT and DKT on a 2 way – 50 shot classification task from the CUB meta-test dataset.







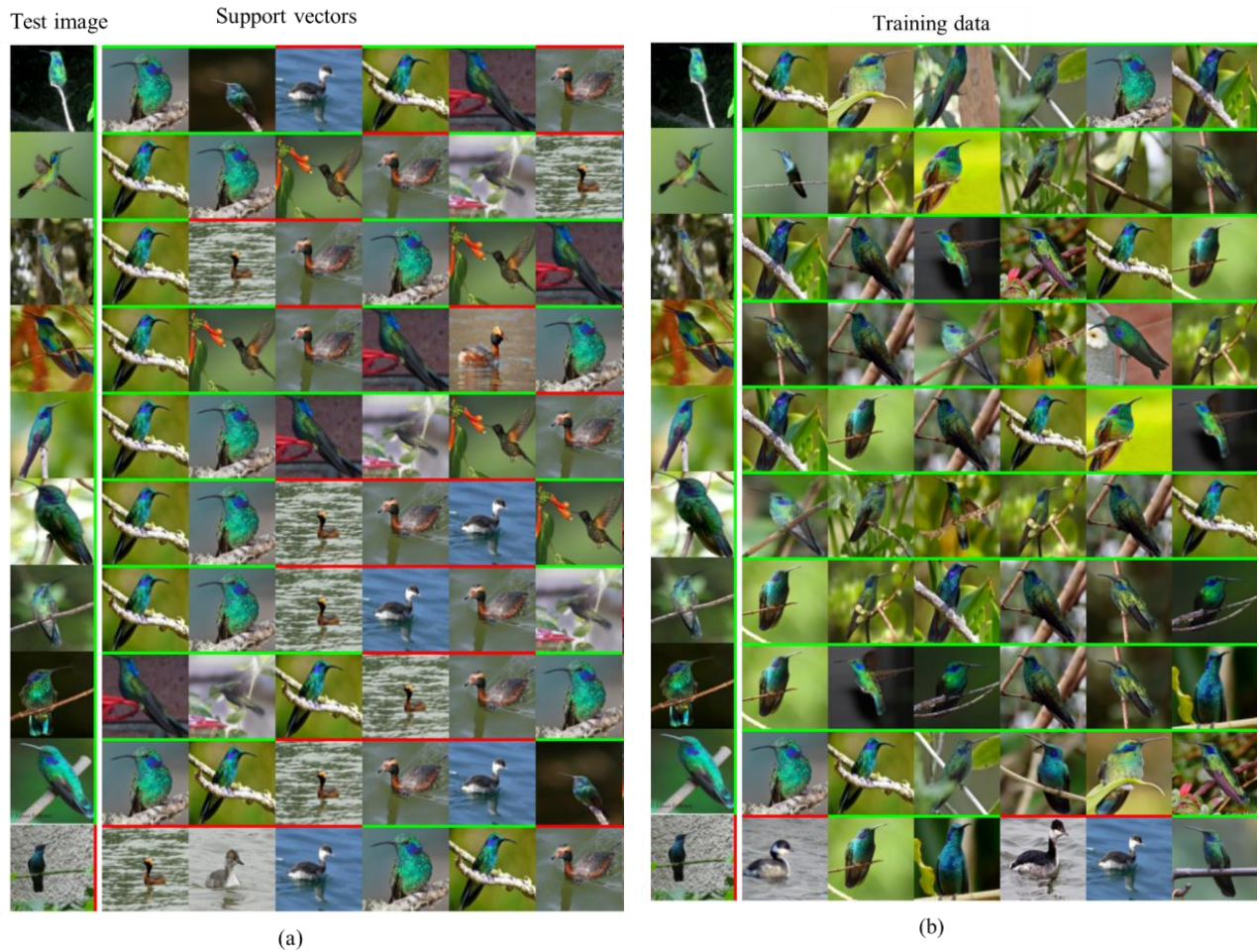


Figure 10. Comparing kernels of a) Sparse DKT and b) DKT: a) the most similar support vectors to test image, b) the most similar training data to test images. The green line above the images on the right, shows that they have the same label as the test image.

We compared the learned kernels of Sparse DKT and DKT in Figure 10. For Sparse DKT similarity of test images to support vectors is computed. In each row, the images are sorted in the order of the most similar from left to right. The green and red lines on top of the right image, show whether

## 5. Conclusion

In this study, we introduced medium-shot learning as a generalization of few-shot learning for real-world applications. Considering that interpretability in deep learning models is becoming increasingly more important, especially in sensitive scenarios, sample-based interpretability can be easily obtained by reducing the data to a small number of support vectors in medium-shot learning. We considered sparse kernel methods from an attention-based perspective to have sample-based interpretability. The proposed Sparse DKT algorithm leverages Sparse Gaussian processes in the meta-learning framework and selects the most important training data as support vectors. At the test time, it makes the predictions based on support vectors.

The impact of marginal likelihood in the trade-off between accuracy and the number of support vectors, as well as the impact of more task training data, is one of the key areas for future work. Using improved versions of RVM [64, 65] would be effective in increasing the accuracy of Sparse DKT. Since SVM in MetaOptNet is less effective in sparsifying,

the labels of the right images are the same or different from those of the test image. The vertical green line in the test image indicates that the model accurately predicted the label. The Sparse DKT kernel can detect the similarity well, even though the number of support vectors is very small.

When data increases, we can use GLASSO [66], which also has a probabilistic solution, as an alternative to SVM in MetaOptNet. Another future work is investigating variational sparse Gaussian processes [67–70] that use variational inference for increasing the lower bound of the marginal likelihood algorithm. We can use the combination of point processes [71] with it to determine the support vectors in sparse variational Gaussian processes.

## 6. References

- [1] Goodfellow, I., Bengio, Y., and Courville, A., Deep learning, MIT press, 2016.
- [2] Wang, Y., Yao, Q., Kwok, J., and Ni, L. M., "Generalizing from a Few Examples: A Survey on Few-Shot Learning", *ACM Computing Surveys (CSUR)*, Vol. 53, No. 3, Apr. 2019, Accessed: Jan. 23, 2022. [Online]. Available: <http://arxiv.org/abs/1904.05046>.
- [3] Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B., "Human-level concept learning through probabilistic program induction", *Science*, Vol. 350, No. 6266, pp. 1332–8, doi: 10.1126/science.aab3050, Dec. 2015.

- [4] Jiang, X., Ding, L., Havaei, M., Jesson, A., and Matwin, S., "Task Adaptive Metric Space for Medium-Shot Medical Image Classification", in *MICCAI*, pp. 147–155. doi: 10.1007/978-3-030-32239-7\_17, 2019.
- [5] Li, X., Sun, Z., Xue, J.-H., and Ma, Z., "A concise review of recent few-shot meta-learning methods", *Neurocomputing*, Vol. 456, pp. 463–468, doi: 10.1016/j.neucom.2020.05.114, Oct, 2021.
- [6] Samek, W., Montavon, G., Vedaldi, A., Hansen, L. K., and Müller, K.-R., *Explainable AI: interpreting, explaining and visualizing deep learning*, Vol. 11700. Springer Nature, 2019.
- [7] Zhang, Y., Tino, P., Leonardis, A., and Tang, K., "A Survey on Neural Network Interpretability", *IEEE Transactions on Emerging Topics in Computational Intelligence*, Vol. 5, No. 5, pp. 726–742, doi: 10.1109/TETCI.2021.3100641, Oct, 2021.
- [8] Patacchiola, M., Turner, J., Crowley, E. J., O'Boyle, M., and Storkey, A., "Bayesian Meta-Learning for the Few-Shot Setting via Deep Kernels", *34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, Oct. 2019, Accessed: Feb. 04, 2021. [Online]. Available: <http://arxiv.org/abs/1910.05199>.
- [9] Huisman, M., van Rijn, J. N., and Plaat, A., "A survey of deep meta-learning", *Artificial Intelligence Review*, Vol. 54, No. 6, pp. 4483–4541, doi: 10.1007/s10462-021-10004-4, Aug. 2021.
- [10] Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., and Lillicrap, T., "One-shot Learning with Memory-Augmented Neural Networks", *33rd International Conference on Machine Learning, ICML 2016*, Vol. 4, pp. 2740–2751, May 2016, Accessed: Oct. 16, 2019. [Online]. Available: <http://arxiv.org/abs/1605.06065>
- [11] Munkhdalai, T., and Yu, H., "Meta Networks", *34th International Conference on Machine Learning, ICML*, Vol. 5, pp. 3933–3943, Mar. 2017, Accessed: Oct. 18, 2019. [Online]. Available: <http://arxiv.org/abs/1703.00837>
- [12] Andrychowicz, M., *et al.*, "Learning to learn by gradient descent by gradient descent", *Advances in Neural Information Processing Systems*, pp. 3988–3996, Jun. 2016, Accessed: Oct. 16, 2019. [Online]. Available: <http://arxiv.org/abs/1606.04474>
- [13] Ravi, S., and Larochelle, H., "Optimization as a model for few-shot learning", 2017.
- [14] Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., and Wierstra, D., "Matching Networks for One Shot Learning", *Advances in Neural Information Processing Systems*, pp. 3637–3645, Jun. 2016, Accessed: Oct. 18, 2019. [Online]. Available: <http://arxiv.org/abs/1606.04080>
- [15] Snell, J., Swersky, K., and Zemel, R. S., "Prototypical Networks for Few-shot Learning", *Advances in Neural Information Processing Systems*, Vol. 2017-Decem, pp. 4078–4088, Mar. 2017, Accessed: Oct. 18, 2019. [Online]. Available: <http://arxiv.org/abs/1703.05175>
- [16] Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P. H. S., and Hospedales, T. M., "Learning to Compare: Relation Network for Few-Shot Learning", in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1199–1208. doi: 10.1109/CVPR.2018.00131, Jun, 2018.
- [17] Gordon, J., Bronskill, J., Bauer, M., Nowozin, S., and Turner, R. E., "Meta-Learning Probabilistic Inference For Prediction", *7th In International Conference on Learning Representations, ICLR*, May 2018, Accessed: Jan. 23, 2022. [Online]. Available: <http://arxiv.org/abs/1805.09921>
- [18] Finn, C., Xu, K., and Levine, S., "Probabilistic Model-Agnostic Meta-Learning", *Advances in Neural Information Processing Systems*, pp. 9516–9527, Jun. 2018.
- [19] Garnelo, M., *et al.*, "Conditional Neural Processes", *35th International Conference on Machine Learning, ICML 2018*, Vol. 4, pp. 2738–2747, Jul. 2018, Accessed: Apr. 21, 2020. [Online]. Available: <http://arxiv.org/abs/1807.01613>
- [20] Edwards, H., and Storkey, A., "Towards a Neural Statistician", *5th International Conference on Learning Representations, ICLR*, Accessed: Apr. 26, 2020, Jun, 2016. [Online]. Available: <http://arxiv.org/abs/1606.02185>
- [21] Finn, C., Abbeel, P., and Levine, S., "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks", *34th International Conference on Machine Learning, ICML 2017*, Vol. 3, pp. 1856–1868, Mar. 2017, Accessed: Oct. 19, 2019. [Online]. Available: <http://arxiv.org/abs/1703.03400>
- [22] Kim, T., Yoon, J., Dia, O., Kim, S., Bengio, Y., and Ahn, S., "Bayesian Model-Agnostic Meta-Learning", *Advances in Neural Information Processing Systems*, pp. 7332–7342, Jun. 2018.
- [23] Grant, E., Finn, C., Levine, S., Darrell, T., and Griffiths, T., "Recasting Gradient-Based Meta-Learning as Hierarchical Bayes", *6th International Conference on Learning Representations, ICLR*, Jan. 2018.
- [24] Ravi, S., and Beatson, A., "Amortized bayesian meta-learning", 2018.
- [25] Raghu, A., Raghu, M., Bengio, S., and Vinyals, O., "Rapid Learning or Feature Reuse? Towards Understanding the Effectiveness of MAML", *8th International Conference on Learning Representations, ICLR 2020*, Sep. 2019.
- [26] Oh, J., Yoo, H., Kim, C., and Yun, S., "BOIL: Towards Representation Change for Few-shot Learning", 2021.
- [27] Zintgraf, L., Shiarlis, K., Kurin, V., Hofmann, K., and Whiteson, S., "Fast context adaptation via meta-learning", in *36th International Conference on Machine Learning, ICML 2019*, Vol. 2019-June, pp. 13262–13276, 2019.
- [28] Nichol, A., Achiam, J., and Schulman, J., "On First-Order Meta-Learning Algorithms", Mar. 2018, Accessed: Apr. 13, 2020. [Online]. Available: <http://arxiv.org/abs/1803.02999>
- [29] Rajeswaran, A., Finn, C., Kakade, S., and Levine, S., "Meta-Learning with Implicit Gradients", *Advances in Neural Information Processing Systems 32(pp. 113-124)*, Sep. 2019, Accessed: Aug. 16, 2020. [Online]. Available: <http://arxiv.org/abs/1909.04630>
- [30] Lee, K., Maji, S., Ravichandran, A., and Soatto, S., "Meta-Learning With Differentiable Convex Optimization", in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10649–10657. doi: 10.1109/CVPR.2019.01091, Jun,

- 2019.
- [31] Bertinetto, L., Henriques, J. F., Torr, P. H. S., and Vedaldi, A., "Meta-learning with differentiable closed-form solvers", *7th In International Conference on Learning Representations, ICLR*, May 2018, [Online]. Available: <http://arxiv.org/abs/1805.08136>
  - [32] Gai, S., and Wang, D., "Sparse Model-Agnostic Meta-Learning Algorithm for Few-Shot Learning", in *2019 2nd China Symposium on Cognitive Computing and Hybrid Intelligence (CCHI)*, pp. 127–130, Sep. 2019. doi: 10.1109/CCHI.2019.8901909.
  - [33] Madan, A., and Prasad, R., "B-Small: A Bayesian Neural Network Approach to Sparse Model-Agnostic Meta-Learning", *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2730–2734, 2021.
  - [34] Tian, H., Liu, B., Yuan, X. -T., and Liu, Q., "Meta-learning with Network Pruning", in *Computer Vision – ECCV 2020. Lecture Notes in Computer Science*, Springer, Cham, pp. 675–700. doi: 10.1007/978-3-030-58529-7\_40, 2020.
  - [35] Sabzevar, R. Z., Ghiasi-Shirazi, K., and Harati, A., "Prototype-based interpretation of the functionality of neurons in winner-take-all neural networks", *ArXiv*, vol. abs/2008.08750, Aug. 2020, [Online]. Available: <http://arxiv.org/abs/2008.08750>
  - [36] Chen, C., Li, O., Tao, C., Barnett, A. J., Su, J., and Rudin, C., "This Looks Like That: Deep Learning for Interpretable Image Recognition", *Advances in Neural Information Processing Systems (NeurIPS 2018)*, Vol. 32, pp. 8930–8941, Jun. 2019, [Online]. Available: <http://arxiv.org/abs/1806.10574>
  - [37] Cao, K., Brbic, M., and Leskovec, J., "Concept Learners for Few-Shot Learning", *International Conference on Learning Representations (ICLR)*, 2021.
  - [38] Koh, P. W., and Liang, P., "Understanding Black-box Predictions via Influence Functions", *International Conference on Machine Learning*, pp. 1885–1894, Mar. 2017, [Online]. Available: <http://arxiv.org/abs/1703.04730>
  - [39] Yeh, C. -K., Kim, J. S., Yen, I. E. H., and Ravikumar, P., "Representer Point Selection for Explaining Deep Neural Networks", *Advances in Neural Information Processing Systems*, Vol. 31, Nov. 2018, [Online]. Available: <http://arxiv.org/abs/1811.09720>
  - [40] Arik, S. O., and Pfister, T., "ProtoAttend: Attention-Based Prototypical Learning", *Journal of Machine Learning Research*, Vol. 21, pp. 1–35, Feb. 2020, [Online]. Available: <http://arxiv.org/abs/1902.06292>
  - [41] Vaswani, A., *et al.*, "Attention is all you need", in *Advances in neural information processing systems*, pp. 5998–6008, 2017.
  - [42] Tsai, Y. -H. H., Bai, S., Yamada, M., Morency, L. -P., and Salakhutdinov, R., "Transformer Dissection: An Unified Understanding for Transformer's Attention via the Lens of Kernel", *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2019.
  - [43] Chen, Y., Zeng, Q., Ji, H., and Yang, Y., "Skyformer: Remodel Self-Attention with Gaussian Kernel and Nystrom Method", *Advances in Neural Information Processing Systems*, Vol. 34, 2021.
  - [44] Song, K., Jung, Y., Kim, D., and Moon, I. -C., "Implicit kernel attention", in *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35, No. 11, pp. 9713–9721, 2021.
  - [45] Choromanski, K. M., *et al.*, "Rethinking Attention with Performers", *International Conference on Learning Representations*, 2021.
  - [46] Scholkopf, B., Smola, A. J., and Bach, F., *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press, 2018.
  - [47] Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P., "Deep Kernel Learning", *Artificial intelligence and statistics, AISTATS*, pp. 370–378, Nov. 2016, [Online]. Available: <http://arxiv.org/abs/1511.02222>
  - [48] Tossou, P., Dura, B., Laviolette, F., Marchand, M., and Lacoste, A., "Adaptive deep kernel learning", *arXiv preprint arXiv:1905.12131*, 2019.
  - [49] Salakhutdinov, R., and Hinton, G. E., "Using Deep Belief Nets to Learn Covariance Kernels for Gaussian Processes", in *NIPS*, Vol. 7, pp. 1249–1256, 2007.
  - [50] Calandra, R., Peters, J., Rasmussen, C. E., and Deisenroth, M. P., "Manifold Gaussian processes for regression", in *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 3338–3345, 2016.
  - [51] Cortes, C., and Vapnik, V., "Support-vector networks", *Machine Learning*, Vol. 20, No. 3, pp. 273–297, doi: 10.1007/bf00994018, Sep. 1995.
  - [52] Rasmussen, C. E., and Williams, C. K. I., *Gaussian Processes for Machine Learning*. The MIT Press, 2006. [Online]. Available: <http://www.gaussianprocess.org/gpml/>
  - [53] Quinero-Candela, J., and Rasmussen, C. E., "A unifying view of sparse approximate Gaussian process regression", *The Journal of Machine Learning Research*, Vol. 6, pp. 1939–1959, 2005.
  - [54] Liu, H., Ong, Y. -S., Shen, X., and Cai, J., "When Gaussian process meets big data: A review of scalable GPs", *IEEE Trans Neural Netw Learn Syst*, Vol. 31, No. 11, pp. 4405–4423, 2020.
  - [55] Smola, A., and Bartlett, P., "Sparse greedy Gaussian process regression", *Adv Neural Inf Process Syst*, Vol. 13, 2000.
  - [56] Seeger, M. W., Williams, C. K. I., and Lawrence, N. D., "Fast forward selection to speed up sparse Gaussian process regression", in *International Workshop on Artificial Intelligence and Statistics*, pp. 254–261, 2003.
  - [57] Keerthi, S. S., and Chu, W., "A matching pursuit approach to sparse Gaussian process regression", *Adv Neural Inf Process Syst*, Vol. 18, 2005.
  - [58] Snelson, E., and Ghahramani, Z., "Sparse Gaussian processes using pseudo-inputs", *Adv Neural Inf Process Syst*, Vol. 18, 2005.
  - [59] Williams, C., and Seeger, M., "Using the Nyström Method to Speed Up Kernel Machines", in *Advances in Neural Information Processing Systems*, Vol. 13, pp. 682–688, 2000.
  - [60] Tipping, M. E., "Sparse Bayesian Learning and the Relevance Vector Machine", *J. Mach. Learn. Res.*, Vol. 1, pp. 211–244, 2001.
  - [61] Bishop, C. M., *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

- [62] Tipping, M. E., and Faul, A. C., "Fast marginal likelihood maximisation for sparse Bayesian models", *International workshop on artificial intelligence and statistics*, pp. 276–283, 2003.
  - [63] Gardner, J., Pleiss, G., Weinberger, K. Q., Bindel, D., and Wilson, A. G., "Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration", *Adv Neural Inf Process Syst*, Vol. 31, 2018.
  - [64] Al-Shoukairi, M., Schniter, P., and Rao, B. D., "A GAMP-based low complexity sparse Bayesian learning algorithm", *IEEE Transactions on Signal Processing*, Vol. 66, No. 2, pp. 294–308, 2017.
  - [65] Zhou, W., Zhang, H. -T., and Wang, J., "An efficient sparse Bayesian learning algorithm based on Gaussian-scale mixtures", *IEEE Transactions on Neural Networks and Learning Systems*, 2021.
  - [66] Roth, V., "The generalized LASSO", *IEEE Trans Neural Netw*, Vol. 15, No. 1, pp. 16–28, 2004.
  - [67] Titsias, M., "Variational learning of inducing variables in sparse Gaussian processes", in *Artificial intelligence and statistics*, pp. 567–574, 2009.
  - [68] Hensman, J., Matthews, A., and Ghahramani, Z., "Scalable variational Gaussian process classification", in *Artificial Intelligence and Statistics*, pp. 351–360, 2015.
  - [69] Hensman, J., Fusi, N., and Lawrence, N. D., "Gaussian processes for Big data", in *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, pp. 282–290, 2013.
  - [70] Wilson, A. G., Hu, Z., Salakhutdinov, R. R., and Xing, E. P., "Stochastic variational deep kernel learning", *Advances in Neural Information Processing Systems*, Vol. 29, 2016.
  - [71] Uhlenholt, A. K., Charvet, V., and Jensen, B. S., "Probabilistic selection of inducing points in sparse Gaussian processes", in *Uncertainty in Artificial Intelligence*, pp. 1035–1044, 2021.
-





# The Impact of Preprocessing Techniques for Covid-19 Mortality Prediction\*

Research Article

Soodeh Hosseini<sup>1</sup>

Zahra Asghari Varzaneh<sup>2</sup>

**Abstract:** Coronavirus 2019 (COVID-19), as a common infectious disease, is spreading rapidly and uncontrollably worldwide. Therefore, early detection of mortality considering the symptoms that appear in patients with Coronavirus is important. The main aim of this study is investigating the effect of data preprocessing methods on the efficiency of data mining approaches. In this study, we propose a hybrid method based on the Covid-19 dataset to predict the mortality of 1255 patients with coronavirus that has three main steps. In the first step, preprocessing methods such as imputing missing values, data balancing, normalization, and filter-based feature selection are used on raw data. Then the classification algorithms are applied to the data and finally, the evaluation is done. The results of the proposed method show its effectiveness in predicting mortality from coronavirus disease. Therefore, doctors and treatment staff can use this model to early diagnose of factors affecting the mortality of patients and with timely treatment, the mortality rate due to Covid-19 is reduced.

**Keywords:** COVID-19, Artificial Intelligence, Data Mining, Feature Selection, Mortality Detection, Preprocessing, KNIME Tool

## 1. Introduction

CORONAVIRUS (COVID-19) is an infectious disease caused by the SARS-CoV-2 virus. The virus was first reported by the World Health Organization (WHO) in a Chinese city in late 2019, it was named the 2019 coronavirus or COVID-19. Although accurate and comprehensive information is not available due to the novelty of this virus, so far the disease has shown itself in the form of respiratory symptoms [1, 2]. Anyone can get Covid-19 disease and become seriously ill or die at any age. This disease is a new phenomenon and at the moment it is not possible to give a definite opinion about the fatality of this disease. But statistics show that the death ration is around 2%, but according to the WHO, this number can change [3, 4].

Today, due to the spread of knowledge and more complex decision-making processes, the use of information systems, especially AI systems in decision-making is more important. AI is one of the broadest branches of computer science related to the construction of intelligent machines [5]. In the field of health, AI uses sophisticated algorithms and software to analyze complex medical data. The main purpose of artificial intelligence programs in the field of health and medicine is analyzing techniques to prevent and treat disease [6]. AI is used for a variety of therapeutic and research purposes, such as diagnosis, management of chronic diseases, and medical and pharmaceutical services. With the spread of the coronavirus, AI is widely used in the diagnosis

and treatment of this disease, and researchers have been able to help medical science by using various techniques, including data mining. Clinical Decision Support Systems (CDSS) are introduced as computer programs that use ML algorithms, and AI, to help physicians make accurate and appropriate decisions [7-9]. Therefore, our goal is to develop and evaluate a new CDSS based on techniques to predict the mortality of patients with COVID-19 disease based on the decision tree, Random forest, MLP, KNN, SVM, and Fuzzy rules algorithms.

Using the collected raw data cannot provide acceptable and reliable results. Therefore, they need to be preprocessed before using. We propose a hybrid method based on the Covid-19 dataset to predict the mortality of patients. The proposed model has three steps. At first, we correct incomplete information by using missing value estimation techniques. We use the KNN Imputer to fill missing values. This method preserves the value and diversity of the dataset while being more accurate and efficient than using other methods. Then we normalize the data so that everyone is in the bound of 0 to 1. Also, since the data we are examining are unbalanced, the SMOTE technique is applied for balancing the distribution of data classes. SMOTE has the advantage of not creating duplicate data points, but rather synthetic data points that differ slightly from the original data points. Next, a filter-based feature selection method called "relief method" is used to select the best features that have the greatest impact on the performance of classification algorithms. After data preprocessing, data mining algorithms are applied and evaluated according to different criteria. Then, using statistical methods, the data mining algorithms are ranked and the best algorithm is selected.

The rest of the article is organized as follows. Some of the researches in this field are presented in Section 2. In Section 3, the proposed prediction model is covered. The evaluation and experimental results are provided in Section 4, along with a comparative analysis of the classification algorithms. Conclusions and future works will be presented in the last section of this paper.

## 2. Related works

Many methods have been recently proposed for COVID-19 diagnosis using data mining tools and machine learning algorithms to automate and help with the diagnosis and treatment of this disease. Some studies and diagnostic methods regarding COVID-19 are briefly described here.

To analyze and predict the growth of COVID-19 infection worldwide, the authors presented an improved mathematical model in [10]. This model is based on machine learning used to predict the spread of disease and is based on a cloud

\* Manuscript Received: 11 July 2022, Revised, 31 July 2022, Accepted, 29 August 2022.

<sup>1</sup>. Corresponding author. Associate Professor, Department of Computer Science, Faculty of Mathematics and Computer, Shahid Bahonar University of Kerman, Kerman, Iran. **Email:** so\_hosseini@uk.ac.ir

<sup>2</sup>. Ph. D. Student, Department of Computer Science, Faculty of Mathematics and Computer, Shahid Bahonar University of Kerman, Kerman, Iran.

computing platform. The results of the study show that the use of the Weibull model based on repetitive weighting can make more accurate statistical predictions and the weaker the fit model, the non-optimal decision and the health status will be poor. In [11], the authors used the SVM algorithm for predicting severe conditions of COVID-19. In their proposed prediction model, they searched and discovered the features that had the greatest impact on the diagnosis of mild or severe disease. The model for predicting severe disease conditions presented by the authors had an almost optimal accuracy. [12] analyzed three different classification algorithms such as Random Forest (RF), logistic regression (LR) to predict the severity of the disease in patients with coronavirus disease at King Fahad Hospital. They used the SMOTE method for balancing the data in the preprocessing phase. The models are implemented in Python language. For partitioning the data, a 10-fold cross-validation technique is used. Experiments are performed on the original dataset and the SMOTE-transformed dataset. The results of their experiments showed that the efficiency of the RF algorithm is better than other classification algorithms. [13] presented a model to predict the recovery from Covid-19. They applied data mining models such as decision trees, SVM, LR, and KNN to the data of patients with Covid-19 in South Korea. Data mining algorithms are applied directly to the dataset using python programming language to develop the models. This model is for predicting the minimum and the maximum number of days for recovery of COVID-19 patients and those at high risk for the recovery of COVID-19. The results of their research showed that the decision tree algorithm is more effective in predicting the possibility of recovery of infected patients.

An Efficient Deep Learning Technique for the screening of COVID-19 can be seen in [15]. The authors propose a vote-based design and cross-data set analysis. This approach is evaluated on two of the largest COVID-19 CT analysis datasets with patient-based division. A cross-data set review is also introduced to evaluate the robustness of the models in a more realistic scenario in which the data come from different distributions. The model is implemented in Python language. The results show that the methods that aim at COVID-19 detection in CT images have to be improved significantly to be considered as a clinical option.

Nikooghadam et al. [16] used a hybrid approach to predict and diagnose the coronavirus. The authors presented their proposed method in two steps. In the first step, they used the relief feature selection method to preprocess the data and select the effective features in the decision-making. Next, they used the ensemble-based classifier, in which the base classifier algorithms are combined to make the diagnosis with more accuracy. Basic classifiers include decision trees, KNN, combined with a random forest algorithm in the stacking section. To execute the proposed model, data mining tools including Rapid Miner and Python are used. The results proved that the combination of these algorithms can have a good effect on classification performance. In [17], it was tried to predict the mortality of COVID-19 disease in patients. In this study, they first identified the factors contributing to patient mortality. For this purpose, they reviewed various studies, and based on known factors, a

variety of classification algorithms such as SVM, random forest, J48, MLP, and KNN were applied to predict the mortality of COVID-19 disease. They used Weka v3.9.2 software to analyze the data, identify the importance of each factor, and implement prediction models. According to the results, the random forest algorithm is superior to other algorithms. In all research studied in this article, methods based on AI and data mining algorithms have been used to diagnose COVID-19 disease and predict its mortality, but what matters is the preprocessing and management of raw data. This study comprehensively examines data preprocessing methods and before applying data mining algorithms to the data, preprocessing methods were used. This step increases the efficiency of classification algorithms.

### 3. The proposed prediction model

The proposed model is a machine learning model that predicts mortality from COVID-19 in three main stages. Initially, raw data sets are collected for all those who are referred for the PCR-COVID-19 diagnostic test. Then, from the collected data, positive and negative diagnostic tests are separated. To predict mortality, only data that are in the positive diagnostic test category are needed. In the following, the raw data collected from the medical records of patients with COVID-19 disease are preprocessed. To reduce the dimensions of data and eliminate redundant features that increase the computational load and reduce the performance of classification algorithms, the feature selection method was used. The feature selection method used in this paper is based on the filter method and the reason for choosing this method is that filter-based feature selection methods are not exposed to "overfitting" and impose less computational load on the system.

After preparing the data, in the second stage, some machine learning methods are developed and used in a prospective study to predict the mortality of patients. Finally, different models for external validation are evaluated and ranked based on statistical methods and the best model is selected. Each step is explained as follows. Figure 1 briefly describes the methodology.

#### 3.1. Dataset

The dataset in this study is collected from the database of Imam Khomeini Hospital in Ilam. These data are related to those who are referred to the hospital for the PCR-COVID-19 test from February 7, 2020, to December 20, 2021. Out of a total of 6854 suspected cases of covid-19, 1853 positive cases of covid-19, 2472 negative cases, and 2529 uncertain cases are identified. Among the 1853 positive samples, unknown cases, discharge or death from the emergency room, missing data > 70%, noise, and abnormal values outside the defined time period were removed from the dataset, and 1225 cases were registered in the database. This dataset contains 54 features that include clinical features (14 features), history of personal diseases (7 features), patient's demographic (5 features), laboratory results (26 features), remedies (one feature), and an output variable (0: Life and 1: Death). Table 1 presents a list of features of Covid-19 dataset.



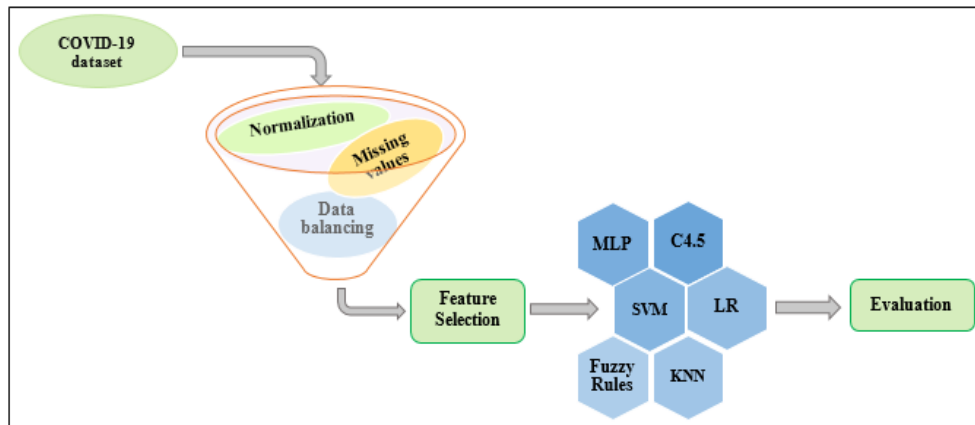


Figure 1. Overall methodology for COVID-19 mortality prediction

Table 1. List of features of Covid-19 dataset

No	Feature name	Variable type	No	Feature name	Variable type
1	Length of hospitalization	Polynomial	28	alcohol addiction	Binominal
2	Age	Polynomial	29	Creatinine	Polynomial
3	Height	Polynomial	30	Red-cell count	Polynomial
4	Weight	Polynomial	31	White-cell count	Polynomial
5	Blood Type	Polynomial	32	Hematocrit	Polynomial
6	Gender	Binominal	33	Hemoglobin	Polynomial
7	Cough	Binominal	34	Platelet count	Polynomial
8	Contusion	Binominal	35	Absolute lymphocyte	Polynomial
9	Nausea	Binominal	36	Absolute neutrophil	Polynomial
10	Vomit	Binominal	37	Calcium	Polynomial
11	Headache	Binominal	38	Phosphorus	Polynomial
12	Gastrointestinal symptoms	Binominal	39	Magnesium	Polynomial
13	Muscular pain	Binominal	40	Sodium	Polynomial
14	Chill	Binominal	41	Potassium	Polynomial
15	Hypersensitive troponin	Binominal	42	Blood ureanitrogen	Polynomial
16	Fever	Binominal	43	Total bilirubin	Polynomial
17	Oxygen therapy	Polynomial	44	Aspartate aminotransferase	Polynomial
18	Dyspnea	Binominal	45	ICU	Binominal
19	Loss of taste	Binominal	46	Albumin	Polynomial
20	Loss of smell	Binominal	47	Glucose	Polynomial
21	Runny nose	Binominal	48	Lactate dehydrogenase	Polynomial
22	Sore throat	Binominal	49	Activated partial	Binominal
23	Other underlying disease	Binominal	50	Prothrombin time	Polynomial
24	Cardiac disease	Binominal	51	Alkaline phosphatase	Polynomial
25	Hypertension	Binominal	52	C-reactive protein	Polynomial
26	Diabetes	Binominal	53	Erythrocyte sedimentation	Polynomial
27	Smoking	Binominal	54	Death	Binominal

### 3.2. Data pre-processing

The COVID-19 raw dataset contains some errors that can negatively affect the effectiveness of data mining models. Hence, to obtain the best results, we remove duplicate values from all attributes and convert raw data into numerical

features. Then, we conduct some well-defined preprocessing methods to achieve the best models.

*Imputing Missing Values:* When working with a dataset, we may encounter observations in which one or more variables or attributes have no value. This problem often

occurs if not enough care is taken when collecting data. In such cases, we say that the observations have a "missing value" or the dataset suffers from the obstacle of missing data. To impute missing values, there are many methods such as replacing the mode, mean or mean of a group [18, 19].

In the dataset used in this study, there are missing values that need to be managed. If we want to remove all the observations that have missing values from the dataset, we may face a lack of information. To address the obstacle of missing values, we delete the observations in which the number of missing values is high. Therefore, considering that the total number of data columns (features) is 54, we remove the observations in which more than 70% of the features are without value and replace the remaining missing values with the mean value of 5 nearest neighbors measured by Euclidean distance (KNN Imputer) of the non-missing values in the column. The idea in KNN Imputer method is to identify "k" similar samples in the dataset. Then we use these "k" samples to estimate the amount of missing data points. The missing values in each sample are estimated using the mean value of k of the nearest neighbors measured by Euclidean distance in the dataset. In this paper, we set the value of "k" to be 5.

There are different methods to handle missing data. These methods can waste valuable data or reduce the diversity of the dataset. In contrast, the KNN Imputer preserves the value and diversity of the dataset while being more accurate and efficient than using other methods.

**Data Balancing:** Unbalanced data class distribution occurs when the number of samples related to one class is significantly less than the number of samples belonging to another class. This will reduce the efficiency of machine learning algorithms [20]. Hence, various techniques have been introduced to deal with the problem of unbalanced data such as under-sampling, over-sampling, and Synthetic Minority Oversampling Technique (SMOTE) [21, 22]. We used different methods to balance the data and got the best result from the SMOTE method.

SMOTE is an algorithm that performs data augmentation by creating artificial data points based on original data points. SMOTE selects a random sample from minority class and determine k nearest neighbors for this sample. Then a vector between the current sample and a chosen neighbor is determined. The synthetic instances are generated by multiplying this vector with a random number between 0 and 1. The advantage of SMOTE is that duplicates are not generated and the data points generated are slightly different from the original data points. Therefore, in this study, to balance the "death" class of patients with COVID-19, we applied the SMOTE method. Before balancing the data, the death class contained only 176 records (13%), while after balancing the data, the death class contained 748 records.

**Normalization:** Data normalization is one of the main phases of data mining. When data have different scales, they have an adverse effect on each other and the algorithm at different change intervals. So the data should be in an equal range with each other. Each of the data recorded in the database will change between 0 and 1 [23]. This allows the data to be shorter in the domain and the model to be better trained. There are several techniques for normalization. In this study, Min-Max Normalization technique is used to

normalize the data as follows [24].

$$x = \frac{x - X_{min}}{X_{max} - X_{min}} \quad (1)$$

In this formula,  $X_{min}$  and  $X_{max}$  are equal to the minimum and maximum values of the data in the database, respectively.

**Relief Feature Selection:** The relief method is a filter-based feature selection algorithm that uses a statistical solution to select features [14]. In this method, at each step, a sample is randomly selected from the samples in the data set. Then, for each of the features of this sample, it finds the nearest Hit and the nearest Miss according to the Euclidean criterion. The nearest Hit is the sample that has the smallest Euclidean distance among other samples with the same class as the selected sample. The nearest Miss is the sample with the smallest Euclidean distance among samples from the opposite class to the selected sample.

$$W_i = W_i - (x_i - \text{nearHit}_i)^2 + (x_i - \text{nearMiss}_i)^2 \quad (2)$$

As shown in Equation 2, if the difference between a feature in the selected sample and the same feature in the sample of the same class is greater than the difference between the same feature in the selected sample and the same feature in the sample of the same class, weight (degree of importance) of this feature is reduced and vice versa.

By weighting the features, those that have a greater impact on the classification accuracy are identified. In order to select the most suitable features, we rank them according to their weight value. In this study, according to the various experiments, we selected 20 of the best features that had a higher rating and applied them to different data mining algorithms to predict the mortality of Covid-19 patients and evaluated the performance criteria of the algorithms.

#### 4. Data mining models

In this subsection, some of the AI algorithms used to develop the CDSS system in this study are introduced. Each of the data mining algorithms introduced in this section is implemented in the original and balanced dataset and their evaluation results are compared. All methods are implemented in the KNIME Analytics Platform.

##### 4.1. Decision tree

In data mining, the decision tree is a predictive model that is used for both regression and classification models [25]. In the decision tree structure, the prediction obtained from the tree is explained as a sequence of rules. The decision tree algorithm classifies the samples so that the classes are actually at the end of the leaf nodes. Each path from the root to a decision tree leaf expresses a rule, and finally, the leaf is labeled with the class in which the most records belong. The decision tree is used in problems that can be posed in such a way that they provide a single answer in the form of a group or class name [26]. In this study, two types of decision trees C4.5 [27] and Random forest [28] are used for the implementation of a decision tree on data from Covid-19 patients. Figure 2 shows the snapshot from the KNIME workflow of C4.5.

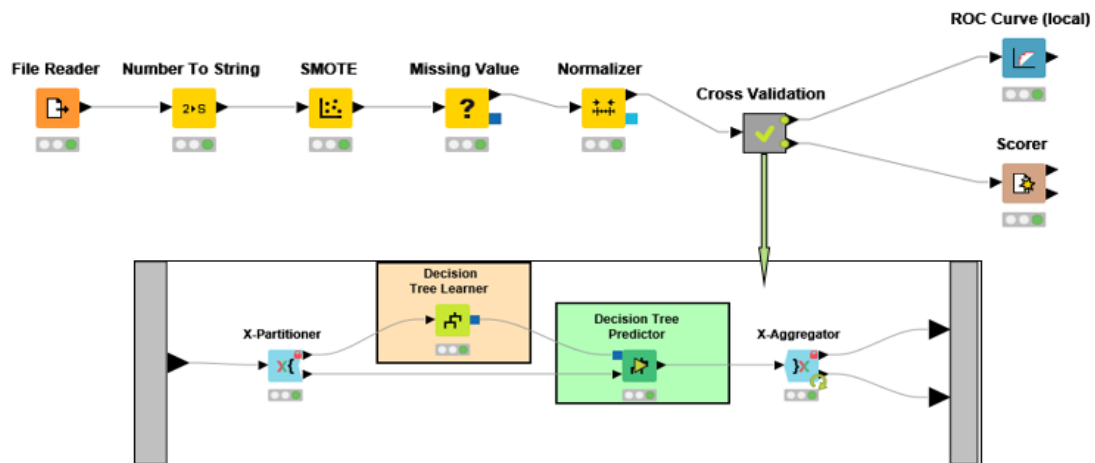


Figure 2. KNIME workflow of C4.5

#### 4.2. Support Vector Machine (SVM)

SVM is mostly used in classification problems. The basis of the SVM classifier is the linear classification of data. In the SVM algorithm, each data sample is drawn as a point in the n-dimensional space on the scatter diagram of the data (n is the number of features of a data sample) and the value of each feature related to the data determines one of the components of the coordinates of the point on the diagram. Then, by drawing a straight line, it categorizes different and distinct data. In the linear division of data, the line is selected that has a more reliable margin [29].

#### 4.3. Logistic regression

Logistic regression is a statistical regression model for two-way dependent variables. Being two-way means that a random event occurs in two possible situations like death or life, which are variables with two positions. Logistic regression can be seen as a special case of the general linear model and linear regression. Logistic regression model is based on completely different hypotheses from linear regression about the relationship between variables. These variables can be dependent or independent. We use logistic regression when we want to measure the relationship between an independent variable with continuous values and a dependent variable with qualitative values [30].

#### 4.4. K-Nearest Neighbor (KNN)

This algorithm classifies a test sample based on k neighboring neighbors. Train samples are presented as vectors in multidimensional feature space. The space is partitioned into areas with train samples. A point in space belongs to the class in which most of the training points belong to that class within the closest instance to k [31]. This study uses the Euclidean distance to find the nearest neighbors. The test sample is presented as a vector in the feature space and the Euclidean distance of the test vector with the total training vectors is calculated and the closest training sample to k is selected.

#### 4.5. Multi-Layer Perceptron (MLP)

MLP is a feed-forward neural network that consists of three main layers: input layer, hidden layer, and output layer. Each layer contains a group of nerve cells that are connected in a

directional graph to all the neurons in other layers. The MLP network establishes a non-linear connection between the input and output vectors using an activator function. In the training phase, training information is given to the perceptron, then the network weights are adjusted to minimize errors between the output and the target [32].

#### 4.6. Fuzzy rules

This algorithm receives numerical data as input and generates fuzzy rules based on the fuzzy intervals generated in the higher dimensional space [33]. Fuzzy intervals are defined by trapezoidal fuzzy membership functions for each dimension. To generate fuzzy rules, the input numeric columns are used as the first section of the rules and the last column, which is the target data in the classification, is introduced as the output of the rules. This column contains class information and can contain degrees between 0 and 1 [34]. The model output port contains the fuzzy rule model, which can be used for prediction in the Fuzzy Rule Predictor node. The number of fuzzy rules generated in this study is 209.

### 5. Evaluation and results

To evaluate different ML algorithms for predicting the mortality of patients, several performance metrics such as the ROC Curve as well as the accuracy, precision, sensitivity, specificity, and F-measure are used [35]. Table 2 shows the calculations of measures. Furthermore, the 10-fold cross-validation method is used to measure the efficiency of algorithms.

Table 2. Definition of performance metrics

Performance Metrics	Definitions
Precision	$TP / (TP + FP)$
Specificity / true negative rate (TNR)	$TN / (TN + FP)$
Sensitivity/ true positive rate (TPR) or Recall	$TP / (TP + FN)$
Accuracy	$(TP + TN) / (TP + TN + FP + FN)$
F-measure	$(2 \times \text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall})$

\* True Positive (TP), True Negative (TN), False Positive (FP),

False Negative (FN)

### 5.1. Predictive models for COVID-19

This subsection evaluates the performance of ML algorithms in predicting early mortality from COVID-19 disease. These algorithms include the decision tree, Random forest, SVM, MLP, KNN, and Fuzzy rules. The most important measure for determining the efficiency of a classification algorithm is classification accuracy. But in real problems, the classification accuracy measure is not a good measure for evaluating the efficiency of classification algorithms, because, concerning classification accuracy, the values of records of different categories are considered the same. Therefore, in problems dealing with unbalanced categories, other measures are used.

Table 3 presents the test results based on performance measures, accuracy, precision, sensitivity (recall), specificity, and F-measure without the use of data preprocessing techniques. The data used to evaluate performance is not normalized and also a large amount of data information have been removed from the dataset due to missing values. The important point is that the data is labeled unbalanced, and the number of data labeled "death" is much less than the number of data labeled "life".

According to the results presented in Table 3, the decision

tree (C4.5) algorithm performs better than other algorithms in terms of precision and F-measure criteria with values of 56.4% and 54.7%, respectively. The MLP algorithm has a higher recall rate than other algorithms with a value of 56.6%. Considering the specificity criterion, the logistic regression algorithm is superior to other algorithms. This algorithm achieved 93.6% specificity of the dataset shown in Table 3. In addition, the fuzzy rule base algorithm has the highest classification accuracy of 86.7% compared to other ML algorithms.

Figure 3 shows the performance results of the models. As can be seen, examining this chart cannot accurately show which algorithm is more efficient than the others. Since ML algorithms are compared based on five different criteria, it is not possible to choose the best algorithm with the highest performance. In this paper, we used Friedman's statistical test to compare the performance of ML algorithms based on different evaluation factors. This statistical test ranks the algorithms with a significance level of 0.05. Figure 4 shows the comparison results of Friedman test. The value of  $P$ -value  $< 0.05$  indicates that there is a significant difference in performance between the algorithms.

Table 3. Performance evaluation results without preprocessing

Model	Precision	Recall	Specificity	F-score	Accuracy
Decision tree (C4.5)	0.564	0.534	0.902	0.547	0.861
Random forest	0.430	0.386	0.926	0.407	0.823
SVM	0.287	0.412	0.375	0.336	0.457
Logistic regression	0.323	0.355	0.936	0.341	0.849
MLP	0.511	0.566	0.893	0.535	0.856
KNN	0.462	0.485	0.912	0.471	0.826
Fuzzy Rules	0.342	0.462	0.924	0.395	0.867

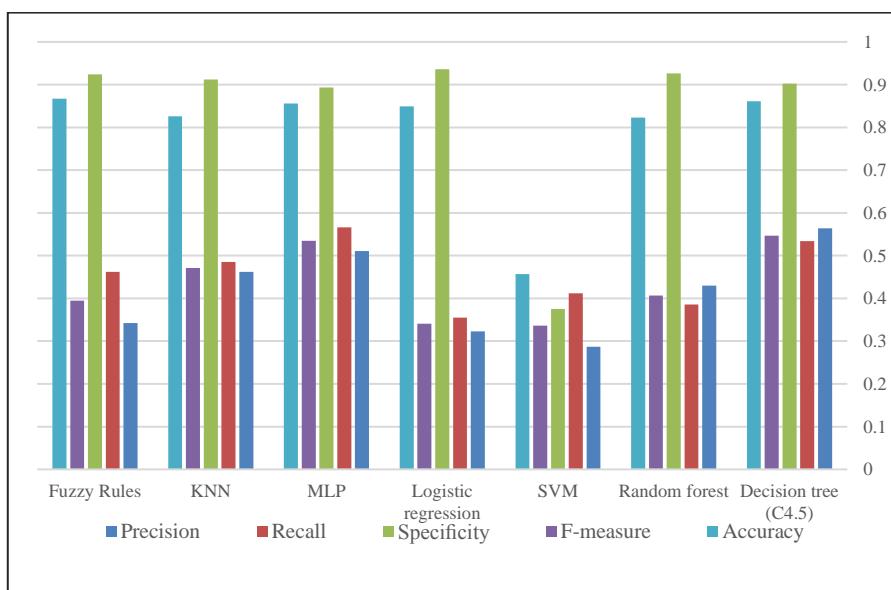


Figure 3. Comparison of performance measure of ML algorithms (without preprocessing)

According to Figure 4, the decision tree (C4.5) algorithm with a rank 7.6 generally performs better than other algorithms. According to the results shown in Table 3 and Figure 4, when preprocessing is not performed on data related to patients with Covid-19, mortality is predicted with low quality, and the results obtained cannot be effective and reliable in the decision and treatment processes.

Table 4 shows the results of comparing the performance of ML algorithms after the preprocessing stage of raw data obtained from patients with Covid-19 disease. The numerical results show that the performance of ML algorithms has improved significantly. All algorithms are applied to preprocess datasets and, by considering all measures, they have better results than before. Table 4 shows that the KNN algorithm works better than other ML algorithms with 94.2% and 92.2% in terms of precision and F-measure, respectively. Moreover, this algorithm has a higher classification accuracy of 97.1% than others. The Random forest algorithm is the best in terms of specificity criteria. The value of specificity for this algorithm is 98.6%. The recall criterion in the Fuzzy Rules base algorithm is 91.6%, which has the highest value compared to other algorithms.

Figure 5 shows a bar chart of comparing ML algorithms in terms of accuracy, precision, sensitivity (recall), specificity, and F-measure. By looking at this diagram, it is not possible to determine which algorithm generally

performs better than other algorithms. Figure 6 shows the mean rank of ML algorithms based on the Friedman test.

As shown in Figure 6, the KNN algorithm has the highest performance. This algorithm ranks first with a rank of 1.4. Then the Random forest algorithm has the best performance. The SVM algorithm with the rank of 6.6 is the weakest algorithm investigated in this study.

Figure 7 compares the performance metrics of the KNN algorithm before and after pre-processing. As we can see, the efficiency of KNN algorithm is significantly improved after data preprocessing. In this algorithm, the precision criterion has increased from 0.462 to 0.942. Moreover, the recall criterion has improved and has increased about 0.42. Appropriate pre-processing on the Covid-19 dataset has also had a good impact on the specificity and accuracy criteria and has improved the efficiency of the KNN algorithm by 0.07 and 0.15, respectively.

In addition to the performance evaluation criteria presented in Table I, the ROC curve is plotted for each of the ML algorithms used in this study. Figure 8 shows the ROC curves. In the ROC curve, the best classification performance will occur at the point with coordinates (0, 1), where we have the lowest error rate and the highest sensitivity rate. This point represents the perfect classification. As shown in Figure 8, the ROC curve is the best for the KNN algorithm because the curve is close to 1.

Friedman Aligned Ranks test (significance level of 0.05)		
Statistic	p-value	Result
18.33843	0.00544	H0 is rejected

Ranking	
Rank	Algorithm
7.60000	Decision tree (C4.5)
8.40000	MLP
17.20000	KNN
18.80000	Fuzzy Rules
20.00000	Random forest
22.20000	Logistic regression
31.80000	SVM

Figure 4. The mean rank of ML algorithms based on the Friedman test (without preprocessing)

Table 4. Performance evaluation results with preprocessing

Model	Precision	Recall	Specificity	F-score	Accuracy
Decision tree (C4.5)	0.941	0.791	0.961	0.863	0.858
Random forest	0.895	0.881	0.986	0.903	0.888
SVM	0.743	0.792	0.921	0.767	0.821
Logistic regression	0.768	0.763	0.954	0.776	0.938
MLP	0.905	0.839	0.947	0.858	0.896
KNN	0.942	0.903	0.982	0.922	0.971
Fuzzy Rules	0.790	0.916	0.978	0.848	0.965

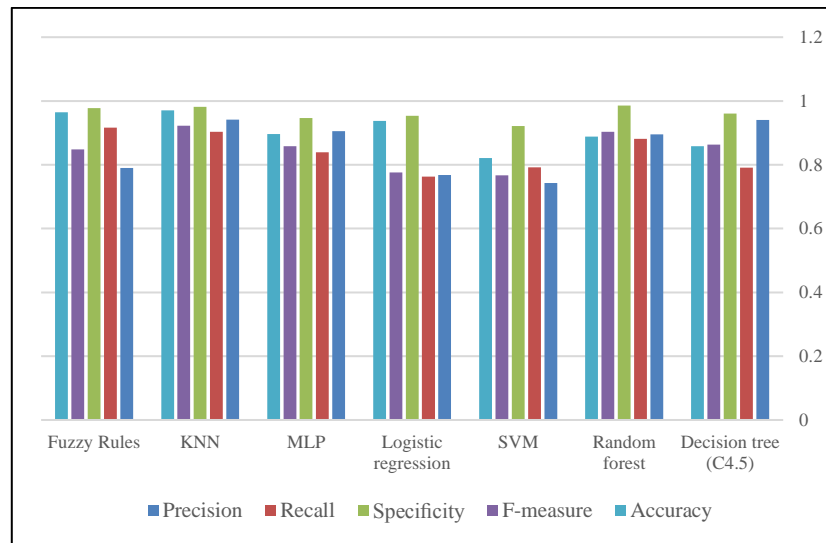


Figure 5. Comparison of performance measure of ML algorithms (with preprocessing)

Friedman test (significance level of 0.05)		
Statistic	p-value	Result
6.37037	0.00041	H0 is rejected

Ranking	
Rank	Algorithm
1.40000	KNN
3.00000	Random forest
3.20000	Fuzzy Rules
4.20000	MLP
4.20000	Decision tree (C4.5)
5.40000	Logistic regression
6.60000	SVM

Figure 6. The mean rank of ML algorithms based on the Friedman test (with preprocessing)

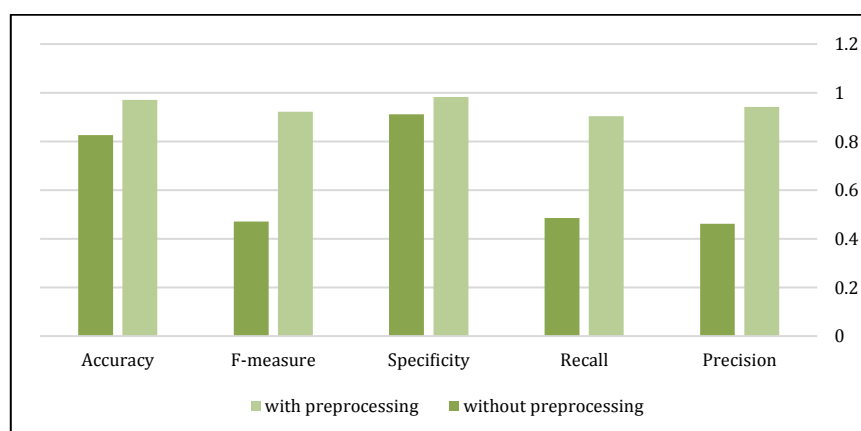


Figure 7. The performance metrics for K-NN before and after pre-processing

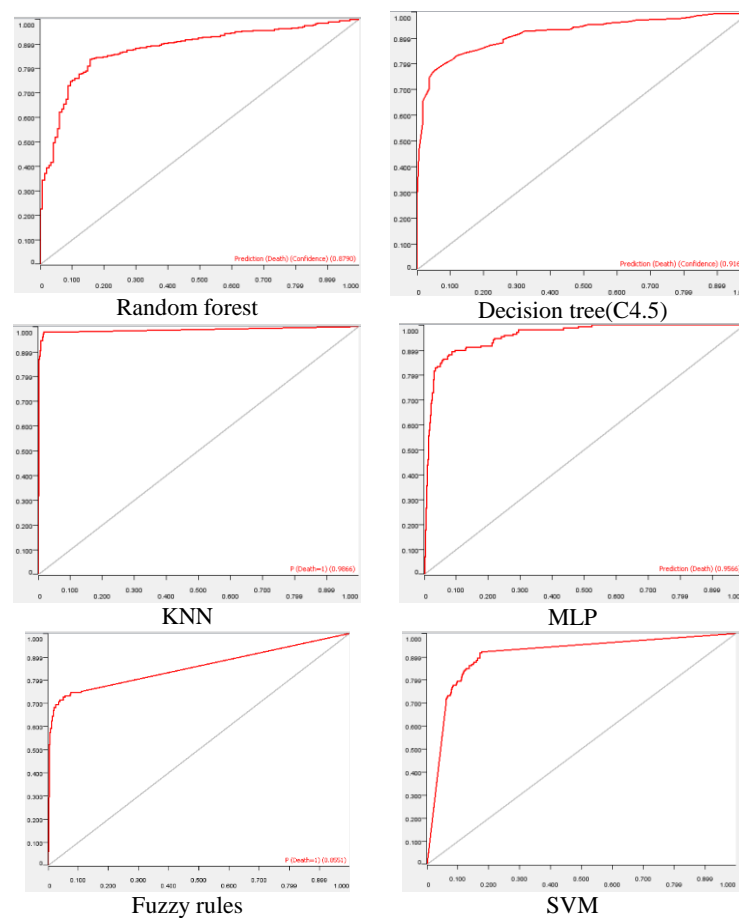


Figure 8. ROC curve for ML algorithms

## 6. Conclusion and future studies

COVID-19 is a viral disease that was declared an international public health emergency by the World Health Organization (WHO). The increase in mortality due to COVID-19 disease caused concerns among countries and economic, social, and educational problems. Early diagnosis of mortality from Covid-19 helps physicians to easily make clinical decisions as well as reduce diagnostic errors. In this study, different machine learning classification algorithms were tested on COVID-19 data to predict the death of infected patients and compared them based on different performance criteria. To increase the performance of these algorithms, the data were preprocessed before the experiment. The experimental results showed that the KNN algorithm is more efficient than other algorithms. In the future, we should use other feature selection methods to reduce data volume and increase the efficiency of classification algorithms.

## 7. References

- [1] Coronavirus Cases: <https://www.worldometers.info/coronavirus/>, accessed: 2020-04-10.
- [2] Liu, Q., Guan, X., Wu, P., Wang, X., Zhou, L., and Tong, Y., "Early transmission dynamics in Wuhan, China, of novel coronavirus-infected pneumonia", *New Journal of Medicine*, 2020.
- [3] Jiang, F., Deng, L., Zhang, L., Cai, Y., Cheung, C. W., and Xia, Z., "Review of the clinical characteristics of coronavirus disease 2019 (COVID-19)", *Journal of General Internal Medicine*, Vol. 6, pp. 1-5, 2020.
- [4] Wu, Z., and McGoogan, J. M., "Characteristics of and important lessons from the coronavirus disease 2019 (COVID-19) outbreak in China: summary of a report of 72314 cases from the Chinese Center for Disease Control and Prevention", 2020.
- [5] Turing, A. M., "Computing machinery and intelligence", In *Parsing the turing test: Springer*, pp. 23-65, 2009.
- [6] Briganti, G., and Le Moine, O., "Artificial Intelligence in Medicine: Today and Tomorrow", *Perspective*, Vol. 7, No. 27, pp. 1-27, 2020.
- [7] Omidian, Z., Hadianfard, A., "The study of clinical decision support systems role in health care (1980-2010)", *Jundishapur J Health Res*, Vol. 2, No. 3, pp. 1-13, 2011.
- [8] Paydar, K., Kalhori, S. R., Akbarian M, M., "A clinical decision support system for prediction of pregnancy outcome in pregnant women with systemic lupus erythematosus", *Int J Med Inform*, No. 97, pp. 239-246, 2017.
- [9] Sadoughi, F., Sheikhtaheri, A., "Applications of artificial intelligence in clinical decision making: opportunities and challenges", *Health Information Management*, Vol. 8, No.19, pp. 440-5, 2011.
- [10] Tuli, S., Tuli, S., Tuli, R., and Gill, S. S., "Predicting the growth and trend of COVID-19 pandemic using



- machine learning and cloud computing", *Internet of Things*, Vol. 11, pp. 100222, 2020.
- [11] Sun, L., Song, F., Shi, N., "Combination of four clinical indicators predicts the severe/critical symptom of patients infected COVID-19", *Journal of Clinical Virology*, Vol. 128, pp. 104431, 2020.
- [12] Aljameel, S. S., Khan, I. U., Aslam, N., Aljabri, M., and Alsulmi, E. S., "Machine Learning-Based Model to Predict the Disease Severity and Outcome in COVID-19 Patients", *Journal of Scientific Programming*, Vol. 21, 2021.
- [13] Muhammad, L., Islam, M. M., Usman, S. S., and Ayon, S. I., "Predictive data mining models for novel coronavirus (COVID-19) infected patients' recovery", *Journal of SN Computer Science*, Vol. 1, No. 4, pp. 1-7, 2020.
- [14] Kononenko, I., "Overcoming the myopia of inductive learning algorithms with RELIEFF", *Applied Intelligence*, Vol. 7, No. 1, pp. 39-55, 1997.
- [15] Silva, P., Luz, E., Silva, G., Moreira, G., Silva, R., Lucio, D., Menotti, D., "COVID-19 detection in CT images with deep learning: A voting-based scheme and cross-datasets analysis", *Informatics in Medicine Unlocked*, Vol. 20, pp. 100-127, 2020.
- [16] Nikooghadam, M., Ghazikhani, A., Saeedi, M., "COVID-19 Prediction Classifier Model Using Hybrid Algorithms in Data Mining", *International Journal of Pediatrics*, Vol. 9, No. 1, 2021.
- [17] Moulaei, K., Ghasemian, F., Bahaadinbeigy, K., Sarbi, R. E., Taghiabad, Z. M., and Engineering, "Predicting mortality of COVID-19 patients based on data mining techniques", *Journal of Biomedical Phys Eng*, Vol. 11, No. 5, pp. 653, 2021.
- [18] Paydar, K., Kalhori, S. R., Akbarian M, M., "A clinical decision support system for prediction of pregnancy outcome in pregnant women with systemic lupus erythematosus", *Int J Med Inform*, No. 97, 2017.
- [19] Han, J., Pei, J., Kamber, M., "Data mining: concepts and techniques", Elsevier; 2011.
- [20] Olson, D. L., "Data set balancing", *In Chinese Academy of Sciences Symposium on Data Mining and Knowledge Management*, pp. 71-80, 2004.
- [21] Bowyer, KW., Hall, LO., "SMOTE: synthetic minority over-sampling technique", *J Artif Intell Res*, Vol. 16, pp. 321-57, 2002.
- [22] Douzas, G., Bacao, F., Last, F., "Improving imbalanced learning through a heuristic oversampling method based on k-means and SMOTE", *Inf Sci*, Vol. 465, pp. 1-20, 2018.
- [23] Al Shalabi, L., and Shaaban, Z., "Normalization as a preprocessing engine for data mining and the approach of preference matrix", *In 2006 International conference on dependability of computer systems*, pp. 207-214, 2006.
- [24] Saranya, C., Manikandan, G., "A study on normalization techniques for privacy preserving data mining", *Journal of Engineering and Technology*, Vol. 5, No. 3, pp. 2701-2704, 2013.
- [25] Quinlan, J. R., "Simplifying decision trees", *International Journal of Man-Machine Studies*, No. 27, Vol. 3, pp. 221-234, 1987.
- [26] Brunello, A., Marzano, E., Montanari, A., "Sciavicco, G. J48ss: A novel decision tree approach for the handling of sequential and time-series data", *Computers*, Vol. 8, No. 1, 2019.
- [27] Wu, X., Kumar, V., Quinlan, JR., Ghosh J, J., Yang, Q., Motoda, H., "Top 10 algorithms in data mining", *Knowl Inf Syst*, Vol. 14, pp. 1-37, 2008.
- [28] Ozçift, A., "Random forests ensemble classifier trained with data resampling strategy to improve cardiac arrhythmia diagnosis", *Comput Biol Med*, Vol. 41, pp. 265-71, 2011.
- [29] Cortes, C., Vladimir, N., "Support-vector networks", *Machine Learning*, Vol. 20, No. 3, pp. 273-297, 1995.
- [30] David, W., Lemeshow, S., "Applied Logistic Regression (2nd ed.)", Wiley. 2000.
- [31] Yuan, J., Douzal-Chouakria, A., Yazdi, S. V., Wang, Z., "A large margin time series nearest neighbor classification under locally weighted time warps", *Knowl. Inform. Syst*, Vol. 59, No. 1, pp. 117-135, 2019.
- [32] Cho, YB., Farrokhkish, M., Norrlinger, B., Heaton, R., Jaffray D, D., Islam, M., "An artificial neural network to model response of a radiotherapy beam monitoring system", *Med Phys*. Vol. 47, 2020.
- [33] Yager, R. R., Zadeh, L. A., "An Introduction to Fuzzy Logic Applications in Intelligent Systems", Kluwer Academic, Dordrecht. 1992.
- [34] Bardossy, A., Duckstein, L., "Fuzzy Rule-Based Modeling with Application to Geophysical", *Biological and Engineering Systems*, CRC, Boca Raton. 1995.
- [35] Zhu, W., Zeng, N., Wang, N., "Sensitivity, specificity, accuracy, associated confidence interval and ROC analysis with practical SAS implementations", *NESUG proceedings: health care and life sciences, Baltimore, Maryland*, Vol. 19, pp. 67, 2010.

*Table of Contents:*

<b>Exploring Effective Features in ADHD Diagnosis among Children through EEG/Evoked Potentials using Machine Learning Techniques</b>	1
Faezeh Rohani - Kamrad Khoshhal Roudposhti Hamidreza Taheri-Ali Mashhadi	
<b>Andreas Mueller Embedding Knowledge Graph through Triple Base Neural Network and Positive Samples</b>	11
Sogol Haghani - Mohammad Reza Keyvanpour	
<b>Efficient and Deception Resilient Rumor Detection in Twitter</b>	21
Milad Radnejad - Zahra Zojaji Behrouz Tork Ladani	
<b>A Deep Neural Network Architecture for Intrusion Detection in Software-Defined Networks</b>	31
Somayeh Jafari Horestani - Somayeh Soltani Seyed Amin Hosseini Seno	
<b>Meta-Learning for Medium-Shot Sparse Learning via Deep Kernels</b>	45
Zohreh Adabi-Firuzjaee - Sayed Kamaledin Ghiasi-Shirazi	
<b>The Impact of Preprocessing Techniques for Covid19 - Mortality Prediction</b>	57
Soodeh Hosseini - Zahra Asghari Varzaneh	