Proceedings of the of the 17th Conference on Computer Science and Intelligence Systems pp. 79–88
DOI: 10.15439/2022F184
ISSN 2300-5963 ACSIS, Vol. 30

A Comparative Study of Short Text Classification with Spiking Neural Networks

Piotr S. Maciąg, Wojciech Sitek, Łukasz Skonieczny, Henryk Rybiński
Warsaw University of Technology
Institute of Computer Science
Nowowiejska 15/19, 00-665, Warsaw, Poland
piotr.maciag@pw.edu.pl, wojciech.sitek@pw.edu.pl, lukasz.skonieczny@pw.edu.pl, hrb@ii.pw.edu.pl

***Abstract*—Short text classification is an important task widely used in many applications. However, few works investigated applying Spiking Neural Networks (SNNs) for text classification. To the best of our knowledge, there were no attempts to apply SNNs as classifiers of short texts. In this paper, we offer a comparative study of short text classification using SNNs. To this end, we selected and evaluated three popular implementations of SNNs: evolving Spiking Neural Networks (eSNN), the NeuCube implementation of SNNs, as well as the SNNTorch implementation that is available as the Python language package. In order to test the selected classifiers, we selected and preprocessed three publicly available datasets: 20-newsgroup dataset as well as imbalanced and balanced PubMed datasets of medical publications. The preprocessed 20-newsgroup dataset consists of first 100 words of each text, while for the classification of PubMed datasets we use only a title of each publication. As a text representation of documents, we applied the TF-IDF encoding. In this work, we also offered a new encoding method for eSNN networks, that can effectively encode values of input features having non-uniform distributions. The designed method works especially effectively with the TF-IDF encoding. The results of our study suggest that SNN networks may provide the classification quality is some cases matching or outperforming other types of classifiers.**

***Index Terms*—short text classification, spiking neural networks, evolving spiking neural networks, NeuCube, SNNTorch, medical documents, PubMed documents**

I. Introduction

EFFECTIVE text classification is a difficult task that often requires adaptation of special types of learning and encoding methods. Classification of short texts is often even more difficult due to the very limited length of documents that can be used as training input data. The already offered methods for short text classification include, for example, Support Vector Machines (SVM), naive Bayes classifier, decision trees [1] or the classifiers that include clustering of input data as a preprocessing step [2].

Spiking Neural Networks (SNNs) are a type of neural networks that are highly inspired by biological mechanisms of learning and cognition of a human brain. Surprisingly, there are no publications applying SNNs to short text classification. Thus, in this work we evaluate three selected implementations of SNNs applied by us to the short text classification task, namely: our prepared classifier that uses *evolving Spiking Neural Networks, eSNNs*, the *NeuCube* implementation of SNNs as well as the *SNNTorch* implementation.

Evolving Spiking Neural Network (eSNN) is a recently introduced classifier that was successfully applied in various domains: transportation prediction [3], air pollution prediction [4]–[6], recognition of moving objects [7], or anomaly detection [8], [9]. To the characteristic of eSNNs belongs: ability to process large amounts of data efficiently and insignificant memory requirements. As it was proven in the enumerated examples of eSNNs usage cases, they can effectively use the biologically inspired learning and prediction mechanisms in typical engineering applications.

The other implementation selected for this study is the *NeuCube* implementation of SNNs [10], [11]. Contrary to the eSNNs implementation, NeuCube consists of three layers of spiking neurons: input, whose aim is to encode input values into firing times, internal, which consists of a reservoir (cube) of hidden neurons, whose weights are trained in an unsupervised manner using synaptic-plasticity rules, and output, which contains neurons responsible for assigning decision classes to testing examples.

Finally, as the third implementation of SNNs we selected recently developed *SNNTorch* implementation available as a package of the Pyhton language [12]. To the advantages of SNNTorch belong its flexibility to construct SNNs that can consists of many layers of neurons which combine not only neuronal models that are typically present in SNNs, such as the Leaky-Integrate-and-Fire (LIF) model, but also sigmoid neuronal models. In addition, SNNTorch can take advantage of a GPU-based processing in order to speed up training and classification procedures.

This paper provides the following contribution:

- To the best of our knowledge, for the first time in the literature we apply SNNs for classification of short texts and, especially, large sets of medical publications based on their metadata (such as a title or an abstract of a publication). To this end, we selected three types of SNNs: *eSNN* networks, the *NeuCube* implementation of SNNs as well as the *SNNTorch* implementation.
- As a part of our implementation of *eSNN* networks we propose a new input data encoding method. The proposed method first creates a histogram of input values of each feature F in the training dataset. The number of bins (subranges) of histogram is specified by a user-given parameter called B. Subsequently, the NI_{size} input

neurons of an eSNN are redistributed to encode the values of each bin of the histogram according to the cardinality of values in bins. As we present in the experiments, the offered encoding method provides much better classification accuracy than the other two encoding methods offered in the literature: a method that directly calculates the firing order of input neurons proposed in [5] and Gaussian Receptive Fields (GRFs) [13].
- We conduct experiments using the frequently-used *20-newsgroup* dataset[1] as well as two real PubMed datasets of medical publications selected from the website of the BioASQ competition[2]. Since we focus on classifying short texts, from each document of the 20-newsgroup we selected only 100 first words. In the case of two selected PubMed datasets, only a title of a publication is used as input data for each tested classifier.
- The obtained results of experiments suggest that SNNTorch implementation is more effective in short text classification than the other selected SNNs implementations. Additionally, for the selected PubMed datasets, SNNTorch gives results of classification slightly superior to the other classifiers tested in the experiments.

The paper is structured as follows. Section II presents the related work. Section III describes the SNNs implementations selected for this study. This section also describes the proposed encoding method for the eSNN networks. In Section IV, we give the description of the obtained datasets and applied preprocessing. Section V provides the results of experiments. Finally, in Section VI we conclude the work and discuss the results.

II. Related Work

A. Short Text Classification

Effective classification of short texts is a topic intensively studied nowadays. The already offered methods offered for text classification include: various types of classifiers, such as Support Vector Machines (SVM), naive Bayes classifiers, decision trees or different types of neural networks [1]. [14] distinguishes two types of approaches that can be applied for text classification: the first one, in which the set of text is first represented using Document-Term Matrix (DTM), which can be obtained using feature extraction method, such as Bag of Words (BoW) or Term Frequency - Inverse Document Frequency (TF-IDF). Subsequently, DTM is used to train a selected classifier, such as SVM. The second approach skips the process of generating DTM matrix and directly provides the set of texts as training data for a deep neural network.

Majority of recent approaches to short text classification with neural network models focused on applying Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs). [15] presented a classification model that is based on CNNs and incremental learning. In order to increase the classification accuracy, [15] applied an approach in which a current short document of a pipeline of documents is classified not only based on its textual content but also based on obtained classification results of preceding documents.

[16] presented the results of experiments on the classification of one-sentence questions using CNNs. The dataset applied in [16] was obtained from the WikiAnswer and contained $608\,650$ questions grouped into several hundred categories. As the results of experiments of [16] suggest, CNN networks can classify questions with accuracy comparable or better than SVM classifier.

In [17], a model that combines CNNs networks with SVM classifier was applied to the classification of simple sentences expressing either positive and negative feelings. In the approach presented in [17], first, a word embedding method (such as Word2Wec) is used to obtain the vector representation of each word in the text corpus. Subsequently, each text in the corpus is represented as a sequence of vectors, each corresponding to one of the text's words. Such a representation of texts is next used as input data for CNN network that consists of convolutional, max-pooling and fully-connected layers. The results obtained from the fully-connected layer are used as training data for the SVM classifier. The results of experiments presented in [17] suggest that this approach can provide better classification results than separate CNNs and SVM classifiers.

The review of the other types of classifiers applied for (short) text classification (and in particular MESH dataset) can be found, for example, in [14], [18], [19].

B. Spiking Neural Networks for Text Classification

We are aware of only two other works adapting spiking neural networks for text classification. However, contrary to this work, both of them applied SNNs for long text classification. In order to classify longer texts, [20] offered a method consisting of two phases, The first phase consists of transforming a text into a vector of numbers using TF-IDF encoding and, subsequently, into a sequence of spikes. In the second phase, an SNN network is taught in an unsupervised way using the spikes generated in the first phase in order to generate spike-based low-dimensional representation of a text. Subsequently, the generated representation is used as input data for the training of logistic regression, which is responsible for the final classification of documents. Thus, in the approach of [20], an SNN network can be perceived as a dimensionality reduction technique of a TF-IDF text representation. While the approach of [20] was shown to provide superior text classification results to the other classifiers tested there, it used only logistic regression, which (as presented, for example, in our experiments) itself can be an effective text classifier.

[21] presents a comparison of classification results for different types of word embeddings (such as Word2Vec and GloVe [22]) and neuron types that were applied in SNN (such as the LIF neuronal model). The results obtained in [20] suggest that SNNs can be effectively applied to classify longer texts.

[1]http://qwone.com/ jason/20Newsgroups
[2]http://participants-area.bioasq.org/datasets

III. The Selected Implementations of Spiking Neural Networks

A. The Evolving Spiking Neural Networks Implementation

In Fig. 1, we present the architecture of our implementation of eSNNs. The designed eSNN network consists of groups of input neurons $\mathbf{NI}^{(F_1)}, \ldots, \mathbf{NI}^{(F_m)}$ encoding values of m features $\mathcal{F} = \{F_1, \ldots, F_m\}$[3]. The number of input neurons in each group $\mathbf{NI}^{(F)}$ is the same and is specified by the user-given parameter NI_{size}. The output neurons in the repository $\mathbf{NO}$ are assigned decision classes present in the training dataset of texts $\mathbf{D}_{tr}$ (we assume that each text in $T \in \mathbf{D}_{tr}$ has one and only one decision class). Thus, given L decision classes, the output neurons $\mathbf{NO}$ are organized into L groups. In the learning process of an eSNN network, a new candidate output neuron n_c is created for each training text $T \in \mathbf{D}_{tr}$ and either added to the repository $\mathbf{NO}$ or merged with the neurons already existing there.

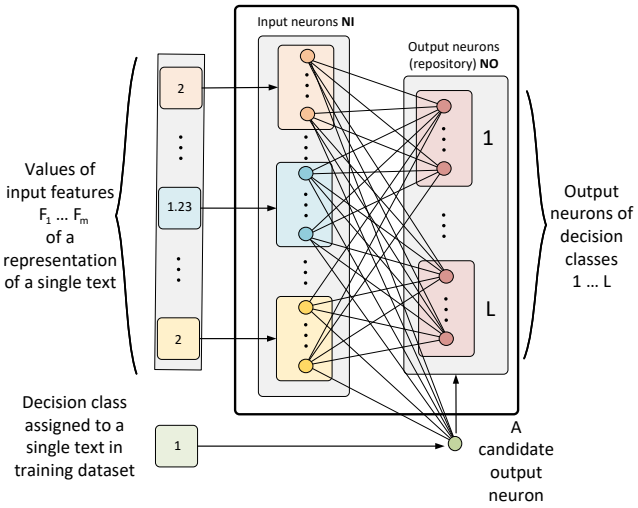

Fig. 1. The architecture of an eSNN network adopted in this study.

1) Input Layer: For the encoding of input values of features $\mathcal{F}$ into spikes we develop a new encoding method. Our motivation to develop the presented method lies in the fact that the previously introduced methods dedicated for eSNNs[4] (such as the GRFs method [24] or the method of [5]) do not work well with some text representations (such as the TF-IDF representation)[5]. Specifically, both the GRF method and the method of [5] are not able to effectively encode input values of features having non-uniform distributions. This can be explained by the fact that both of these methods divide the range of input values of a feature F into a number of equal subranges, the number of which is equal to the user-given number of input neurons NI_{size}. A center value of each of such subranges is associated with one input neuron. Given an input value to be encoded and the obtained center values of input neurons, the GRFs method and the method of [5] calculate Euclidean distances between the input value and the center values of input neurons. The non-decreasing order of such Euclidean distances between the input value and the center values of input neurons identifies the firing order of input neurons. In the case of non-uniform distributions of input values (such as a normal distribution or skewed distributions), it can often happen that a relatively high number of input values will be encoded using the same firing order of input neurons, while the values of other ranges will be associated with more distinguishing firing order of input neurons. To alleviate this problem, we offer the method presented below.

The proposed encoding method requires two user-given input parameters, which are the number of input neurons NI_{size} encoding values of each feature $F \in \mathcal{F}$ as well as the number of bins (subranges) B (where $B < NI_{size}$) used to create a histogram of values of each feature F. The proposed method first creates a histogram of input values of a feature F using solely the training dataset. Subsequently, the NI_{size} input neurons are allocated to encode the values of each bin of the histogram in the following way. First, each bin is assigned at least one input neuron of NI_{size} neurons. The rest of $NI_{size} - B$ input neurons is allocated as follows.

Let $Min^{(F)}$ and $Max^{(F)}$ be minimal and maximal values of feature F in training dataset $\mathbf{D}_{tr}$, respectively. The width of range of each bin equals $Bins_{width} = \frac{Max^{(F)} - Min^{(F)}}{B}$. The range of each bin is calculated as follows:

- $\big[Min^{(F)} + (i-1) \cdot Bins_{width}, Min^{(F)} + i \cdot Bins_{width}\big)$, for $Bin_i, i = \{1, \ldots, B-1\}$.
- $\big[Min^{(F)} + (B-1) \cdot Bins_{width}\big]$, for Bin_B.

Subsequently, the number of values of a feature F in each bin is calculated and remembered as $Bin_i.\overline{\overline{Values}}$. The number of neurons allocated to each bin is obtained using Proposition 1.

Proposition 1. *Let $Bin_i.\overline{\overline{Neurons}}$ represents a number of input neurons allocated to encode values of Bin_i and $\overline{\overline{\mathbf{D}_{tr}}}$ be a number of training examples (texts):*

- $Bin_i.\overline{\overline{Neurons}} = \left\lfloor \frac{Bin_i.\overline{\overline{Values}}}{\overline{\overline{\mathbf{D}_{tr}}}} \cdot (NI_{size} - B) \right\rfloor + 1$, *for $i \in \{1, \ldots, B-1\}$,*
- $Bin_i.\overline{\overline{Neurons}} = \left\lceil \frac{Bin_i.\overline{\overline{Values}}}{\overline{\overline{\mathbf{D}_{tr}}}} \cdot (NI_{size} - B) \right\rceil + 1$, *for $i = B$.*

Given the number of input neurons allocated to each bin, we can obtain the center value μ_j of each input neuron $n_j \in \mathbf{NI}$. The center values μ_j of input neurons are directly used to calculate firing order of input neurons. The center values are calculated according to Proposition 2. For each Bin_i, let

$$Bin_i.\Delta = Bins_{width} / Bin_i.\overline{\overline{Neurons}}$$

[3]The number of features $\mathcal{F}$ depends on the used text representation: for example, Word2Vec and Doc2Vec representations allow the user to specify the number of features [23].

[4]Unlike the other implementations of SNN networks, eSNN does not require exact firing times of input neurons to be propagated into the network. Rather than, it is enough to calculate the firing order of input neurons $\mathbf{NI}^{(F)}$ encoding a value of each feature $F \in \mathcal{F}$.

[5]In fact, in Section V we present experiments comparing the mentioned encoding methods for eSNNs.

denote the width between center values of input neurons allocated to Bin_i.

Proposition 2. *For each bin $i \in \{1, \ldots, B\}$, the center value of each input neuron $n_j \in Bin_i.Neurons, j \in \{1, \ldots, Bin_i.\overline{\overline{Neurons}}\}$ is calculated as follows*

$$\mu_j = Min^{(F)} + (i-1) \cdot Bins_{width} + (j - 0.5) \cdot Bin_i.\Delta$$

Finally, given the center values of input neurons (please note, that it is enough to calculate the center values μ_j one time after $\mathbf{D}_{tr}$ is load by an eSNN), we can calculate firing order $order_{n_j}$ of spikes that are propagated into the network. Let assume that $x^{(F)}$ is the value of a feature F to be encoded by the proposed method. Proposition 3, shows how to obtain the center value closest to an input value $x^{(F)}$.

Proposition 3. *Let μ_k be the center value closest to input value $x^{(F)}$. Index k is calculated as follows:*

$$k = \begin{cases} j \Big| \ |x^{(F)} - \mu_j| \text{ is smallest}, & \text{if } x^{(F)} \in [Min^{(F)}, Max^{(F)}] \\ 1, & \text{if } x^{(F)} < Min^{(F)}, \\ NI_{size}, & \text{if } x^{(F)} > Max^{(F)}. \end{cases}$$

Given the k index, the firing order of all input neurons $n_j \in \mathbf{NI}$ is obtained as given in Algorithm 1. We illustrated the example coding using the proposed method in Fig. 2.

Algorithm 1 Calculate firing order of input neurons

Input: $x^{(F)}$ - input value to be encoded, NI_{size} - number of input neurons.
Ensure precalculated: Bin_i - the structure containing parameters of i-th bin of a feature F (i.e. $Bin_i.Neurons, Bin_i.\Delta, Bin_i.\overline{\overline{Values}}$), μ_j - center values of input neurons $n_j \in \{1, \ldots, NI_{size}\}$, k - index of a center value μ closest to $x^{(F)}$.
Output: Firing order of input neurons $n_j \in \{1, \ldots, NI_{size}\}$.

$order_{n_k} \leftarrow 0, ord \leftarrow 0$
$l \leftarrow k-1; r \leftarrow k+1$.
while $l \geq 1$ **OR** $r \leq NI_{size}$ **do**
  if $l \geq 1$ **then** $dist_l \leftarrow |\mu_l - x^{(F)}|$ **end if**
  if $r \leq NI_{size}$ **then** $dist_r \leftarrow |\mu_r - x^{(F)}|$ **end if**
  if $l < 1$ **AND** $r \leq NI_{size}$ **then**
    $order_{n_r} \leftarrow ord, ord \leftarrow ord + 1, r \leftarrow r + 1$
  else if $l \geq 1$ **AND** $r > NI_{size}$ **then**
    $order_{n_l} \leftarrow ord, ord \leftarrow ord + 1, l \leftarrow l - 1$
  else if $l \geq 1$ **AND** $r \leq NI_{size}$ **then**
    if $dist_l < dist_r$ **then**
      $order_{n_l} \leftarrow ord, ord \leftarrow ord + 1, l \leftarrow l - 1$
    else
      $order_{n_r} \leftarrow ord, ord \leftarrow ord + 1, r \leftarrow r + 1$
    end if
  end if
end while

Algorithm 1 calculates firing order of input neurons as follows. As a first fires the input neuron n_k, whose center value μ_k is closest to the input value $x^{(F)}$ (the firing order function $order_{n_k}$ of the input neuron n_k is set to 0). Subsequently, Algorithm 1 calculates firing order of the rest input neurons, whose center values are located to the left and to the right of the center value of the first firing input neuron n_k. The firing order of these neurons is calculated in a single scan using the distances between their center values and the input value $x^{(F)}$. To this end, the algorithm uses three counters: l and r which point to the neurons whose center values are located to the left and to the right of the center value μ_k, respectively, as well as ord counter which stores the current firing counter (initially set to 0). Initially, l and r are set to $k-1$ and $k+1$, respectively. In each iteration of the main while loop, the algorithm calculates firing order of one input neuron pointed either by l or r counters. If the distance between center value of an input neuron n_l and $x^{(F)}$ is smaller than the distance between center value of an input neuron n_r and $x^{(F)}$, then $order_{n_l}$ is set to the current value of the ord counter, ord is incremented and l is decremented. Otherwise, $order_{n_r}$ is set to ord, ord is incremented and r is incremented. The computational complexity of Algorithm 1 is linear in the number of input neurons NI_{size}, similarly to the encoding algorithms presented in [5] or [13].

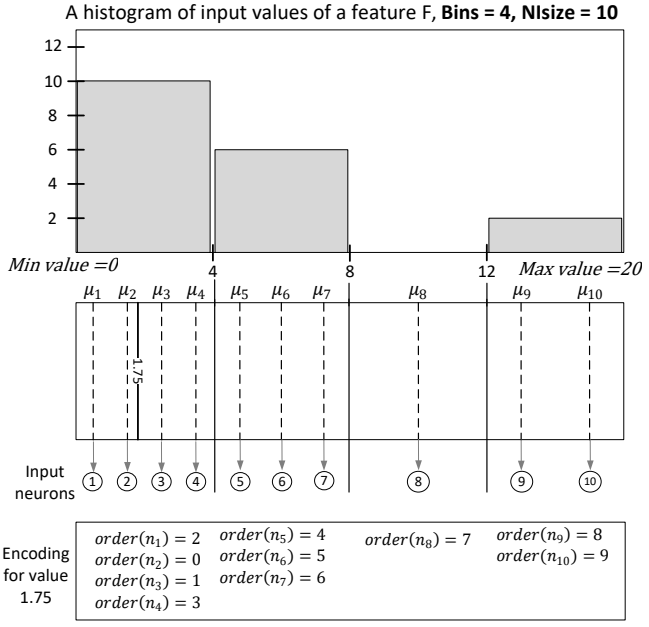

Fig. 2. The proposed encoding method - the NI_{size} input neurons are redistributed to code the values of each bin of a histogram; the encoded value is 1.75.

2) Network's Learning and Classification: The firing orders of input neurons are calculated separately for each text in either training or testing dataset. The firing orders obtained for texts of the training dataset are used in the network's learning phase, while the firing orders calculated for texts of the testing dataset are used to classify these texts.

In the eSNN learning process, for each training text T in dataset $\mathbf{D}_{tr}$, there is created a candidate output neuron n_c

which is assigned a single decision class of text T. We denote a decision class assigned to n_c as: $Class(n_c)$.

The candidate output neuron is connected through synapses to all input neurons $\mathbf{NI}$. The vector of weights of such synapses is denoted as $\mathbf{w}_{n_c} = [w^{(A_1)}_{n_1,n_c}, \ldots, w^{(F_1)}_{n_{|\mathbf{NI}^{(F_1)}|},n_c}, \ldots, w^{(F_m)}_{n_1,n_c}, \ldots, w^{(F_m)}_{n_{|\mathbf{NI}^{(F_m)}|},n_c}]$, where m is the number of features $\mathcal{F}$. To initialize the weights of synapses we apply the rank-order rule [24]. Each weight of a vector $\mathbf{w}_{n_c}$ is initialized according to Eq. (1).

$$w^{(A)}_{n_j n_c} = mod^{order_{n_j}}, n_j \in \mathbf{NI}, \tag{1}$$

where mod is a modulation factor whose value is specified by the user and should be in the range $(0, 1)$.

Each output neuron (either candidate n_c or the output neuron n_i already present in $\mathbf{NO}$) has also an update counter M. The value of such update counter is first set to 1 when a candidate is created, and subsequently is incremented when an output neuron present in $\mathbf{NO}$ is updated using a candidate output neuron.

After the candidate neuron n_c is created and its synapses' weights are initialized, it is either added to the repository of output neurons $\mathbf{NO}$ or merged with one of the output neurons already existing in $\mathbf{NO}(Class(n_c))$, that is in the group of output neurons of class $Class(n_c)$ in $\mathbf{NO}$. To this end, Euclidean distances $Dist_{n_c,n_i}$ between the vector of synapses' weights $\mathbf{w}_{n_c}$ and the vectors of synapses weights $\mathbf{w}_{n_i}$ of each output neuron $n_i \in \mathbf{NO}(class_{n_c})$ are calculated. If there exists such an output neuron n_s for which $Dist_{n_c,n_s}$ is minimal and below the value $simTr \cdot \overline{Dist}$, then the vector $\mathbf{w}_{n_s}$ and counter M_{n_s} are updated according to Eq. (2) and n_c is discarded. Otherwise, n_c is simply inserted into $\mathbf{NO}(class_{n_c})$. $simTr$ is a user-specified similarity threshold, whose value is in the range $[0, 1]$[6].

$$\mathbf{w}_{n_s} = \frac{\mathbf{w}_{n_s} \cdot M_{n_s} + \mathbf{w}_{n_c}}{M_{n_s} + 1}, M_{n_s} = M_{n_s} + 1. \tag{2}$$

$\overline{Dist}$ is the tight upper bound on the Euclidean distances between any possible candidate output neuron and any output neuron in $\mathbf{NO}$ and, as presented in [5], can be calculated using Eq. (3).

$$\overline{Dist} = \left[\sum_{F \in \mathcal{F}} \sum_{j=1}^{NI_{size}} \left(mod^{j-1} - mod^{NI_{size}-j-1} \right)^2 \right]^{\frac{1}{2}} \tag{3}$$

After eSNN is taught using the training dataset $\mathbf{D}_{tr}$, each testing text $T \in \mathbf{D}_{ts}$ is assigned one decision classes of all decision classes of output neurons in $\mathbf{NO}$. To this end, the value of Post-Synaptic Potential PSP_{n_i} of a membrane of each output neuron n_i in $\mathbf{NO}$ is calculated according to Eq. (4).

$$PSP_{n_i} = \sum_{F \in \mathcal{F}} \sum_{n_j \in \mathbf{NI}^{(F)}} w^{(F)}_{n_j n_i} \cdot mod^{order_{n_j}}. \tag{4}$$

[6]Please note, that the greater values of $simTr$ increase the chance that a candidate output neuron will be merged with one of the neurons already existing in the repository $\mathbf{NO}$

In Eq. (4), $w_{n_j n_i}$ is weight of a synapse connecting the input neuron n_j to the output neuron n_i that is calculated in the network's learning phase. $order_{n_j}$ is a firing order value of input neuron $n_j \in \mathbf{NI}^{(A)}$ given the encoding of a value of feature F in a testing text T. Finally, the testing text T is assigned a decision class of an output neuron $n_{max} \in \mathbf{NO}$, whose membrane PSP value $PSP_{n_{max}}$ is maximal. One can find the pseudocode of described learning and classification procedures of eSNNs, for example, in [5], [9], [24]. We posted our implementation that was used in the experiments along with the used dataset at the GitHub repository[7].

B. The NeuCube Implementation of Spiking Neural Networks

As the second implementation of SNNs that is selected by us for the experiments we used the NeuCube implementation [10][8]. Unlike the eSNNs implementation presented in subsection III-A, NeuCube implements an SNN network that consists of three layers of neurons: input, internal and output. The aim of the input layer of NeuCube is to convert values of features of a text representation into a sequence of spikes that is propagated into the network.

Since NeuCube implements four temporal coding algorithms, it requires each text (either from the training $\mathbf{D}_{tr}$ or testing $\mathbf{D}_{tr}$ datasets) to be represented as a time series. In our approach, each text is represented as a single time series TS containing all values of features $(F_1, \ldots, F_m)$. Thus, given a text representation having m features $\mathcal{F}$, the time series of each text consists of a series of m values. The temporal encoding algorithms implemented in NeuCube are: Threshold-based Representation (TR), Moving Window (MV), Step Forward (SF), and Bens Spiker Algorithm. The results presented in [25] suggests that the most effective is the TR algorithm, which was used by us in the experiments. Given time series representation TS of a text, the TR algorithm generates spikes by first calculating $ATB = \mu SR \cdot \sigma$ value (where μ and σ are mean and standard deviation of values of TS, respectively). Next, a positive spike is generated if the difference between two consecutive values of TS is positive and greater than ATB. If the difference is negative and smaller than ATB, then a negative spike is generated. The generated example of TR encoding of the time series values of the first document of the 20-newsgroup dataset is given in Fig. 3.

The internal layer of NeuCube consists of a cube of Leaky-Integrate-and-Fire (LIF) neurons [26]–[28] that are interconnected using both excitatory and inhibitory synapses. The number of such neurons in the cube and their topological locations can be defined by the user. The initial synapses and their weights are generated using the *small-world* principle (according to which the neurons located in a topological proximity have a grater chance to be connected). In the cube's learning process, the weights of synapses are calculated according to the Spike-Time Dependent Plasticity (STDP)

[7]https://github.com/piotrMaciag32/eSNN-short-text-classifier

[8]NeuCube is a an application with a graphical user interface implemented in Matlab and is free for download form https://kedri.aut.ac.nz/R-and-D-Systems/neucube

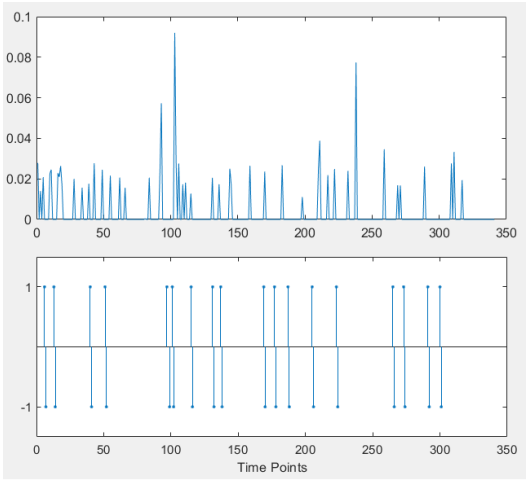

Fig. 3. The NeuCube TR encoding of the time series values of the first document of the 20-newsgroup dataset (the upper plot presents an input time series of TF-IDF values, the lower plot shows the obtained encoding).

rules. Let us consider two neurons: n_j and n_i presented in the internal layer of NeuCube and let us assume that there is a synapse from neuron n_j to neuron n_i. Given emission of spikes from neurons n_j and n_i at times t_j and t_i, respectively, the change of the weight value of the synapse from n_j to n_i is calculated according to Eq. (5).

$$w_{j,i}(t) = \begin{cases} w_{j,i}(t-1) + \eta/\Delta t, & \text{if } t_i > t_j, \\ w_{j,i}(t-1), & \text{if } t_i = t_j, \\ w_{j,i}(t-1) - \eta/\Delta t, & \text{otherwise}, \end{cases} \quad (5)$$

where η is the STDP rate learning parameter specified by the user.

Finally, the third layer of NeuCube consists of output neurons whose aim is to represent decision classes present in the training dataset $\mathbf{D}_{tr}$. Each output neuron in the output layer of NeuCube is connected to all neurons in the internal layer. The output neurons in NeuCube are grouped according to decision classes similarly to the output neurons **NO** of eSNNs. As in eSNNs, for each training text there is created one candidate output neuron that is always added to the set of output neurons of NeuCube (unlike in eSNNs, in which candidate output neurons can be merged with the output neurons already existing in the output layer).

The membrane Post-Synaptic Potentials (PSP) values of both internal and output neurons are calculated according to Eq. (4). Both internal and output neurons emit a spike when their PSP values exceed a certain firing threshold C, which is specified by the user. Specifically, a neuron n_i emits a spike according to Eq. 6.

$$\text{Emit a spike by } n_i \text{ at time } t = \begin{cases} True & \text{if } PSP_{n_i} \geq C, \\ False & \text{if } PSP_{n_i} < C, \end{cases} \quad (6)$$

In Fig. 4, we present the architecture of NeuCube along with the selected representation of each text. In Table I, we show the learning parameters of NeuCube along with their description.

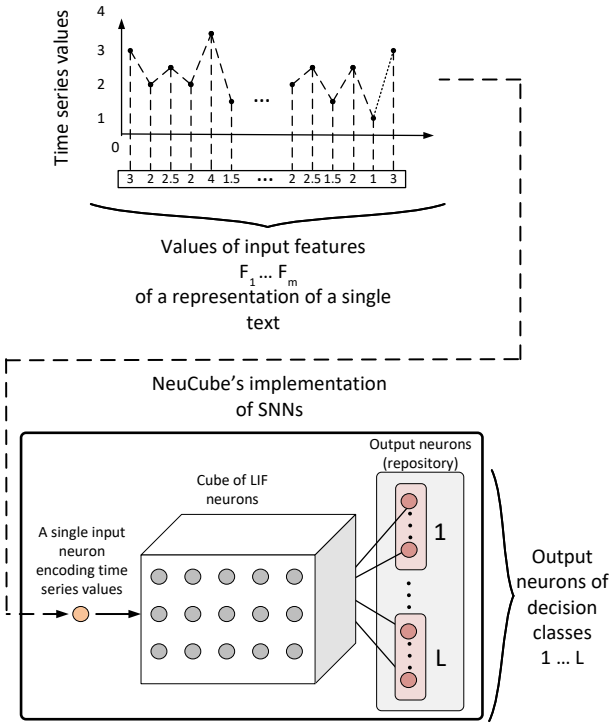

Fig. 4. The NeuCube's architecture and the applied representation of a text as a series of TF-IDF values.

C. The SNNTorch Implementation

The third implementation of SNNs selected for the experiments is the SNNTorch implementation, recently developed as the Python language package [12]. To the advantages of the SNNTroch implementation belongs the fact that it is built on the basis of the well known deep-learning Python framework *Pytorch*. SNNTorch allows us to combine SNNs with such types of neural networks as Multilayer Perceptron neural network or CNN network. Currently, SNNTorch offers eight types of spiking models of neurons: Alpha, Lapique, Leaky (LIF), RLeaky, RSynaptic, SConv2dLSTM, SLSTM and Synaptic. In our experiments, we applied the LIF neuronal model. Our applied architecture of neurons in SNNTorch consists of four layers as presented in Fig. 5. The number of input neurons equals the number of features in the input data, while the number of output neurons is the same as the number of decision classes present in the training parto of the dataset.

IV. Characteristic of Selected Datasets and Their Preprocessing

In the experiments, we used three publicly available datasets that are widely used as benchmarks in the evaluation of text classifiers. In the experiments, we applied the TF-IDF representation of all texts. As previous experiments with text

TABLE I
THE MAIN PARAMETERS OF SNN USED IN NEUCUBE.

Parameter	Description
SR	TR algorithm threshold.
η (STDP Rate)	STDP rate for weights modification in the internal layer.
Refractory time	A period in which an input is inactive to incoming spikes after emission of a spike by itself.
Mod	Modulation parameter as given in Eq. 5.
Training iters.	Number of its. of the unsupervised learning stage.
Firing treshold	Firing threshold for spike emission by neurons.

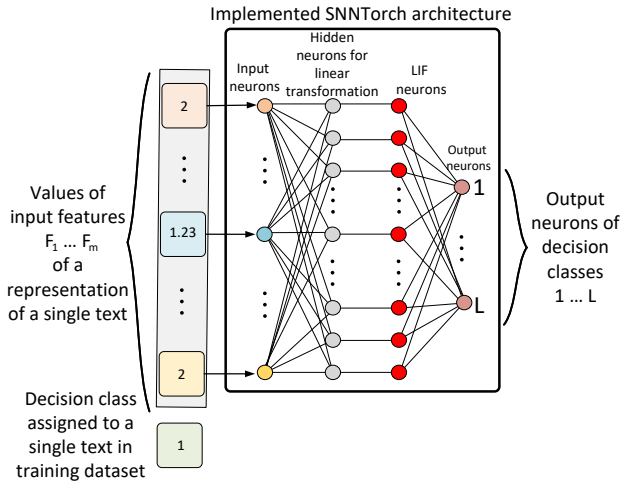

Fig. 5. The SNN architecture that was constructed using SNNTorch package for the purpose of experiments.

data and SNN networks suggest (see, for example, [20]), TF-IDF is a suitable representation for these type of neural networks. In order to obtain the TF-IDF representation, first, for each dataset we calculated the Document-Term Matrices (DTMs). DTM is a matrix that contains as many rows as the number of documents in either training or testing part of the dataset, and as many columns as the size of vocabulary (the set of all terms present in the entire dataset). Each cell of a DTM matrix can contain, for example, a number of occurrences of a given word of vocabulary (defined by a column of DTM) in a document (defined by a row of DTM)[9].

For all selected datasets, the DTM matrices are obtained as follows. First, the vocabulary is calculated. In order to obtain the vocabulary of each dataset, we applied the *text-mining* package of the R language [29] (for the 20-newsgroup dataset) and *Gensim* package of the Python language [30] (for the PubMed Mesh dataset). The vocabulary is obtained by applying the following steps to the set of texts of each dataset:

1) Removing all numbers from a text.
2) Removing punctuation.
3) Removing English stopwords and articles.
4) Transforming all upper-case letters to lower-case letters.
5) Applying texts' stemming.

After the execution of the above steps, we calculated the DTM matrices for the training and testing parts of datasets separately as follows.

A. The Preprocessed 20-newsgroup Dataset

The 20-newsgroup dataset contains 18 846 texts that are grouped into 20 news categories. To the distinguishing characteristic of the 20-newsgroup dataset belongs the fact that 20 categories often belong to very different domains. For example: politics, sociology, religion or computer devices. The texts are split into training and testing parts in the proportion 6:4 by the author of the dataset. Thus, the training part consists of 11 307 training texts and 7538 testing texts. Most of the texts in the datasets consists of several hundreds of words. Thus, we have shorten each text by selecting only its 100 first words as text used for classification. The vocabulary calculated using such preprocessed shorten texts contains 132 370 terms. The short texts are used to obtain the Document Term Matrices (DTMs) for training and testing parts separately. Since the obtained vocabulary contains 132 370 terms, each DTM matrix would also contain 132 370 columns - far too many for most of classifiers. Thus, we decided to remove sparse words from DTMs as follows:

- For the NeuCube implementation we remove from the vocabulary these terms which are not present in at least 95% of text (this reduces the number of columns of DTM to 341) - such a reduction was forced by the memory constraint of NeuCube, which prevents loading too large datasets.
- For all other tested classifiers we remove from the vocabulary these terms which are not present in at least 99% of text (this reduces the number of columns of DTM to 751).

B. The Preprocessed Imbalanced and Balanced PubMed Dataset

The PubMed dataset contains several millions of medical publications that are categorized according to the Medical Subject Headings (MeSH). MeSH is a set of classes organized into a hierarchical structure with 16 main branches (in overall, MeSH consists of nearly 30 000 categories). Each PubMed

[9]Obviously, such a representation usually leads to a significant size of a DTM, whose cells mostly contain 0 (which indicates the situation when a word is not present in a document).

document is usually indexed (either by the authors of a document/publication or by a publisher) using several MeSH categories. The obtained by us metadata of PubMed documents is posted as a part of the BioASQ competition [31].

For the purpose of our experiments, we randomly selected metadata of 10 000 PubMed documents, each of which is assigned one of the 16 main categories of the MeSH classification. Since the documents in these main categories are unevenly distributed, the resultant dataset is *imbalanced* (for example, majority of publications belong to the category *Chemicals and Drugs*, while there are few publications belonging to the category *Information Sciences*). Since we are focused on classification of short texts, we decided that input data provided to classifiers will contain only a title of a publication. The selected 10 000 publications are split into a training and testing parts in the ratio 9:1.

We applied the TF-IDF encoding to obtain the DTM matrices of the training and testing parts according to the steps given at the beginning of this section. The vocabulary consists of *14553* terms from which we selected *1670* terms that occur in at least 99.9% of texts to represent input features of datasets.

In a similar way, we obtained the *balanced* PubMed dataset, which differs from the imbalanced in that it contains the 10 000 documents that are evenly distributed in the 16 main categories.

V. Experiments

In this section, we first describe the applied input parameters of the selected classifiers. Next, we present the results of experiments on the selected datasets. The results were obtained for the three above-described implementations of SNNs as well as for the other four classification methods: Binomial logistic regression, a single Decision tree, the MLP neural network as well as Support Vector Classifier (SVC). In the second part of the experiments, we focus specifically on the presentation of the classification accuracy for the eSNN encoding method that is offered by us in Section III.

A. Parameters of Selected Classifiers

The parameters of the selected SNNs implementations that were ran on the datasets are given in Table II. The parameters were selected using the grid search procedure on a suitable set of parameters of each implementation.

The parameters of the other classifiers selected for experiments were as follows:

- Decision tree - spilt criterion = Gini index, maximal depth = none.
- Logistic regression - maximal no. of training iterations = 100.
- Multilayer Perceptron neural network (MLP) - no. of hidden neurons = 1000, sigmoid activation function, learning rate for weights modification in thw error backpropagation phase 0.0001, training iterations = 8, the ADAM optimization method for error backpropagation.
- Support Vector Classifier - radial kernel, misclassification cost coefficient = 1.

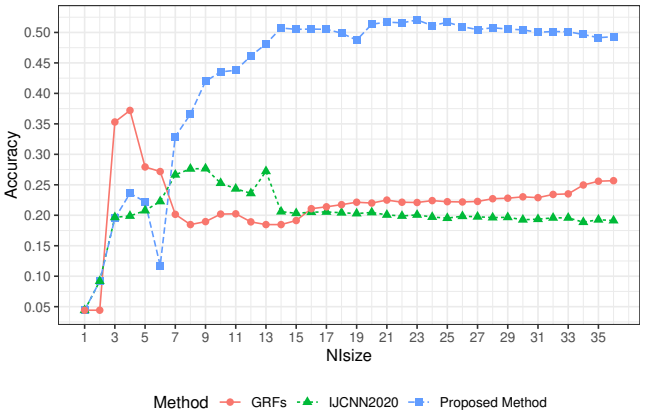

Fig. 6. Comparison of the classification accuracy for the selected input methods. *IJCNN2020* referes to the method offered in [5]. The method given in this work was ran with $Bins = 3$.

B. Results of Short Text Classification

The results of classification accuracy for the classifiers are given in Table III. As it can be noted, in the case of short texts of the 20-newsgroup dataset, the most effective classifier is SVC. For both balanced and imbalanced PubMed datasets, two best performing classifiers are SNNTorch and logistic regression. Among the selected SNNs implementations, the eSNN implementation provides slightly better results than NeuCube for 20-newsgroup dataset, while NeuCube is slightly superior to eSNNs in the cases of PubMed datsets. The best performing implementation of SNNs is SNNTorch. This can be explained by the fact that it contains not only spiking LIF neurons, but also incorporates learning mechanisms of traditional feedforward neural networks, such as the error backpropagation phase that applies ADAM optimizer.

C. Comparison of the Classification Accuracy for the Proposed Encoding Method of eSNNs

Our method is compared with the encoding method offered in [5] that directly calculates firing order of input neurons as well as with the widely-used GRFs method used with spiking neural networks [13]. Both the method of [5] as well as the GRFs uniformly allocate the NI_{size} input neurons to encode input values of each feature $F \in \mathcal{F}$. In this experiments we use 20-newsgroup dataset, however to better illustrate the results of encoding we selected full texts of this dataset.

In Fig. 6, we present the obtained classification accuracy of the selected methods for varying number of input neurons NI_{size}. For the proposed method, we used the $B = 3$ bins parameter to create the histogram of values of each feature in our encoding method. As it can be noticed from the figure, the offered method can significantly improve the classification accuracy comparing to the other two tested methods. For example, for $NI_{size} = 20$ the proposed method gives accuracy 0.52, while the method of [5] and GRFs method give 0.2 and 0.21 accuracy values, respectively.

TABLE II
THE APPLIED PARAMETERS OF THE SELECTED SNN CLASSIFIERS.

Classifier	20 newsgroup dataset	Imbalanced PubMed dataset	Balanced PubMed dataset
eSNN	NI_{size} = 25, $Bins = 3$, mod = 0.95, $simTr$ = 0.05,	NI_{size} = 15, $Bins = 3$, mod = 0.95, $simTr$ = 0.05	NI_{size} = 15, $Bins = 3$, mod = 0.95, $simTr$ = 0.05
NeuCube	η = 0.01, Firing thr. = 0.5, *mod* = 0.95, Refractory time = 3, Training its. = 1	η = 0.01, Firing thr. = 0.5, *mod* = 0.4, Refractory time = 8, Training its. = 1	η = 0.01, Firing thr. = 0.5, *mod* = 0.4, Refractory time = 8, Training its. = 1
SNNTorch	Hidden neu. = 1000, Firing thr. = 1, Decay rate = 0.92, Learn. rate = 0.0001, Training its. = 8	Hidden neu. = 1000, Firing thr. = 1, Decay rate = 0.92, Learn. rate = 0.0001, Training its. = 8	Hidden neu. = 1000, Firing thr. = 1, Decay rate = 0.92, Learn. rate = 0.0001, Training its. = 8

TABLE III
THE OBTAINED CLASSIFICATION ACCURACY RESULTS.

Classifier	20 newsgroup	Imbalanced PubMed	Balanced PubMed
eSNN	0.19	0.15	0.16
NeuCube	0.10	0.17	0.29
SNNTorch	0.24	**0.39**	**0.39**
Decision tree	0.43	0.25	0.26
Logistic regression	0.55	0.36	0.32
MLP network	0.01	0.32	0.31
SVC	**0.61**	0.37	0.37

VI. DISCUSSION AND CONCLUSIONS

In this work, we presented the results of short text classification using three different implementations of SNNs networks, namely: evolving Spiking Neural Networks, the NeuCube implementation of SNNs and the SNNTorch implementation. In order to test the selected classifiers, we selected and preprocessed three publicly available datasets: 20-newsgroup dataset as well as imbalanced and balanced PubMed datasets of medical publications. The preprocessed 20-newsgroup dataset consists of the first 100 words of each text, while for the classification of PubMed datasets we used only a title of each publication. As a text representation of documents, we applied the TF-IDF encoding. In this work, we also offered a new encoding method for eSNN networks, that can effectively encode unevenly distributed values of each input feature. The designed method works especially effectively with the TF-IDF encoding.

The presented results of experiments indicate, that SNNs implementations that solely use the neuronal models traditionally applied in SNNs, such as the LIF model, as well as apply unsupervised learning rules like STDP, may not perform as effective as the implementations that combine SNNs with the learning methods present in traditional neural networks, such as the MLP networks. Specifically, in the conducted experiments, the SNNTorch implementation performed better than the eSNN and NeuCube implementations. Furthermore, the computational and memory complexity of SNN networks (as in the case of NeuCube) can be a bottleneck in processing large sets of texts. In the experiments, SNNTorch was able to slightly outperform the results obtained by other selected classifiers in the case of two PubMed datasets.

REFERENCES

[1] J. Weissbock, A. A. Esmin, and D. Inkpen, "Using external information for classifying tweets," in *2013 Brazilian Conference on Intelligent Systems*, 2013, pp. 1–5.

[2] M. Kozlowski and H. Rybinski, "Clustering of semantically enriched short texts," *Journal of Intelligent Information Systems*, vol. 53, no. 1, pp. 69–92, 2019.

[3] I. Laña, J. L. Lobo, E. Capecci, J. Del Ser, and N. Kasabov, "Adaptive long-term traffic state estimation with evolving spiking neural networks," *Transportation Research Part C: Emerging Technologies*, vol. 101, pp. 126 – 144, 2019.

[4] P. S. Maciąg, N. Kasabov, M. Kryszkiewicz, and R. Bembenik, "Air pollution prediction with clustering-based ensemble of evolving spiking neural networks and a case study for london area," *Environmental Modelling & Software*, vol. 118, pp. 262 – 280, 2019.

[5] P. S. Maciąg, M. Kryszkiewicz, and R. Bembenik, "Online evolving spiking neural networks for incremental air pollution prediction," in *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1–8.

[6] H. Liu, G. Lu, Y. Wang, and N. Kasabov, "Evolving spiking neural network model for PM2.5 hourly concentration prediction based on seasonal differences: A case study on data from beijing and shanghai," *Areosol and Air Quality Research*, vol. 21, no. 2, p. 200247, 2021.

[7] L. Paulun, A. Wendt, and N. Kasabov, "A retinotopic spiking neural network system for accurate recognition of moving objects using neucube and dynamic vision sensors," *Frontiers in Computational Neuroscience*, vol. 12, p. 42, 2018.

[8] P. S. Maciąg, M. Kryszkiewicz, R. Bembenik, J. L. Lobo, and J. Del Ser, "Unsupervised anomaly detection in stream data with online evolving spiking neural networks," *Neural Networks*, vol. 139, pp. 118–139, 2021.

[9] K. Demertzis and L. Iliadis, "A hybrid network anomaly and intrusion detection approach based on evolving spiking neural network classification," in *E-Democracy, Security, Privacy and Trust in a Digital World*, A. B. Sideridis, Z. Kardasiadou, C. P. Yialouris, and V. Zorkadis, Eds. Cham: Springer International Publishing, 2014, pp. 11–23.

[10] N. K. Kasabov, "Neucube: A spiking neural network architecture for mapping, learning and understanding of spatio-temporal brain data," *Neural Networks*, vol. 52, pp. 62–76, Apr. 2014.

[11] N. Kasabov and E. Capecci, "Spiking neural network methodology for modelling, classification and understanding of eeg spatio-temporal data measuring cognitive processes," *Information Sciences*, vol. 294, pp. 565 – 575, 2015, innovative Applications of Artificial Neural Networks in Engineering.

[12] J. K. Eshraghian, M. Ward, E. Neftci, X. Wang, G. Lenz, G. Dwivedi, M. Bennamoun, D. S. Jeong, and W. D. Lu, "Training spiking neural networks using lessons from deep learning," *arXiv preprint arXiv:2109.12894*, 2021.

[13] J. L. Lobo, I. Oregi, A. Bifet, and J. Del Ser, "Exploiting the stimuli encoding scheme of evolving spiking neural networks for stream learning," *Neural Networks*, vol. 123, pp. 118 – 133, 2020.

[14] Q. Li, H. Peng, J. Li, C. Xia, R. Yang, L. Sun, P. S. Yu, and L. He, "A survey on text classification: From traditional to deep learning," vol. 13, no. 2, apr 2022. [Online]. Available: https://doi.org/10.1145/3495162

[15] J. Y. Lee and F. Dernoncourt, "Sequential short-text classification with recurrent and convolutional neural networks," *arXiv preprint arXiv:1603.03827*, 2016.

[16] Chen, Yahui, "Convolutional neural network for sentence classification," Master's thesis, 2015. [Online]. Available: http://hdl.handle.net/10012/9592

[17] Y. Hu, Y. Li, T. Yang, and Q. Pan, "Short text classification with a convolutional neural networks based method," in *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*. IEEE, 2018, pp. 1432–1435.

[18] L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 7370–7377, Jul. 2019.

[19] K. Kowsari, K. Jafari Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, and D. Brown, "Text classification algorithms: A survey," *Information*, vol. 10, no. 4, p. 150, 2019.

[20] M. Białas, M. M. Mirończuk, and J. Mańdziuk, "Biologically plausible learning of text representation with spiking neural networks," in *Parallel Problem Solving from Nature – PPSN XVI*, T. Bäck, M. Preuss, A. Deutz, H. Wang, C. Doerr, M. Emmerich, and H. Trautmann, Eds. Cham: Springer International Publishing, 2020, pp. 433–447.

[21] Y. Wang, Y. Zeng, J. Tang, and B. Xu, "Biological neuron coding inspired binary word embeddings," *Cognitive Computation*, vol. 11, no. 5, pp. 676–684, 2019.

[22] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[23] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International conference on machine learning*, 2014, pp. 1188–1196.

[24] J. L. Lobo, I. Laña, J. Del Ser, M. N. Bilbao, and N. Kasabov, "Evolving spiking neural networks for online learning over drifting data streams," *Neural Networks*, vol. 108, pp. 1 – 19, 2018.

[25] B. Petro, N. Kasabov, and R. M. Kiss, "Selection and optimization of temporal spike encoding methods for spiking neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–13, 2019.

[26] E. M. Izhikevich and G. M. Edelman, "Large-scale model of mammalian thalamocortical systems," *Proceedings of the National Academy of Sciences*, vol. 105, no. 9, pp. 3593–3598, 2008.

[27] F. Ponulak and A. Kasiński, "Supervised learning in spiking neural networks with resume: sequence learning, classification, and spike shifting," *Neural computation*, vol. 22, no. 2, pp. 467–510, 2010.

[28] F. Ponulak and A. Kasinski, "Introduction to spiking neural networks: Information processing, learning and applications." *Acta neurobiologiae experimentalis*, vol. 71, no. 4, pp. 409–433, 2011.

[29] I. Feinerer, "Introduction to the tm package text mining in R," *Avail. on line: http://cran.r-project.org/web/packages/tm/vignettes/tm.pdf*, 2013.

[30] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Valletta, Malta: ELRA, May 2010, pp. 45–50, http://is.muni.cz/publication/884893/en.

[31] BioASQ Team. (2021) A challenge in large-scale biomedical semantic indexing and question answering. [Online]. Available: http://www.bioasq.org/participate/challenges