A novel method for extracting interpretable knowledge from a spiking neural classifier with time-varying synaptic weights

Abeegithan Jeyasothy\textsuperscript{a,c,*}, Suresh Sundaram\textsuperscript{b}, Savitha Ramasamy\textsuperscript{c}, Narasimhan Sundararajan\textsuperscript{a}

\textsuperscript{a}School of Computer Science and Engineering, Nanyang Technological University, Singapore
\textsuperscript{b}Department of Aerospace Engineering, Indian Institute of Science, Bangalore, India
\textsuperscript{c}Institute for Infocomm Research, Agency for Science, Technology and Research, Singapore

Abstract

This paper presents a novel method for information interpretability in a Multi-Class Synaptic Efficacy Function based leaky-integrate-fire neuron (MC-SEFRON) classifier that uses time-varying synaptic weights. To develop a method to extract knowledge stored in a trained multi-class classifier, first, the binary-class SEFRON classifier developed earlier is extended to handle multi-class problems. For a given input sample, MC-SEFRON uses the population encoding scheme to encode the real-valued input data into spike patterns. MC-SEFRON is trained using the same supervised learning rule given in the binary-class SEFRON classifier. After training, the proposed knowledge encoding method extracts the knowledge for a given class stored in the classifier by mapping the weighted postsynaptic potential in the time domain to the feature domain as Feature Strength Functions (FSFs). A set of FSFs corresponding to each output class represents the extracted knowledge from the MC-SEFRON classifier. This knowledge encoding method is derived to maintain consistency between the classification in the time domain and the feature domain. Also, the correctness of the extracted knowledge is quantitatively measured by using the FSFs directly for classification tasks. For a given input, each FSF is sampled at the input value to obtain the corresponding feature strength value. Then the aggregated feature strength values obtained for each class are used to determine the output class labels during classification. For a given input, feature strength values are used to interpret the predictions during the classification task. Using ten benchmark UCI machine learning datasets and the MNIST dataset, the knowledge extraction method, interpretation and the reliability of the extracted knowledge are demonstrated. Based on the studies, it can be seen that on an average, the difference in the classification accuracies using the extracted knowledge directly and those obtained by MC-SEFRON classifier is only around 0.9% & 0.1% for UCI machine learning datasets and the MNIST dataset respectively. This clearly shows that the knowledge represented by the FSFs of the MC-SEFRON has acceptable reliability and the interpretability of classification using the classifier’s knowledge has been justified.

Keywords: Interpretable classifier, Knowledge extraction, Time-varying weight model, Multi-class classification, Spiking neural network, Spike-Timing-Dependent Plasticity

1. Introduction

Although artificial neural networks have great potential for prediction and pattern recognition in several applications, they have remained mostly as black-boxes that are difficult to interpret the reasons for their predictions. This renders them mostly uncertain and unreliable, especially, for decision making in sensitive applications like healthcare. Only recently, there has been considerable research attention towards developing interpretable machine learning approaches.

*Corresponding author
Email address: abeegith1@e.ntu.edu.sg (Abeegithan Jeyasothy)
Earlier methods on deriving interpretability from machine learning classifiers focused on gradient descent-based sensitivity analysis methods tailored to specific classification techniques. For example, gradient propagation from the output layer to the input layer in a convolutional neural network enables one to visualize class sensitive input regions in Simonyan et al. (2013). On the other hand, gradients are propagated through a deconvolution method in Zeiler & Fergus (2014) to map the relationship between specific regions of the input to the inferences of a convolutional neural network. During the sensitivity analysis, the gradient values coming into the Rectified Linear Unit (ReLU) are set to zero if the input value to the ReLU is negative during the forward pass in Simonyan et al. (2013), whereas the gradient values coming into the ReLU are set to zero only if it is negative in Zeiler & Fergus (2014). Guided backpropagation combines both the above approaches in Simonyan et al. (2013); Zeiler & Fergus (2014) and is used in Springenberg et al. (2014), where the gradient values coming into ReLU are set to zero if either the gradient value or input during the forward pass is negative. However, these methods suffer in determining the input regions that are negatively influencing the inference. Propagation of importance score is conserved within the layers of the network in Bach et al. (2015) to include the negatively influencing input regions. As there are class dependencies in these important scores, Deep Learning Important FeaTures (DeepLIFT) Shrikumar et al. (2017) computes the changes in the importance scores for input regions for each class.

In addition to these gradient-based sensitivity analysis methods, there exist input perturbation-based sensitivity analysis methods to interpret the significance of individual features to the inferences in each class Ribeiro et al. (2016); Lundberg & Lee (2017). These methods are often validated by eliminating the ranked features in sequence to establish their empirical significance to the obtained inference Samek et al. (2017); Shrikumar et al. (2017); Lundberg & Lee (2017). Although all the above methods help to establish empirical relationships between the set of input features to their corresponding model-based predictions, they do not quantify the uncertainty measures of their interpretations.

An alternative approach in explaining the prediction of a classifier is by expressing the knowledge acquired after training the neural network in an easily explainable form Hinton et al. (2015); Tan et al. (2018); Che et al. (2016). In Frosst & Hinton (2017), soft targets from the deep neural network are used to train a soft decision tree to interpret the predictions. A decision tree is learned to provide semantic level explanations for convolutional neural network predictions in Zhang et al. (2018). In these methods, the classification accuracy of the new explainable model is used to show the reliability of the explanations for the classifier’s predictions. However, these explainable models are not extracted from the trained classifier rather they are also a newly trained classifier that produces similar outputs as the base-classifier. Due to the fundamental differences in the classifiers, the interpretation of the prediction inferred from these new classifiers may not be aligned with the base-classifier’s predictions.

Although there are several studies on interpreting decisions made by deep neural networks as indicated above, these methods do not have a framework to validate the consistency of the explanation with the classifier’s prediction. Also due to the difference in the activation unit (spiking neuron), these methods may require major modifications for use in Spiking Neural Networks (SNNs). Even though SNNs are energy efficient and hardware friendly compared to other artificial neural networks as mentioned in the recent review Tavanaei et al. (2019), SNNs still remain as black-boxes and hardly any study on deriving an interpretation from the trained SNN classifiers exist. To overcome these problems, we propose a new method herein to extract the knowledge stored in a spiking neural classifier with time-varying weights that is also reliable and consistent.

In this paper, a multi-class classifier for the spiking neuron with time-varying weight model Jeyasothy et al. (2018), referred (hereafter) to as a Multi-Class-SEFRON (MC-SEFRON) is first developed. MC-SEFRON classifier is trained using the modified Spike-Timing-Dependent Plasticity (STDP) rule developed earlier in Jeyasothy et al. (2018). In a trained MC-SEFRON classifier, the input information is encoded in synaptic efficacy functions (time-varying weight functions). Interpreting the predictions made by MC-SEFRON in the time domain is very challenging. However, interpreting and visualizing the predictions are much easier in the feature domain. Hence, in this paper, we propose a new knowledge encoding method to extract knowledge from a trained MC-SEFRON classifier by mapping the weighted postsynaptic potentials in the time domain into the actual feature space as functions of the features, referred to as Feature Strength Functions (FSF). FSF is the new interpretable form of an MC-SEFRON classifier in the feature domain. A
set of FSFs provides the representation of the knowledge learned by a trained MC-SEFRON classifier. This new knowledge encoding method is derived to maintain the consistency between the classifications made by the MC-SEFRON classifier and the FSF both in the time domain and the feature domain respectively. FSF is then used to demonstrate the interpretability of MC-SEFRON’s predictions during classification.

The correctness of using the FSF for classification directly is measured by its performance (classification accuracy) on the same classification task used for the MC-SEFRON classifier. For a given input, sampling the FSFs for a given output class at the input feature values gives the corresponding feature strength values. During the classification using FSF, output class label for a given input sample is predicted by the output class that corresponds to the highest aggregated feature strength value. The explanation for prediction during the classification is provided based on both the given input sample’s individual and aggregated feature strength values. Performance of both the MC-SEFRON and FSF are evaluated using ten benchmark data sets from the UCI machine learning repository and the MNIST dataset. Based on the study results, it can be seen that on an average the difference between the classification accuracy of MC-SEFRON and FSF (for the same classification problem) is around 0.9% for the datasets from UCI machine learning repository and is 0.1% for the MNIST dataset.

The paper is organized as follows: first, the multi-class classification problem formulation and the learning algorithm for an MC-SEFRON classifier are presented in Section 2. In Section 3 the detailed framework for extracting interpretable knowledge from the MC-SEFRON classifier is presented. Section 4 presents the performance and the interpretability of the MC-SEFRON classifier using both the ten UCI machine learning datasets and the MNIST dataset. Finally, the conclusions from the study are summarised in Section 5.

2. MC-SEFRON classifier for multi-class problems

MC-SEFRON is a spiking neural classifier without any hidden layers, where input neurons are directly connected to output neurons via time-varying weight models (synaptic efficacy functions). The architecture of an MC-SEFRON classifier is shown in Fig 1. Here the weight between an input and an output neuron is a time-varying function instead of a fixed value. The time-varying weight model is represented by a sum of multiple time-varying kernels, here Gaussian kernels (time-varying functions) are used and they are learned by the supervised learning rule as given in Jeyasothy et al. (2018). In MC-SEFRON, a modified STDP Markram et al. (2012) rule is used to determine the required weight updates corresponding to each presynaptic spike. The amplitudes and the centers of the time-varying kernels are determined by the amplitudes of the weight updates and the time of the corresponding presynaptic spikes, respectively.

2.1. Multi-class classification problem

Here the function $S(\cdot)$ represents an ideal MC-SEFRON classifier. For an input $x = [x_1, x_2, \ldots, x_m]$ where $x \in [0,1]^m$ (here $m$ is the number of features in the input data), $S(x)$ produces an output $y = [y_1, y_2, \ldots, y_n]$ where $y$ is the vector of first postsynaptic spike time from each output classes and $y \in [0, T + \delta T]^n$. Here $n$ is the total number of output classes and $T$ is the presynaptic spike interval limit and $\delta T$ represents an incremental smaller time after $T$ allow late postsynaptic spikes.

The classification rule for MC-SEFRON is the same rule as in other multi-class SNN classifiers [Bohte et al. (2002)]. In this classification rule, output class label for an input pattern is determined by the postsynaptic
neuron (output neuron) that fires first and the class label $\hat{c}$ is predicted as,

$$\hat{c} = \arg\min_j \{y_j\}$$ (2)

By examining equation 1 and 2, it can be observed that an ideal classifier will always predict the correct class for a given input sample. In this paper, MC-SEFRON is trained to approximate the ideal classifier function $S(.)$. It may be noted that the input $x$ is in the feature domain and the output $y$ is in the time domain. Hence, coded output class labels for a supervised learning framework are chosen as some desired postsynaptic spike times. In this paper, the same supervised learning rule proposed in Jeyasothy et al. (2018) is used to determine a reference postsynaptic spike time $\hat{t}_{rf}^j$ for each class as the coded output class label. The approximated classifier function is denoted as $\hat{S}(.)$, where the $\hat{S}(x)$ produces an output $\hat{y} \ (y = \hat{S}(x) + \varepsilon = y + \varepsilon)$. The approximation error $\varepsilon$ is generally caused by differences within the same class input patterns.

2.2. Population encoding scheme

Since the input $x$ is real-valued, first, an encoding scheme is required to convert the real values to spike trains. A real-valued input $x$ is encoded into a spike pattern $s$ using the population encoding scheme. In the population encoding scheme, multiple Gaussian functions with evenly spaced centres are used as Receptive Field (RF) neurons and the input data is projected into a higher dimensional space ($\mathbb{R}^{m \times q}$, where $q$ is the number of RF neurons used in the population encoding) as shown in Fig 1.

For a given input $x$, each RF neuron produces a firing strength $\phi_{ri}$, where $i \ (i \in \{1, m\})$ represents the $i^{th}$ input feature and $r \ (r \in \{1, q\})$ represents the $r^{th}$ RF neuron. The firing strength $\phi_{ri}$ determines the presynaptic spike time $s_{ri}^j \left( s_{ri}^j = T \cdot (1 - \phi_{ri}) \right)$ for the $r^{th}$ RF neuron corresponding to the $i^{th}$ input feature.
A higher RF neuron firing strength ($\phi_i^r$ closer to 1) corresponds to an early presynaptic firing ($s_i^r$ closer to 0s) and a lower RF neuron firing strength ($\phi_i^r$ closer to 0) correspond to a late presynaptic firing ($s_i^r$ closer to $T_s$).

A generic population encoding scheme is denoted by $G(.)$ and given by,

$$G(x) = s$$  \hspace{1cm} (3)

where $s \in [0, T]^{m \times q}$. For an $i^{th}$ input feature $x_i$, an encoded presynaptic spike pattern $s_i$ is obtained by,

$$G(x_i) = s_i = \{s_i^1, s_i^2, \ldots, s_i^q\}$$  \hspace{1cm} (4)

It may be noted that in this scheme each presynaptic neuron will fire only one spike for a given input pattern. Hence, there are $m \times q$ number of presynaptic neurons in the classifier.

2.3. MC-SEFRON’s learning algorithm

In a MC-SEFRON classifier, $j^{th}$ output neuron fires a postsynaptic spike $\hat{t}_j$ when the postsynaptic potential $v_j(t)$ crosses its firing threshold $\theta_j$.

$$\hat{t}_j = \{t | v_j(t) = \theta_j\}$$  \hspace{1cm} (5)

Here, the postsynaptic potential $v_j(t)$ of $j^{th}$ output neuron is determined as,

$$v_j(t) = \sum_{i \in \{1,m\}} \sum_{r \in \{1,q\}} w_{ir}^r(s_i^r,j).\epsilon(t - s_i^r) . H(t - s_i^r)$$  \hspace{1cm} (6)

where, $\epsilon(t)$ is the spike response function as in [Jeyasothy et al. 2018] ($\epsilon(t) = \frac{1}{\tau} . \exp\left(1 - \frac{t}{\tau}\right)$). $H(t)$ represents a Heaviside step function. $w_{ir}^r(s_i^r,j)$ is the momentary weight at the presynaptic spike time $s_i^r$ of the synapse (connection) between the $j^{th}$ output neuron and the input neuron corresponding to the $r^{th}$ RF neuron of the $i^{th}$ input feature. $w_{ir}^r(s_i^r,j)$ is obtained by sampling the time-varying weight function $w_{ir}^r(t,j)$ at the time instant $s_i^r$ as shown in Fig 1.

In MC-SEFRON, synaptic efficacy functions are initialized using the first sample from the each classes. For a $j^{th}$ output neuron, synaptic efficacy function $w_{ir}^r(t,j)$ of the $r^{th}$ ($r \in \{1,q\}$) RF neuron of the $i^{th}$ input feature is initialized as,

$$w_{ir}^r(t,j)_{\text{initial}} = u_{ir}^{t}(\hat{t}_d).\exp\left(\frac{-(t - s_i^r)^2}{2\sigma^2}\right)$$  \hspace{1cm} (7)

Here, $u_{ir}^{t}(\hat{t}_d)$ is the normalized STDP with respect to the desired postsynaptic firing time $\hat{t}_d$. A generic $u_{ir}^{t}(\hat{t})$ with respect to a given postsynaptic spike time $\hat{t}$ is calculated as,

$$u_{ir}^{t}(\hat{t}) = \begin{cases} \frac{\exp(-|\hat{t} - s_i^r|/\tau)}{\sum_{i=1}^{m} \sum_{r=1}^{q} \exp(-|t - s_i^r|/\tau)} & \text{for all } s_i^r \leq \hat{t} \\ \frac{-\exp(-|\hat{t} - s_i^r|/\tau)}{\sum_{i=1}^{m} \sum_{r=1}^{q} \exp(-|t - s_i^r|/\tau)} & \text{for all } s_i^r > \hat{t} \end{cases}$$  \hspace{1cm} (8)

where, the sum of $u_{ir}^{t}(\hat{t})$ corresponding to presynaptic spikes fired before and after $\hat{t}$ are equal to 1 and $-1$ respectively. The firing threshold $\theta_j$ of $j^{th}$ output neuron is initialized as,

$$\theta_j = \sum_{i \in \{1,m\}} \sum_{r \in \{1,q\}} u_{ir}^{t}(\hat{t}_d).\epsilon(\hat{t}_d - s_i^r)$$  \hspace{1cm} (9)
During training, if the correct class output neuron fires earlier than the other class output neurons with a marginal time \( t_m \), then the sample is not used to update any of the \( w_i^r(t, j) \). Synaptic efficacy functions are only updated if a wrong class output neuron fires within the marginal time during a correct classification or if there is a misclassification. The weight update of the connection between the \( j^{th} \) output neuron and the input neuron corresponding to the \( r^{th} \) RF neuron of the \( i^{th} \) input feature is denoted by \( \Delta w_i^r(s_i^r, j) \) and determined as,

\[
\Delta w_i^r(s_i^r, j) = \lambda . w_i^r(t_i^j). \theta_j . \left( \frac{1}{\sum_{i=1}^{m} \sum_{r=1}^{q} w_i^r(t_i^j).e(t_d - s_i^r)} - \frac{1}{\sum_{i=1}^{m} \sum_{r=1}^{q} w_i^r(t_j).e(t_j - s_i^r)} \right)
\]

(10)

Here, \( \lambda \) is the learning rate, \( t_i^j \) is the actual postsynaptic firing time of the \( j^{th} \) output neuron and \( t_i^f \) is the reference postsynaptic spike time (coded output signal for supervised learning). Please refer to Jeyasothy et al. (2018) for a detailed discussion on deriving the weight update rule. For an input belonging to \( j^{th} \) output class, during a wrong classification, \( t_i^f \) is set to \( t_d \) and \( t_h^f (h \neq j) \) is set to \( min(\hat{t}_j + t_m, T + \delta T) \). Algorithm 1 presents the pseudocode for determining \( t_i^f \) and also training the MC-SEFRON classifier.

The change in weight \( \Delta w_i^r(s_i^r, j) \) is a single value and it has been embedded in a time-varying function \( g_i^r(t, j) \) as,

\[
g_i^r(t, j) = \Delta w_i^r(s_i^r, j).\exp\left(\frac{-(t - s_i^r)^2}{2\sigma^2}\right)
\]

(11)

where \( \sigma \) is the efficacy update range. More information on a single synapse can be stored by setting a smaller value to \( \sigma \) whereas an infinite value for \( \sigma \) results in a single-weight model instead of a time-varying weight model.

For the \( j^{th} \) output neuron, the synaptic efficacy function of the \( r^{th} \) RF neuron for the \( i^{th} \) input feature is updated as,

\[
w_i^r(t, j)_{\text{new}} = w_i^r(t, j)_{\text{old}} + g_i^r(t, j)
\]

(12)

**Algorithm 1:** Pseudo code to train a MC-SEFRON classifier

```plaintext
for all samples do
    if first sample from class j then
        Initialize \( w_i^r(t, j) \) for all \( i \) and \( r \) and firing threshold \( \theta_j \) (using equation 7 and 9 respectively);
    else
        if sample belongs to class j then
            \( \hat{t}_j \) ← Postsynaptic spike time of class \( j \) output neuron;
            \( \hat{t}_h \) ← Postsynaptic spike times of class \( h \) (\( h \neq j \)) output neurons;
            if \( \hat{t}_j + t_m \leq min(\hat{t}_h) \) then
                The sample is not used to update the network parameters.;
            else
                if \( \hat{t}_j > \hat{t}_d \) then
                    Update \( w_i^r(t, j) \) for all \( i \) and \( r \) with \( \hat{t}_i^j = \hat{t}_d \);
                end
            end
        end
        for all output class \( h \) (\( h \neq j \)) do
            if \( \hat{t}_h < \hat{t}_j + t_m \) then
                Update \( w_i^r(t, h) \) for all \( i \) and \( r \) with \( \hat{t}_i^h = min(\hat{t}_j + t_m, T + \delta T) \);
            end
        end
    end
end
```
3. Knowledge encoding method to extract interpretable knowledge from MC-SEFRON

The previous section laid the groundwork for building and training an MC-SEFRON classifier. A well-trained classifier may have a better representation of knowledge on the dataset compared to a weak classifier. In this section, a framework is derived to extract the knowledge encoded in a trained MC-SEFRON classifier.

The classification rule in equation 2 for an ideal MC-SEFRON classifier can be re-written as,

\[ \hat{c} = \arg \min_{j} \{ y \} \]
\[ = \arg \min_{j} \{ t \mid \frac{1}{\theta_j} v_j(t) > 1 \} \]
\[ = \arg \min_{j} \{ t \mid \frac{1}{\theta_j} \sum_{i=1}^{m} \sum_{r=1}^{q} w_{i}^{r}(s_{r}^{c}, j).\epsilon(t - s_{r}^{c}).H(t - s_{r}^{c}) > 1 \} \] (13)

For a given input, an ideal classifier would fire the postsynaptic spikes at the desired firing times. Therefore, at the time of classification, the correct output neuron would have fired a postsynaptic spike at \( t_d \) and the remaining output neurons would fire postsynaptic spikes later at \( t_d + t_m \) (refer to equation 1). Hence, the term \( \frac{1}{\theta_j} \sum_{i=1}^{m} \sum_{r=1}^{q} w_{i}^{r}(s_{r}^{c}, j).\epsilon(t - s_{r}^{c}).H(t - s_{r}^{c}) \) is maximum at \( t = t_d \) for the correct output class. Therefore, the classification rule can be modified as,

\[ \hat{c} = \arg \max_{j} \frac{1}{\theta_j} \sum_{i=1}^{m} \sum_{r=1}^{q} w_{i}^{r}(s_{r}^{c}, j).\epsilon(t_d - s_{r}^{c}).H(t_d - s_{r}^{c}) \]
\[ = \arg \max_{j} \frac{1}{\theta_j} \sum_{i=1}^{m} \sum_{r=1}^{q} w_{i}^{r}(s_{r}^{c}, j).\epsilon(t_d - s_{r}^{c}).H(t_d - s_{r}^{c}). \] (14)

The classification rule in (14) is only applicable for an ideal classifier as \( t_d \) is always the earliest firing time in an ideal classifier. However a trained MC-SEFRON classifier (\( \hat{S}(.) \)) produces an output \( \hat{y} \) that only approximates the ideal output \( y \) (produced by \( S(.) \)). Hence, the earliest firing time may not be same as the desired firing time due to the differences in the input patterns. Therefore, the earliest firing time has to be approximated to avoid this loss in performance. Here \( t_o \) (\( t_o = t_d + \epsilon \)) denotes the approximated earliest firing time. The equation (14) can be rewritten after the approximation as,

\[ \hat{c}^{*} = \arg \max_{j} \frac{1}{\theta_j} \sum_{i=1}^{m} \sum_{r=1}^{q} w_{i}^{r}(s_{r}^{c}, j).\epsilon(t_o - s_{r}^{c}).H(t_o - s_{r}^{c}). \] (15)

The approximated classification rule in equation 15 is preserved during the extraction of the interpretable knowledge. In the earlier section, population encoding scheme was used to convert the real valued input to spike times (refer to section 2.2). Here, an inverse of the population encoding scheme is used to determine the preimage of the spike time under the map of \( G(.) \). Multiple Gaussian functions are used as RF neurons in the population encoding scheme. Hence, in the inverse population encoding scheme, preimage of each presynaptic spike under the map of corresponding RF neuron will have two solutions. However, the collective solutions for the set of presynaptic spikes \( \{s_{1}^{r}, s_{2}^{r}, ..., s_{q}^{r}\} \) have only one common solution \( x_{i} \). Hence, in an inverse population encoding scheme, preimage of the encoded spike pattern \( s_{i} \) (\( s_{i} = \{s_{1}^{r}, s_{2}^{r}, ..., s_{q}^{r}\} \)) has a unique solution \( x_{i} \). The inverse population encoding scheme is given by,

\[ G^{-1}(s_{i}) = \{ x_{i} \mid G(x_{i}) = s_{i} \} \] (16)

Due to the unique solutions in the inverse mapping it can be written as,

\[ s_{i} = G\left(G^{-1}(s_{i})\right) \]
\[ = G(x_{i}) \] (17)

7
Using this inverse population encoding scheme, the feature strength function $\psi_i(x_i, j)$ of the $i^{th}$ input feature for the $j^{th}$ output class is extracted from the weighted postsynaptic potential as,

$$\psi_i(x_i, j) = \frac{1}{\theta_j} \sum_{r=1}^{q} \omega^r_i \left( s^j_r, j \right) \epsilon \left( \hat{t}_o - s^r_i \right) H \left( \hat{t}_o - s^r_i \right)$$

$$= \frac{1}{\theta_j} \sum_{r=1}^{q} \omega^r_i \left( G(x_i)^r, j \right) \epsilon \left( \hat{t}_o - G(x_i)^r \right) H \left( \hat{t}_o - G(x_i)^r \right)$$  (18)

This FSF imitates the input-output function learned by MC-SEFRON as templates of feature functions. Feature strength for a given feature value is sampled from the corresponding FSF. For a given input, the collective sampled feature strength values are used directly for classification and are also used to explain why a particular classification is made for those input values. For a classification task, RHS of the equation 15 is replaced by the FSF in the equation 18. Using FSFs, the output class label $\hat{c}^*$ for a given input sample $x^k = \{ x^k_1, x^k_2, ..., x^k_i, ..., x^k_m \}$ is predicted as,

$$\hat{c}^* = \arg \max_j \sum_{i=1}^{m} \psi_i(x^k_i, j)$$  (19)

here $x^k_i \in [0, 1]$ is the $i^{th}$ feature value. For classification tasks, FSF can be considered as a single input-output layer classifier. Fig 2 shows the architecture for using FSFs for classification tasks directly that relies on equation 19. It can be seen from Fig 2 that for a classification task FSF values are sampled and the feature strength values are summed at the output neuron. The output class label is predicted by that output neuron corresponding to the highest aggregated feature strength value.

Figure 2: Architecture for using FSFs for classification tasks. Zoomed in view of the Feature Strength Function $\psi_i(x_i, j)$ shows the sampling of feature strength value $\psi_i(x^k_i, j)$ for an input feature value $x^k_i$. 

8
4. Performance evaluation of MC-SEFRON and FSF using UCI datasets and MNIST dataset

4.1. Performance evaluation based on UCI datasets

Performance of MC-SEFRON and the accuracy of the extracted knowledge are evaluated using ten benchmark datasets from the UCI machine learning repository for classification tasks. Performance of MC-SEFRON classifier is also compared with other existing SNN classifiers. The population encoding scheme is used to convert the real-valued input data into spike patterns. In the population encoding scheme, the number of RF neurons, overlap constant and the presynaptic spike interval ($T$) are set to 6, 0.7 and 3ms respectively as given in Dora et al. (2018). For all the experiments, $\delta T$, $t_m$ and $\hat{t}_o$ are set to 1ms, 0.05ms and 2ms respectively. For each dataset, the efficacy update range $\sigma$ and the time constant $\tau$ of the STDP rule are chosen using cross validation. To enable fairness in evaluation, for the seven datasets, 10-random fold cross-validation is conducted and for the remaining three datasets a single fold validation is conducted as stated in Dora et al. (2018). Cross-validation is used to choose $\hat{t}_o$ to extract FSFs from the trained MC-SEFRON classifier. Description of the ten UCI machine learning datasets and the chosen values for $\sigma$, $\tau$-STDP and $\hat{t}_o$ are given in table 1.

Table 1: Description of Dataset used for validation

| Dataset               | # Features | # Classes | # Samples | # Samples | $\sigma$ (ms) | $\tau$-STDP (ms) | $\hat{t}_o$ (ms) |
|-----------------------|------------|-----------|-----------|-----------|---------------|-----------------|-----------------|
| Iris                  | 4          | 3         | 75        | 75        | 0.55          | 1.6             | 2.38            |
| Wine                  | 13         | 3         | 60        | 118       | 0.85          | 4.9             | 1.54            |
| Acoustic emission     | 5          | 4         | 62        | 137       | 0.9           | 3.7             | 2.99            |
| Liver                 | 6          | 2         | 170       | 175       | 0.25          | 7.35            | 2.37            |
| Breast Cancer         | 9          | 2         | 350       | 333       | 0.4           | 3.35            | 2.03            |
| Ionosphere            | 34         | 2         | 175       | 176       | 0.3           | 4.5             | 2.64            |
| PIMA                  | 8          | 2         | 384       | 384       | 0.35          | 3.7             | 1.85            |

10 fold cross-validation

| Dataset               | # Features | # Classes | # Samples | # Samples | $\sigma$ (ms) | $\tau$-STDP (ms) | $\hat{t}_o$ (ms) |
|-----------------------|------------|-----------|-----------|-----------|---------------|-----------------|-----------------|
| Image Segmentation    | 18         | 7         | 210       | 2100      | 0.45          | 4.55            | 2.41            |
| EEG eye state         | 14         | 2         | 9990      | 4990      | 0.3           | 3.6             | 2.11            |
| Yeast                 | 8          | 10        | 990       | 490       | 0.35          | 2.05            | 1.85            |

Single fold validation

Experimental results with the 10-random fold cross-validations are compared with SpikeProp Bohte et al. (2002), SWAT Wade et al. (2010), SRESN Dora et al. (2016), and TMM-SNN Dora et al. (2018). Single fold experimental results are compared with SpikeTemp Wang et al. (2017), eSNN Dhoble et al. (2012) and TMM-SNN Dora et al. (2018). All the results used for comparison are reproduced from Dora et al. (2018). Table 2 shows the performance comparison of MC-SEFRON with other algorithms and the performance of FSFs.

From the table 2 it can be seen that the classification performance of MC-SEFRON is on par with other classifiers on the 10-fold cross-validation and outperforms all the classifiers on single fold validation. For the EEG eye state dataset, MC-SEFRON performs 15% better than any other classifier. This also highlights that the performance of MC-SEFRON is better than other SNN classifiers. However, the main focus of this work is not to outperform other algorithms in the classification task but to emphasise on the quality of the knowledge extracted from MC-SEFRON. It can be seen from table 2 that the performance loss is very minimal ($0.0 - 2.5\%$) when FSF is used directly for the classification task. This implies that the knowledge represented by FSFs is reliable. Next, the Iris dataset is used to illustrate the classification by using FSFs that directly represents the knowledge stored in an MC-SEFRON classifier.

4.1.1. Illustration of direct classification using FSFs for the Iris dataset

Iris dataset contains 3 classes, with each input data having 4 features (attributes). A trained MC-SEFRON classifier with accuracies of 98.67% and 97.33% for training and testing datasets respectively is
Table 2: Performance comparison on UCI dataset

| Dataset         | Method   | Classifier | Training/Testing accuracy (%) | Interpretable knowledge | Training/Testing accuracy (%) |
|-----------------|----------|------------|-------------------------------|-------------------------|-------------------------------|
|                 |          |            | 10 fold cross-validation      |                         |                               |
| Iris            | SpikeProp| 97.2/96.7  | -                             |                         | 97.3/96.7                    |
|                 | SWAT     | 96.7/92.4  | -                             |                         |                               |
|                 | SRESN    | 96.9/97.3  | -                             |                         |                               |
|                 | TMM-SNN  | 97.5/97.2  | -                             |                         |                               |
|                 | MC-SEFRON| 98.4/97.1  | -                             |                         | 97.5/94.5                    |
| Wine            | SpikeProp| 99.2/96.8  | -                             |                         |                               |
|                 | SWAT     | 98.6/92.3  | -                             |                         |                               |
|                 | SRESN    | 96.9/91.0  | -                             |                         |                               |
|                 | TMM-SNN  | 100/97.3   | -                             |                         |                               |
|                 | MC-SEFRON| 98.8/94.6  | 97.5/94.5                     |                         |                               |
| Acoustic emission| SpikeProp| 98.5/97.2  | -                             |                         |                               |
|                 | SWAT     | 93.1/91.5  | -                             |                         |                               |
|                 | SRESN    | 93.9/94.2  | -                             |                         |                               |
|                 | TMM-SNN  | 97.6/97.5  | -                             |                         |                               |
|                 | MC-SEFRON| 98.2/97.7  | 97.3/96.0                     |                         |                               |
| Liver           | SpikeProp| 71.5/65.1  | -                             |                         |                               |
|                 | SWAT     | 74.8/60.9  | -                             |                         |                               |
|                 | SRESN    | 60.4/59.7  | -                             |                         |                               |
|                 | TMM-SNN  | 74.2/70.4  | -                             |                         |                               |
|                 | MC-SEFRON| 77.3/69.6  | 75.1/67.2                     |                         |                               |
| Breast Cancer   | SpikeProp| 97.3/97.2  | -                             |                         |                               |
|                 | SWAT     | 96.5/95.8  | -                             |                         |                               |
|                 | SRESN    | 97.7/97.2  | -                             |                         |                               |
|                 | TMM-SNN  | 97.4/97.2  | -                             |                         |                               |
|                 | MC-SEFRON| 98.4/97.4  | 98.0/97.4                     |                         |                               |
| Ionosphere      | SpikeProp| 89.0/86.5  | -                             |                         |                               |
|                 | SWAT     | 86.5/90.0  | -                             |                         |                               |
|                 | SRESN    | 91.9/88.6  | -                             |                         |                               |
|                 | TMM-SNN  | 98.7/92.4  | -                             |                         |                               |
|                 | MC-SEFRON| 94.2/89.7  | 92.0/87.2                     |                         |                               |
| PIMA            | SpikeProp| 78.6/76.2  | -                             |                         |                               |
|                 | SWAT     | 77.0/72.1  | -                             |                         |                               |
|                 | SRESN    | 70.5/69.9  | -                             |                         |                               |
|                 | TMM-SNN  | 79.7/78.1  | -                             |                         |                               |
|                 | MC-SEFRON| 77.5/75.4  | 76.2/74.8                     |                         |                               |
| Image Segmentation| SpikeTemp| 89.1/82.0  | -                             |                         |                               |
|                 | eSNN     | 71.9/70.9  | -                             |                         |                               |
|                 | TMM-SNN  | 96.2/88.9  | -                             |                         |                               |
|                 | MC-SEFRON| 98.1/90.3  | 97.1/90.2                     |                         |                               |
| EEG eye state   | SpikeTemp| 55.4/54.6  | -                             |                         |                               |
|                 | eSNN     | 55.4/54.6  | -                             |                         |                               |
|                 | TMM-SNN  | 55.1/55.2  | -                             |                         |                               |
|                 | MC-SEFRON| 71.4/70.8  | 71.3/70.8                     |                         |                               |
| Yeast           | SpikeTemp| 56.7/31.6  | -                             |                         |                               |
|                 | eSNN     | 50.5/31.4  | -                             |                         |                               |
|                 | TMM-SNN  | 59.3/62.4  | -                             |                         |                               |
|                 | MC-SEFRON| 56.2/55.5  | 56.2/55.5                     |                         |                               |

used to extract the FSFs. Fig 3a, 3c, 3e and 3g show the FSFs of input feature 1, 2, 3 and 4 respectively for all the classes. Three input data $S_1 = [0.083, 0.583, 0.068, 0.083]$, $S_2 = [0.472, 0.083, 0.508, 0.375]$ and $S_3 = [0.556, 0.208, 0.678, 0.75]$ from class-1 ($C_1$), class-2 ($C_2$) and class-3 ($C_3$) respectively are used to show the sampling of feature strength values from FSFs (refer to Fig 2 for the architecture). In Fig 3a, 3c, 3e and 3g, sampling feature strength values from FSFs for $S_1$, $S_2$ and $S_3$ are denoted by symbols that have shapes of 'diamond' (green), 'circle' (blue) and 'square' (black) respectively. Fig 3b, 3d, 3f and 3h show the sampled feature strength values for input feature 1, 2, 3 and 4 respectively.

From Fig 3b, 3d, 3f and 3h, it can be seen that for $S_1$, all the feature strength values are higher for $C_1$. From Fig 3a, 3c, 3e and 3g, it can be seen that for $S_2$, all the feature strength values are higher for $C_2$. From Fig 3e, 3f, 3h and 3g, it can be seen that for $S_3$, all the feature strength values are higher for $C_3$. From Fig 3a, 3c and 3e, it can be seen that for $C_2$, all the feature strength values are higher for $S_2$.
compared to other classes. Hence, it can be said that $S_1$ is easily classifiable as $C_1$. Similarly for $S_3$, all the features are higher for $C_3$ compared to other classes. However, for feature 1 and 3, feature strength values for $C_2$ are very close to that of $C_3$. This makes both the features 2 and 4 to be very significant for $S_3$ during classification. For $S_2$, feature strength values for feature 1 and 2 are higher for $C_3$ and feature strength values for feature 3 and 4 are higher for $C_2$. Individual feature strength values alone are not sufficient to interpret the classification for $S_2$. By looking at the aggregated feature strength values for $S_2$ in Fig 4, it can be observed that class label for $S_2$ is predicted as $C_2$. Hence, it can be said that the feature 3 and 4 are significant for $S_2$ during classification.

Fig 4 shows the aggregated feature strength values for $S_1$, $S_2$, and $S_3$ for all the classes. For example, for $S_1$, the aggregated feature strength value corresponding to $C_1$ ($\sum_i \psi_i(S_1, C_1)$) is higher than that of other classes. Hence, using the equation [19] the class label for $S_1$ is predicted as $C_1$, implying a correct classification. The same can be observed for $S_2$ and $S_3$.

4.2. Experimental results for MNIST dataset

This section presents the classification performance of MC-SEFRON and FSFs for the MNIST dataset [Le-Cun et al. (1998)] and shows the use of the extracted FSFs to interpret the predictions made by the MC-SEFRON classifier. Experiments on MNIST dataset are repeated 10 times to ensure that the performance is not affected by the first sample presented from each class to initialize the $w_{r,i}(t,j)$ and $\theta_j$. In the population encoding scheme, 5 RF neurons are used in all the experiments. For the 10 experiments, average training and testing accuracies of 93.64% and 92.30% were obtained respectively. Cross-validation is used to choose $i_t$ to extract FSFs from the 10 trained MC-SEFRON classifiers. Average accuracies of 93.47% and 92.20% were obtained for the training and testing dataset respectively by using FSFs for classification. Classification accuracies are compared in Table 3 with other SNN methods in the literature.

| Model                        | Architecture                  | Method                      | Classifier’s Accuracy (%) | Interpretable knowledge’s Accuracy (%) |
|------------------------------|-------------------------------|-----------------------------|---------------------------|---------------------------------------|
| Tavanaei et al. 2018         | Spiking CNN                   | STDP learning rule          | 98.60                     | -                                     |
| Lee et al. 2016              | Spiking CNN                   | Backpropagation             | 99.31                     | -                                     |
| Zhao et al. 2015             | Spiking CNN                   | Tempotron learning rule     | 91.29                     | -                                     |
| Fully connected SNN models   |                               |                             |                           |                                       |
| O’Connor et al. 2013         | (784-500-500-10)              | Spiking DBN                 | 94.09                     | -                                     |
| Neftci et al. 2014           | (784+40)-500-10               | Spiking RBM                 | 91.90                     | -                                     |
| Mostafa 2018                 | 784-400-400-10                | Temporal backpropagation    | 97.14                     | -                                     |
| Tavanaei & Maida 2019        | 784-500-150-10                | 3-layer SNN                 | 97.20                     | -                                     |
| Hussain et al. 2014 *        | [25×200]_{10} - 10            | Spiking Cells               | 90.26                     | -                                     |
| MC-SEFRON (This work)        | 784×5 - 10                    | Spiking neuron with time-varying weight Modified STDP rule | 92.30 | 92.20 |

* Full dataset is not used

From Table 3, it can be seen that spiking Convolutional Neural Networks (spiking CNNs) achieve higher accuracies which are comparable to other non-spiking deep learning methods. It can be noted that MC-SEFRON is the only classifier with the simplest architecture (single input-output layer). The accuracy of MC-SEFRON is better than the spiking Restricted Boltzmann Machines (Spiking RBM) and Spiking cells. Here the parameters for the MC-SEFRON classifier were optimized to produce a testing accuracy higher than 91% and this do not purport to be the best performance of MC-SEFRON. However, the main focus of this work is not to achieve the state-of-the-art performance on (MNIST) handwritten digit recognition but to highlight the quality of the knowledge that can be extracted from an SNN with time-varying weight model and also the interpretability of those extracted knowledge.
Figure 3: FSF is used for classification task for three input samples $S_1 = [0.083, 0.583, 0.068, 0.083]$ (diamond), $S_2 = [0.472, 0.083, 0.508, 0.375]$ (circle) and $S_3 = [0.556, 0.208, 0.678, 0.75]$ (square) from class-1, class-2 and class-3 respectively. FSF is sampled at input feature values to obtain the corresponding feature strength values.

Except for MC-SEFRON, all the other methods do not have a framework to extract interpretable knowledge from the trained network. From Table 3, it can be seen that the classification accuracy for the MNIST dataset using interpretable knowledge (FSF) is 92.20% and the loss in accuracy is minimal (0.1%). This
4.2.1. Interpreting MC-SEFRON’s prediction using FSF on MNIST dataset

Figure 5: Synaptic efficacy functions of all the RF neurons for $i^{th}$ feature and the extracted Feature strength function for $i^{th}$ feature from all the Synaptic efficacy functions

(a) Synaptic efficacy functions $w_i(t,j)$ of all RF neurons SEFs in Fig 5a
(b) Extracted Feature strength Function $\psi_i(x_i,j)$ from

Figure 6 shows the examples of correct classification for all the input digits (0 to 9). The first column shows the image of the input sample used and the subsequent columns show the feature strength heatmaps for the given input image and the aggregated feature strength values for each output class. In each row, the output classes corresponding to the correct classes produce the highest aggregated feature strength values for the inputs. This implies a correct classification. Values of heatmap are in the ranges of $[-0.05, 0.05]$, where ‘blue’, ‘green’ and ‘red’ colours represent $-0.05$, $0$ and $0.05$ respectively. The hue between ‘green’ and ‘red’ in the heatmap corresponds to positive feature strength values and hue between ‘blue’ and ‘green’ corresponds to negative feature strength values.
| Input | Feature strength heatmap and aggregated value \((\sum_{i=1}^{m} \psi(x^*_{i}, j))\) for each output class |
|-------|--------------------------------------------------------------------------------------------------|
| ![Image of 0] | ![Heatmap] 1.19 0.69 0.84 0.85 0.83 0.86 0.84 0.77 0.69 0.73 |
| ![Image of 1] | ![Heatmap] 0.65 1.09 0.88 0.88 0.89 0.83 0.81 0.81 0.69 0.73 |
| ![Image of 2] | ![Heatmap] 0.87 0.90 1.25 0.97 0.64 1.01 0.97 0.21 0.80 0.52 |
| ![Image of 3] | ![Heatmap] 0.65 0.75 0.81 0.89 0.80 0.80 0.78 0.55 0.74 0.61 |
| ![Image of 4] | ![Heatmap] 0.74 0.74 0.79 0.76 1.06 0.71 0.79 0.77 0.60 0.82 |
| ![Image of 5] | ![Heatmap] 0.77 0.84 0.69 0.94 0.84 1.08 0.67 0.59 0.82 0.43 |
| ![Image of 6] | ![Heatmap] 0.74 0.71 0.89 0.93 0.88 0.69 1.06 0.48 0.90 0.65 |
| ![Image of 7] | ![Heatmap] 0.79 0.75 0.74 1.04 0.83 0.75 0.57 1.40 0.63 0.88 |
| ![Image of 8] | ![Heatmap] 0.85 0.70 1.08 0.79 0.89 0.87 0.87 0.52 1.19 0.94 |
| ![Image of 9] | ![Heatmap] 0.47 0.89 0.65 0.84 0.95 0.80 0.72 0.64 0.66 1.02 |

Figure 6: Correct classifications and its corresponding feature strength heatmap and aggregated feature strength values for each output class
The reasoning for these predictions can be explained by the feature strength values. It can be seen clearly in the second row (the row corresponding to input digit ‘1’) in figure[6] that each output neuron has a region with negative feature strengths (blue regions) that looks similar to the output class digit. That region acts as the template to match the input samples. Whenever an input image aligns well with the template, the aggregated feature strength increases as some parts of the template become positive (boosting). On the other hand, the aggregated feature strength decreases (weakening) if there is a mismatch. This boosting and weakening mechanism ensures that for a given output class, the correct class samples get higher aggregated feature strengths to make a correct prediction. The same interpretation can be transformed into the time domain where MC-SEFRON classifier actually makes the prediction.

5. Conclusions

In this paper, a novel knowledge encoding method to extract knowledge from a trained Multi-Class SEFRON classifier and its interpretation have been presented. The knowledge encoding method ensures the consistency between the classification in the time domain and feature domain. First, the earlier developed binary-class SEFRON classifier is extended to handle multi-class classification problems. In the MC-SEFRON classifier, input data is encoded into spike patterns using the population encoding scheme. Binary-class SEFRON’s learning rule (modified STDP rule) is used to train the MC-SEFRON classifier. Weights in an MC-SEFRON classifier are time-varying functions. The weighted postsynaptic potentials in the time domain are transformed into the feature domain as functions of features using the knowledge encoding method. Those transformed functions in the feature domain are referred to as Feature Strength Functions (FSF). A set of FSF for each class represents the knowledge extracted from MC-SEFRON classifier for the corresponding class. FSFs enable one to easily interpret the prediction of the classifier. The correctness of the FSF (extracted knowledge) is measured by the classification accuracy when used directly. Aggregated values of feature strengths that are sampled from FSFs for a given input is used for classification. Performance of MC-SEFRON and FSFs have been validated using ten UCI machine learning datasets and the MNIST dataset. MC-SEFRON classifier is trained on the given dataset and the knowledge is extracted to interpret the predictions of the MC-SEFRON classifier. The classification accuracy obtained using the FSFs indicates that the performance using the extracted knowledge is minimal. Hence, one can conclude that the logical explanation provided in the feature domain for the predictions of MC-SEFRON classifier is reliable.

Acknowledgement

This work was supported by the Science and Engineering Research Council of A*STAR (Agency for Science, Technology and Research), Singapore.

References

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., & Samek, W. (2015). On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLOS ONE*, 10, 1–46. URL: http://doi.org/10.1371/journal.pone.0130140

Bohte, S. M., Kok, J. N., & La Poutré, H. (2002). Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, 48, 17–37. doi:10.1016/S0925-2312(01)00658-0

Che, Z., Purushotham, S., Khemani, R., & Lin, Y. (2016). Interpretable deep models for icu outcome prediction. In *AMIA Annual Symposium Proceedings* (p. 371). American Medical Informatics Association volume 2016.

Dhoble, K., Nuntalid, N., Indiveri, G., & Kasabov, N. (2012). Online spatio-temporal pattern recognition with evolving spiking neural networks utilising address event representation, rank order, and temporal spike learning. In *The 2012 International Joint Conference on Neural Networks (IJCNN)* (pp. 1–7). doi:10.1109/IJCNN.2012.6252439

Dora, S., Subramanian, K., Suresh, S., & Sundararajan, N. (2016). Development of a Self-Regulating Evolving Spiking Neural Network for classification problem. *Neurocomputing*, 171, 1216–1229. doi:10.1016/j.neucom.2015.07.088

Dora, S., Sundaram, S., & Sundararajan, N. (2018). An interclass margin maximization learning algorithm for evolving spiking neural network. *IEEE Transactions on Cybernetics*, (pp. 1–11). doi:10.1109/TCYB.2018.2791282

Frosst, N., & Hinton, G. (2017). Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784*, .

15
Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*,.

Hussain, S., Liu, S.-C., & Basu, A. (2014). Improved margin multi-class classification using dendritic neurons with morphological learning. In *Circuits and Systems (ISCAS), 2014 IEEE International Symposium on* (pp. 2640–2643). IEEE.

Jeyasothy, A., Sundaram, S., & Sundararajan, N. (2018). Seltron: A new spiking neuron model with time-varying synaptic efficacy function for pattern classification. *IEEE Transactions on Neural Networks and Learning Systems*, (pp. 1–10). doi:10.1109/TNNLS.2018.2868874

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86, 2272–2324.

Lee, J. H., Delbruck, T., & Pfeiffer, M. (2016). Training deep spiking neural networks using backpropagation. *Frontiers in Neuroscience*, 10, 508. doi:10.3389/fnins.2016.00508

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems* (pp. 4765–4774).

Markram, H., Gerstner, W., & Sjöström, P. J. (2012). Spike-Timing-Dependent Plasticity: A Comprehensive Overview. *Frontiers Media SA*. doi:10.3389/978-2-88919-043-0

Mostafa, H. (2018). Supervised learning based on temporal coding in spiking neural networks. *IEEE transactions on neural networks and learning systems*, 29, 3227–3245.

Neftci, E., Das, S., Pedroni, B., Kreutz-Delgado, K., & Caudehonghe, G. (2014). Event-driven contrastive divergence for spiking neuromorphic systems. *Frontiers in neuroscience*, 7, 272.

O'Connor, P., Neil, D., Liu, S.-C., Delbruck, T., & Pfeiffer, M. (2013). Real-time classification and sensor fusion with a spiking deep belief network. *Frontiers in neuroscience*, 7, 178.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). “why should i trust you?”: Explaining the predictions of any classifier. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* KDD ’16 (pp. 1135–1144). New York, NY, USA: ACM. URL: http://doi.acm.org/10.1145/2939672.2939778

Samek, W., Binder, A., Montavon, G., Lapuschkin, S., & Müller, K.-R. (2017). Evaluating the visualization of what a deep neural network has learned. *IEEE transactions on neural networks and learning systems*, 28, 2660–2673.

Shrikumar, A., Greenside, P., & Kundaje, A. (2017). Learning important features through propagating activation differences. In *International Conference on Machine Learning* (pp. 3145–3153).

Simonyan, K., Vedaldi, A., & Zisserman, A. (2013). Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*,.

Tan, S., Caruana, R., Hooker, G., & Gordo, A. (2018). Transparent model distillation. *arXiv preprint arXiv:1801.08640*,.

Tavanaei, A., Ghodrat, M., Kheradpisheh, S, R., Masquelier, T., & Maida, A. (2019). Deep learning in spiking neural networks. *Neural Networks, 111*, 47 – 63. doi:https://doi.org/10.1016/j.neunet.2018.12.002

Tavanaei, A., & Maida, A. (2019). Bp-stdp: Approximating backpropagation using spike timing dependent plasticity. *Neurocomputing*, 330, 39 – 47. doi:https://doi.org/10.1016/j.neucom.2018.11.014

Tavanaei, A., Zhou, A., & Maida, A. S. (2018). Training spiking convnets by stdp and gradient descent. *2018 International Joint Conference on Neural Networks (IJCNN)*, (pp. 1–8).

Wade, J. J., Medaïd, L. J., Santos, J. A., & Sayers, H. M. (2010). SWAT : A Spiking Neural Network Training Algorithm for Classification Problems. *IEEE transactions on neural networks*, 21, 1817–1830.

Wang, J., Belatreche, A., Maguire, L. P., & Megnimit, T. M. (2017). SpikeTemp : An Enhanced Rank-Order-Based Learning Approach for Spiking Neural Networks With Adaptive Structure. *IEEE transactions on neural networks and learning systems*, 28, 30–43.

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision* (pp. 818–833). Springer.

Zhao, B., Ding, R., Chen, S., Linares-Barranco, B., & Tang, H. (2015). Feedforward categorization on aer motion events using cortex-like features in a spiking neural network. *IEEE transactions on neural networks and learning systems*, 26, 1963–1978.