A gradient descent algorithm for SNN with time-varying weights for Reliable Multiclass Interpretation

Abeegithan Jeyasothy[a,c,*], Savitha Ramasamy[c], Suresh Sundaram[b]

[a]*School of Computer Science and Engineering, Nanyang Technological University, Singapore*
[b]*Department of Aerospace Engineering, Indian Institute of Science, Bangalore, India*
[c]*Institute for Infocomm Research, Agency for Science, Technology and Research, Singapore*

Abstract

In this paper, we develop a modified gradient descent learning algorithm for a spiking neural network with time-varying weights (SNN-t) to solve multi-class classification problems. We also explore the interpretability of the SNN-t, based on the Generative Additive Model framework, referred to as Spiking Additive Model (SAM). Although GAMs have been extensively explored for binary classification problems, the presence of multiple shape functions severely affects the ability of GAM to provide interpretations for multi-class classification problems. Therefore, we propose a post-processing method to enable better visualization of multiple shape functions, towards better *relative interpretation* for multiclass classification problems. We first train the SNN-t using the proposed modified gradient descent learning algorithm. The time-varying weights in a single layer SNN-t is functionally similar to that of shape functions in GAMs for classification tasks. Hence, we transform the time-varying weights in the temporal domain of SNN-t, to shape functions in the feature space of GAMs. The resultant Spiking Additive Model (SAM) has the generalization ability of SNN-t and the interpretable characteristics of GAM for multiclass problems. As the modified gradient descent based learning algorithm of SNN-t provides better generalization ability, the interpretations provided by the SAM is more reliable. We evaluate the performance of SAM on large public datasets. The proposed SAM has better generalization accuracy than other state-of-the-art multi-class GAM, while preserving the classification accuracy of SNN-t. Then, we evaluate the interpretation ability of the proposed SAM. We estimate a reference shape function for each feature to provide a relative interpretation. In a relative interpretation, the importance of the feature in multiclass classification is established. It is observed that relative interpretation of SAM is more reliable, as the classification is more accurate.

Keywords: Spiking neural network, multiclass classification, time-varying weight model, Generalized additive model, model interpretation, temporal encoding

1. Introduction

Interpretation of a model's decision is given based on the importance of the feature that influences the decision. White-box models are inherently interpretable as the decision-making mechanism is obvious and the importance of the feature can be measured directly from the model. On the other hand, importance of the feature can be measured through some post-hoc methods [1, 2, 3, 4, 5] to interpret the black-box model predictions, with superior performance. However, these post-hoc methods do not provide the overall behavior of the base model [6]. However, white-box models are the popular choice for high-stake decision-making, as the knowledge of decision-making mechanisms in a model is essential for applications in various domains, including criminal justice, health, finance, etc.

*Corresponding author
Email address: abeegith1@e.ntu.edu.sg (Abeegithan Jeyasothy)

In order to trust the interpretations of the predictions, the machine learning model should be able to generalize well, with high accuracies. To this end, it is important to develop a interpretable model with high classification accuracy towards improving reliability. In [7], the SNN with time-varying weight (Mc-SEFRON) is trained using Spike-timing-dependent plasticity (STDP) rule [8]. Although this MC-SEFRON is accurate, we also develop a modified gradient descent based learning algorithm to improve generalization accuracy of the SNN-t to ensure improved interpretability.

We leverage on an inherently interpretable Generative Additive Model (GAM), which is a white-box model, to explore the interpretability of the SNN-t trained with mofified gradient descent based learning algorith. A standard GAM can be written as,

$$g(\mathbb{E}[y]) = \beta_0 + \sum_i f_i(x_i) \tag{1}$$

where $g(.)$ is the link function, $f_i(.)$ is a shape function with $\mathbb{E}[f_i] = 0$, x is the input and i is the index of the input feature. The shape function f_i in a GAM captures the importance of the i^{th} feature alone, and it can be visualized as a graph. GAMs are predominantly used in binary classification or regression problems, and interpreting these models is possible because the importance of input feature is independent of other features. Furthermore, the values in a shape function of GAMs are directly proportional to, and hence, directly attributable to the likelihood of prediction in a binary classification problem. Owing to their inherent interpretability, GAMs are preferred candidates for several applications such as healthcare [9], environmental studies [10], finance [11] etc.

Recently, a direct weight transformation method to obtain multiclass GAM from a spiking neural network (SNN) with time-varying weight is proposed in [7]. Functional similarity between the time-varying weights [12] in SNN and the shape functions in GAM for classification task is exploited to perform a lossless weight transformation. The advantage of the weight transformation method is that the learn-able parameters in the base SNN classifier are directly transformed to shape functions. Hence the different feature interactions do not have to be modelled separately, as all the features are jointly trained to minimize an error function by the base SNN classifier. It is to be noted that [13, 7] are the only two works in the literature on multiclass GAMs.

In this paper, we propose a Spiking Additive Model (SAM), and a post-processing method to enhance the visualization of shape functions to derive *relative interpretation* in a multiclass setting. First, the Spiking Neural Network with time varying weights (SNN-t) is trained through a modified gradient descent based learning algorithm [14]. To the best of our knowledge, this work is the first to modify gradient descent based learning algorithm to train time-varying weights for spiking neural networks. Next, we transform the time-varying weights in temporal domain of the trained SNN-t, to shape functions in feature space of GAMs. This results in a SAM combining the classification accuracy of SNN-t and the interpretation ability of GAMs for multiclass problems. Then, we post-process the shape functions in SAM to provide a relative interpretation for each feature, without any loss in the performance accuracy. This post-processing modifies the shape functions such that only one class per feature value has a positive shape value, while the other classes are either zero or negative. Thus, the relative interpretation emphasizes the positive shape values to establish the importance of the features in multiclass classification. We compare performance of SAM with the other multiclass GAMs and other multiclass classifiers on real-world datasets. Our experimental results show that SAM has overall $0.1\% - 11\%$ improved performance compared to other multiclass GAMs. Also, even for very complex shape functions, the interpretation obtained from the relative interpretation method is more visually explainable, compared to those obtained from axiomatic interpretation method. Our main contributions in this paper can be summarized as follow:

- We train SNN-t using modified gradient decent algorithm, to improve its generalization ability towards reliable interpretation.
- SAM is a low resource, low computational interpretation method. Current methods require millions of decision trees or hundreds of neural networks to fit shape function, while SAM uses only one trained single layer SNN-t.

- We develop a post-processing method for SAM to provide relative interpretation, which is more visually explainable for multiclass classification.
- Our source code utilizes current deep learning frameworks and it can be trained on GPU. High-dimensional and big data can be easily trained. link for source code:

The rest of the paper is organized as follows: Section 3 presents the training of SNN-t. Generating multiclass SAM from SNN-t and its post-processing method to provide relative interpretation is presented in Section 4. Section 5 presents the performance of SAM compared with other models on public datasets. Section 6 presents the use case of relative interpretability in SAM. Finally, in section 7, we summarise the conclusions in our paper.

2. Related Works

Initially the shape functions in GAMs are fitted using splines [15]. Traditionally, there have been many methods to fit a shape function. It has been found in [16] that boosting methods outperform backfitting, smoothing function optimization and mixed model approach, especially in high dimensional data. In [17] shape functions have been fitted using bagged trees with boosting methods that cycles through the input feature one by one. Generally, the GAMs obtained with gradient boosted bagged trees are called the Explainable Boosting Machine (EBM). In [18], EBM has been developed with pairwise interaction between the features, where the interaction of the features is captured in two dimensional space as heatmap of the two features. In [11], subbagged trees with boosting have been used to speed-up the training process in EBM. Another recently emerging method is to fit a shape function by using neural networks. In [19, 20, 21], multi-layer neural network operates on a single feature and trained jointly using backpropagation. Additionally in [20, 21], pairwise interaction of the features have also been modelled using neural network. All the above mentioned shape function fitting methods are developed for binary classification or regression problems. However, developments in multiclass GAMs are still in its infancy, as interpreting shape functions in multiclass GAMs are not straight forward. In a multiclass setting, there are multiple shape functions to correlate each feature to each class. This inhibits interpreting multiclass GAMs. Following the same line of works in EBM, bagged trees with boosting method are used to fit multiclass EBM in [13]. Then the multiclass EBMs are post-processed to modify the shape functions to derive axiomatic interpretations, as this is obtained by satisfying two axioms of smoothness and monotonicity. This axiomatic interpretation does not exploit the complete advantage of individual shape functions to derive importance of each feature for individual classes.

3. Spiking Neural Network with time-varying weights (SNN-t)

We consider multi-class classification problem in real-valued domain. Let $\mathbb{X} \in \mathbb{R}^m$ be the inputs with m real-valued features, and its output be $\mathbb{Y} = \{1, 2, ..., n\}$, where n is the total number of output classes and the set $\{1, 2, ..., n\}$ is denoted as $[n]$. Without loss of generality, any set of continuous numbers will be denoted in similar manner hereafter. Real-valued inputs have to be encoded into spike patterns to work with SNNs. In SNN, inputs and outputs are in temporal domain and are called spikes (discrete temporal event). We use a nonleaky integrate and fire neuron with time-varying weight model [12] in our SNN-t classifier. The individual weight for each input spike is sampled from the time-varying weight function. The postsynaptic potential of the output neuron is obtained through the sum of the weighted spike response for all the input spikes. When this postsynaptic potential of an output neuron exceeds a user-defined threshold, the neuron fires a spike. Thus, the class label for an input is predicted based on the output neuron that produces the earliest spike.

3.1. Converting the real-valued input into spike pattern

In the SNN-t, we use the population encoding scheme [22] to convert the real-valued input into input spike pattern. Population encoding scheme mimics the functioning of the biological receptive field neurons. In population encoding scheme, the input data is projected into higher dimensional space, $\mathbb{R}^{m\times q}$ using multiple Receptive Field (RF) neurons, where q is the number of RF neurons. RF neurons are represented by some closed functions (Eg: Gaussian, cosine square and etc).

Let x_i^k denote the i^{th} feature of the k^{th} sample $\mathbf{x}^k; k \in N$, where N is the number of samples in the data set. For simplicity, we drop the superscript k hereafter. In population encoding scheme, let ϕ_i^r denote the firing strength produced by the r^{th} RF neuron for x_i. The spike pattern s_i^r is then determined by $s_i^r = T.(1-\phi_i^r)$, where T in the input spike time interval.

Let $P(.)$ denote a generic population encoding scheme and an encoded spike pattern for x_i is obtained by,

$$P(x_i) = \mathbf{s}_i = \{s_i^1, s_i^2 ....s_i^q\} \tag{2}$$

Population encoding scheme is invertible and has a unique solution. The inverse population encoding scheme is given by,

$$P^{-1}(s_i) = \{x_i | P(x_i) = s_i\} \tag{3}$$

Next, we present the supervised learning algorithm of a single-layer SNN-t.

3.2. Supervised Learning in SNN-t

Let $v_j(t)$ and θ_j denote the postsynaptic potential and the threshold of the j^{th} ($j \in [n]$) output neuron, respectively. An output spike $\hat{s}_j$ will be fired by the j^{th} output neuron when $v_j(t)$ crosses θ_j. Postsynaptic potential $v_j(t)$ for $t > 0$ is given as,

$$v_j(t) = \sum_{i\in[m]} \sum_{r\in[q]} w_{ij}^r(s_i^r).(1 - exp\bigg(- \frac{(t-s_i^r)}{\tau}\bigg)).H(t-s_i^r) \tag{4}$$

Here $w_{ij}^r(s_i^r)$ is sampled from the time-varying weight $w_{ij}^r(t)$, τ is the time constant and H(t) is the Heaviside step function. The objective of the SNN-t is to learn the time-varying weights $w_{ij}^r(t)$, given the input X and the targets Y.

Assume that only a subset of the input spikes have made the $v_j(t)$ cross the θ_j. Let $C_m \subseteq [m]$ and $C_q \subseteq [q]$ are the subset of the indices for input feature and RF neuron respectively. From eq 4, the firing time of the output neuron $\hat{s}_j$ is implicitly defined as,

$$\theta_j = \sum_{i\in[C_m]} \sum_{r\in[C_q]} w_{ij}^r(s_i^r).(1 - exp\bigg(- \frac{(\hat{s}_j-s_i^r)}{\tau}\bigg)).H(\hat{s}_j-s_i^r) \tag{5}$$

$H(\hat{s}_j - s_i^r)$ is always positive, as $\hat{s}_j > s_i^r$ for $\{i \in [C_m]$ & $r \in [C_q]\}$.

Input and output spike times are always in the exponential terms. Therefore, we can re-write Eq. (5) to show the direct relationships between input and output spikes, as below:

$$exp\bigg(\frac{\hat{s}_j}{\tau}\bigg) = \frac{\sum_{i\in[C_m]}\sum_{r\in[C_q]} w_{ij}^r(s_i^r).exp\bigg(\frac{s_i^r}{\tau}\bigg)}{\sum_{i\in[C_m]}\sum_{r\in[C_q]} w_{ij}^r(s_i^r) - \theta_j} \tag{6}$$

Replacing the $exp(\frac{s}{\tau})$ with z in Eq. (6) for ease of gradient descent estimation,

$$\hat{z}_j = \frac{\sum_{i\in[C_m]}\sum_{r\in[C_q]} w_{ij}^r(s_i^r).z_i^r}{\sum_{i\in[C_m]}\sum_{r\in[C_q]} w_{ij}^r(s_i^r) - \theta_j} \tag{7}$$

From eq 7, it can be deduced that the subsets of input spikes that fires the output spikes should satisfy the following three conditions:

- Condition 1: $\sum_{i\in[C_m]}\sum_{r\in[C_q]} w_{ij}^r(s_i^r) > \theta_j$
  Inorder to fire an output spike, its $\hat{z}_j$ should be positive. Failure to satisfy the condition will result in a negative value for $\hat{z}_j$. As negative exponential value is non-existent, the output neuron does not fire.

- Condition 2: $\hat{z}_j$ ¿ z_i^r, for $i \in [C_m] \wedge\ r \in [C_q]$
  This condition ensures that the output neuron should be fired, only after the subset of input spikes are fired.

- Condition 3: $\hat{z}_j$ ¡ z_i^r, for $(i \notin [C_m]\ \wedge\ i \in [m]) \wedge (r \notin [C_q]\ \wedge\ r \in [q])$
  This condition ensures that the input spikes that are not in the subset are fired only after the output spike.

The algorithm to fire an output spike in the SNN-t classifier is described in Algorithm 1.

Algorithm 1: Algorithm to fire an output spike of SNN-t

Input: **z**-Vector of input spike time of length $m \times q$
Input: **w**-Weight vector sampled from time-varying weights
Output: $\hat{\mathbf{z}}$-Output spike time

sort_indices $\leftarrow$ *argsort*(**z**) `// Ascending order`
$[\mathbf{C}_m] = [], [\mathbf{C}_q] = []$ `// Empty subset of indices`
for *sort_index* ***in*** *sort_indices* **do**
  $[\mathbf{C}_m] = [\mathbf{C}_m] + feature(sort\_index)$ `// Add feature index of the sort_index to the subset`
  $[\mathbf{C}_q] = [\mathbf{C}_q] + RF\_neuron(sort\_index)$ `// Add RF_neuron index of the sort_index to the subset`
  if *Condition 1* $\wedge$ *Condition 2* $\wedge$ *Condition 3* **then**
    return $\frac{\sum_{i\in[C_m]}\sum_{r\in[C_q]} w_{ij}^r(s_i^r).z_i^r}{\sum_{i\in[C_m]}\sum_{r\in[C_q]} w_{ij}^r(s_i^r)-\theta_j}$
return $\emptyset$

In SNN-t classifier, the class label for a given input is determined by the output neurons that fires a spike earlier than the other output neurons. Let $\hat{y}$ denote the predicted class label for input $\mathbf{x}$ and it is predicted using the following classification rule

$$\hat{y} = \underset{j}{\operatorname{argmin}}\, \hat{z}_j \tag{8}$$

The goal of the training procedure is to update the network parameters ($w_{ij}^r(t)$) such that the output neuron corresponding to the correct class label fires before other output neurons. This is achieved by maximizing the negative values of the first output spike time through minimizing the cross entropy loss function. For the set of first output spike time in exponentiated form and the correct class (target class) index g, the cross entropy loss function is defined as,

$$L(g, -\hat{\mathbf{z}}) = -ln\left(\frac{exp(-\hat{z}_g)}{\sum_{j\in[n]} exp(-\hat{z}_j)}\right) \tag{9}$$

From eq 7, the derivative of an output spike time with respect to the weight and the input spike time can be derived and given by,

$$\frac{d\hat{z}_j}{dw_{ij}^r(s_i^r)} = \begin{cases} \frac{z_i^r - \hat{z}_j}{\sum_{i\in[C_m]}\sum_{r\in[C_q]} w_{ij}^r(s_i^r)-\theta_j} & \text{if } i \in [C_m] \wedge r \in [C_q] \\ o & \text{otherwise} \end{cases} \tag{10}$$

$$\frac{d\hat{z}_j}{dz_i^r} = \begin{cases} \frac{w_{ij}^r(s_i^r)}{\sum_{i\in[C_m]}\sum_{r\in[C_q]} w_{ij}^r(s_i^r)-\theta_j} & \text{if } i \in [C_m] \wedge r \in [C_q] \\ o & \text{otherwise} \end{cases} \tag{11}$$

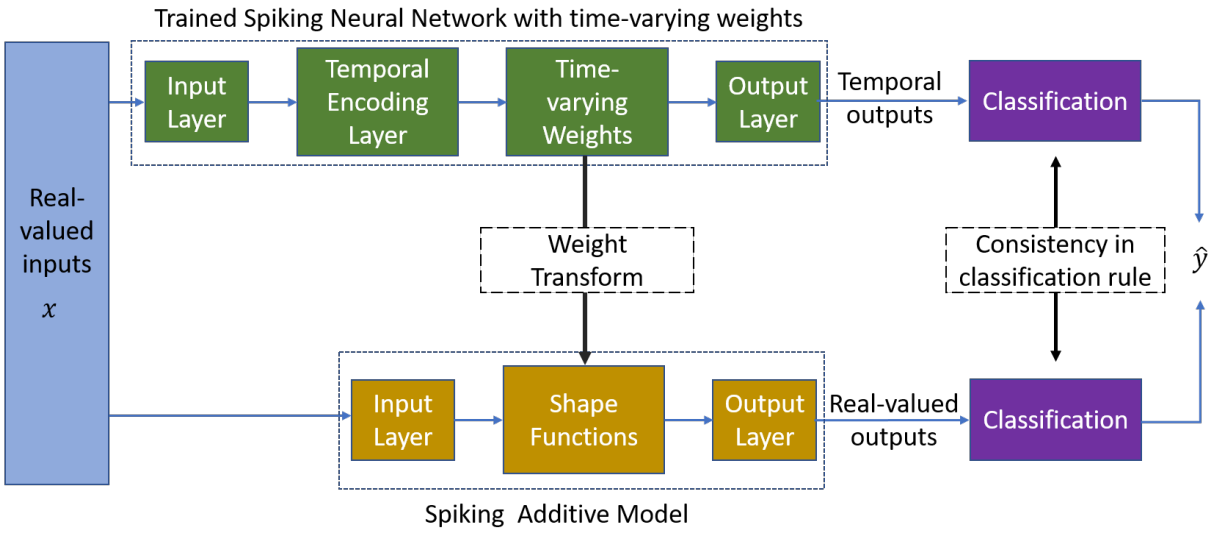

Figure 1: Development of a Spiking Additive Model

Eq 10 and 11 are used in backpropagation pipeline to train our SNN-t classifier. Eq 11 is only needed if multi-layer architecture is used. A single layer SNN with time-varying weights is functionally similar to that of GAMs. Hence we have only used a single layer. In backpropagation, the derivative of error with respect to the weight will be a single value. It is embedded in a Gaussian function to generate the time-varying weight update.

$$w_{ij}^r(t)_{new} = w_{ij}^r(t)_{old} - \frac{\partial L(j, -\hat{\mathbf{z}})}{\partial w_{ij}^r(s_i^r)}.exp\left(\frac{-(t-s_i^r)^2}{2\sigma^2}\right) \tag{12}$$

Here σ is the weight update range. It determines the range of temporal neighbours that will be affected by the weight update. A higher value of σ result in a constant weight function, which is equivalent to single weight model.

4. Development of SAM and its Interpretation

As mentioned earlier, the single layer SNN-t and GAM have very similar functions for classification tasks as shown in Fig 1. It must be noted that this unique similarity only holds if the weights in an SNN are time-varying functions [7]. First we present the weight transformation method to obtain shape functions from time-varying weights. We then present the post-processing method for those shape functions to derive relative interpretations for the SAM.

4.1. Transformation of the Time-varying Weights to Shape Functions

Let $F_j(x)$ be an additive function of j^{th} output class in a multiclass GAM, where $F_j(x) = \sum_{i\in[m]} f_{ij}(x_i)$. Here $f_{ij}(.)$ is the shape function of the i^{th} input feature for the j^{th} output class. Then, the classification rule of the multiclass GAM is given by,

$$\hat{y}_{GAM} = \underset{j}{\text{argmax}}\, F_j(x) \tag{13}$$

$$= \underset{j}{\text{argmax}} \sum_{i\in[m]} f_{ij}(x_i) \tag{14}$$

Here $\hat{y}_{GAM}$ is the predicted class label of the GAM. Our goal in this section is to show that the SNN-t classification function (eq 8) can be transformed to GAM classification function (eq 14). Hence, this will allow us to transform the temporal elements of SNN-t to feature space of SAM, to obtain the shape functions.

We start with eq 4 and eq 5. It is known that the $v_j(t)$ should cross the threshold θ_j to produce an output $\hat{s}_j$. The firing condition is written as,

$$\hat{s}_j = min\ \{t | v_j(t) > \theta_j\} \tag{15}$$

Using eq 15, classification in eq 8 is re-written implicitly with output spike time as,

$$\hat{y} = \underset{j}{\text{argmin}}\, min\ \{t | \frac{1}{\theta_j} v_j(t) > 1\} \tag{16}$$

Let $\hat{s}_o$ be an arbitrary earliest output spike time among the output spike times of all the output neuron ($\hat{s}_o \simeq min\ \hat{s}_j$). Then the term $\frac{1}{\theta_j} v_j(t)$ is maximum at $t = \hat{s}_o$ for the output neuron that fires the output spike at $\hat{s}_o$. The Predicted class label $\hat{y}$ of the SNN-t is approximated by using $\hat{s}_o$, and is written as,

$$\hat{y}^* = \underset{j}{\text{argmax}} \frac{1}{\theta_j} v_j(\hat{s}_o) \tag{17}$$

Replacing $v_j(t)$ with RHS of eq 4 and s_i^r with $P(x_i)^r$ gives a classification rule in feature space as,

$$\hat{y}^* = \underset{j}{\text{argmax}} \frac{1}{\theta_j} \sum_{i \in [m]} \sum_{r \in [q]} w_{ij}^r (P(x_i)^r).(1 - exp\bigg(- \frac{(\hat{s}_o - P(x_i)^r)}{\tau} \bigg)).H(\hat{s}_o - P(x_i)^r) \tag{18}$$

We obtain the shape function of i^{th} input feature for the j^{th} output class from eq 18 as,

$$f_{ij}(x_i) = \frac{1}{\theta_j} \sum_{r \in [q]} w_{ij}^r (P(x_i)^r).(1 - exp\bigg(- \frac{(\hat{s}_o - P(x_i)^r)}{\tau} \bigg)).H(\hat{s}_o - P(x_i)^r) \tag{19}$$

Replacing RHS of eq 18 with f_{ij} gives,

$$\hat{y}^* = \underset{j}{\text{argmax}} \sum_{i \in [m]} f_{ij}(x_i) \tag{20}$$

This classification rule is equivalent to the classification rule of multiclass GAM in eq 14. The shape functions are obtained by tuning $\hat{s}_o$ to minimize the overall difference between $\hat{y}$ and $\hat{y}^*$ for the training dataset.

4.2. Relative Interpretability of multiclass SAM

In a multiclass setting, there are multiple shape functions to correlate each feature to each class of the GAM. This poses severe challenges in deriving the interpretations of multiclass GAMs. However, we can perform re-scaling or addition of feature functions on multiclass GAM to provide a better visualization, while retaining the same accuracy. Multiclass GAMs have the following properties for classification problem that we exploit to obtain relative interpretation.

Property 1:Feature wise operation on multiclass GAM is independent of classification performance

Proposition 1: For a given input, let the predicted class labels of GAM^1 and GAM^2 be $\hat{y}^1$ and $\hat{y}^2$, respectively, and be defined as,

$$\hat{y}^1 = \underset{j}{\text{argmax}} \sum_{i \in [m]} f_{ij}(x_i)$$

$$\hat{y}^2 = \underset{j}{\text{argmax}} \sum_{i \in [m]} f_{ij}(x_i) + g_i(x_i)$$

Then for any arbitrary function $g_i(.)$ the predicted class labels $\hat{y}^1$ and $\hat{y}^2$ are equal.

Proof 1: Let $\mathbf{a} = [a_1, a_2, a_3]$ a set with $a_2 > a_1 > a_3$. Then $\text{argmax}_k\, a_k = 2$. Due to the transitive property adding an arbitrary constant will not change the ranking of the elements in the set. i.e. If $a* = a+\varepsilon$ then $a_2 + \varepsilon > a_1 + \varepsilon > a_3 + \varepsilon$ is true $\forall \varepsilon \in \mathbb{R}$. This implies $\text{argmax}_k\, a^*_k = 2$.

We use this transitive property to show $\hat{y}^1 = \hat{y}^2$. The $\sum_{i\in[m]} g_i(x_i)$ is a constant and it is added to all the classes. Due to the transitive property, the ranking of $\sum_{i\in[m]} f_{ij}(x_i)$ will not be changed by the addition of $\sum_{i\in[m]} g_i(x_i)$. Hence the predicted class labels will be same.

Property 2: Scaling of multiclass GAM is independent of classification performance

Proposition 2: For a given input, let the predicted class labels of GAM^1 and $\text{GAM}^1 \times C$ be $\hat{y}^1$ and $\hat{y}^2$, respectively, and be defined as,

$$\hat{y}^1 = \underset{j}{\text{argmax}} \sum_{i\in[m]} f_{ij}(x_i)$$

$$\hat{y}^2 = \underset{j}{\text{argmax}} \sum_{i\in[m]} f_{ij}(x_i) \times C$$

Then for any $C > 0$ the predicted class labels $\hat{y}^1$ and $\hat{y}^2$ are equal.

Proof 2: Let $\mathbf{a} = [a_1, a_2, a_3]$ a set with $a_2 > a_1 > a_3$. Then $\text{argmax}_k\, a_k = 2$. Similar to the proof 1, due to the transitive property, if $a* = a \times \varepsilon$ then $a_2 \times \varepsilon > a_1 \times \varepsilon > a_3 \times \varepsilon$ is true $\forall\ \varepsilon \in \mathbb{R}^+$. Hence $\hat{y}^1 = \hat{y}^2$, $\forall\ C \in \mathbb{R}^+$.

Finally, We leverage on both the properties of multiclass GAM to post-process the shape functions to provide a relative interpretation for each feature, without any loss in the performance accuracy, through the following steps:

- Step 1: Re-scale all the shape values of the SAM in [0,1] using the min-max scaler.
- Step 2: Determine a reference feature function for each input feature using the second highest shape values for each feature.
- Step 3: Shift the reference feature function to zero shape value for each feature.

.

These post-processing steps ensures that only one class per feature has a positive shape value, while the other classes are either zero or negative. Thus, the relative interpretation emphasizes the positive shape values to establish the importance of the features in multiclass classification. The 3 step post-processing method for interpretation of SAM is summarized in Algorithm 2.

Algorithm 2: Post-processing of multiclass SAM to provide relative interpretation

Input: SAM with $\mathbf{G} = \{f_{ij}(.)\}$
Output: $\mathbf{G}'$ that enables relative interpretation

$\mathbf{G} = \frac{\mathbf{G} - min(\mathbf{G})}{max(\mathbf{G}) - min(\mathbf{G})}$ `// Resacling`
for *i* ***in*** *[m]* **do**
  sort_indices $\leftarrow$ *argsort*$(f_{ij}(.))$ `// Ascending order`
  $K =$ **sort_indices**$[2, :]$ `// Select class label` 2^{nd} `highest shape value`
  $g_i(.) = f_{iK}(.)$ `// Generate the reference feature function`
return $\mathbf{G}' = \{f_{ij}(.) - g_i(.)\}$ `// Shift the reference feature function to 0`

5. Experiments

In this section, we evaluate the performance of SAM with six baselines on four multiclass classification datasets from UCI repository [23]. GAMs provide individual feature wise explanation. Hence tabular datasets are selected with various class imbalance factor. Imbalance factor (I.F) is determined by, I.F=

$1 - \frac{n}{C_n} \min_j C_j$, where, n and C_n are the total number of classes and total number of samples respectively. C_j is the number of samples in class j. I.F = 0 and I.F = 1 indicates perfectly balanced and perfectly unbalanced dataset, respectively. Description of the dataset is given in Table 1.

Table 1: Description of Dataset

Dataset	Classes	Features	I.F	Size
Shuttle	7	9	0.998	58,000
Covertype	7	12	0.967	581,012
Diabetes	3	39	0.661	77,975
Sensorless	11	48	0.000	58,509

5.1. Baselines

We compare the performance of SAM with the following baselines:

- **Logistic Regression (LR)**: This is a simple interpretable classifier. Comparison with LR highlights the improvement of SAM in learning the non-linear relationships between the input features and the output class.
- **Multiclass Explainable Boosting Machines (Mc-EBM)**: EBMs are the widely used GAMs. We have used the multiclass implementation of EBMs [13]. Mc-EBM is a multiclass GAMs, obtained through fitting of shape functions. Comparison of SAM with Mc-EBM helps to elicit the advantages of obtaining shape functions through weight transformations.
- **Mc-Sefron (GAM)**: This is the multiclass GAM obtained by transforming the time-varying weights in Mc-Sefron [7]. Comparison with Mc-Sefron (GAM) indicates the improvement in performance due to the backpropagation learning algorithm to train SNN-t.
- **Mc-Sefron**: This is the SNN classifier with time-varying weights in [7]. Mc-Sefron is trained using STDP learning rule. Comparison with Mc-Sefron indicates the improvement in performance due to the backpropagation learning algorithm to train SNN-t.
- **SNN with time-varying weight (SNN-t)**: This is the SNN-t classifier trained using the modified gradient decent algorithm (Section 3). Comparison with SNN-t helps to estimate the loss incurred during weight transformation from SNN-t to SAM.
- **Gradient Boosted Trees (GBT)**: GBT is a non-interpretable classifier. GBT has high accuracies for tabular datasets and is a benchmark for upperbound performance. Comparison with GBT helps to assess the tradeoff between interpretability and accuracy of classification in SAM.

5.2. Experimental design

We split the dataset into train, validation and test splits using 80%, 10% and 10% of the dataset respectively. We repeat the experiment five times and report mean balanced accuracy (μ_A) and standard deviation of the test splits. μ_A is determined as, $\mu_A = \frac{1}{n} \sum_{j \in [n]} \frac{q_j}{C_j}$. Here, q_j is the number of correctly classified j^{th} class samples. Table 2 shows the performance comparison of SAM with other baseline classifiers. For ease of reproduction, we have summarized the experimental setting in the Appendix 1.

5.3. Results and Discussion

Table 2 show the results on multiclass classification. We make a few observations from the results in the table.

1. Performance of SAM is significantly higher (on average 23.9%) than the performance of logistic regression. This indicates the non-linear relationship between the input features and the output is captured well by SAM.

Table 2: Balanced Accuracy on Test set

Model	Shuttle	Covertype	Diabetes	Sensorless
GBT	0.997+ 0.008	0.938+0.003	0.447+0.004	0.999+0.000
SAM*	0.988+0.006	0.714+0.003	0.454+0.004	0.993+0.001
SNN-t*	0.992+0.006	0.711+0.003	0.449+0.005	0.993+0.000
Mc-Sefron	0.992+0.004	0.604+0.029	0.434+0.004	0.991+0.000
Mc-Sefron(GAM)	0.985+0.003	0.601+0.029	0.429+0.008	0.992+0.000
Mc-EBM	0.972+0.031	0.538+0.003	0.428+0.004	0.997+0.001
LR	0.617+0.060	0.356+0.004	0.387+0.002	0.832+0.006

* Our work

2. In comparison to Mc-EBM, SAM has improved performance of $1.6 - 17.6\%$ on three out of the four datasets. Drop in performance in sensorless dataset is only 0.5%.
3. In comparison to Mc-Sefron(GAM), SAM has improved performance of $0.1 - 11.3\%$ on all of the four datasets. Also in comparison to Mc-Sefron, SNN-t has improved performance of $0.2 - 10.7\%$ on three out of the four datasets (For shuttle dataset both Mc-Sefron and SNN-t has same accuracy). This indicates that the gradient descent based learning algorithm improves the performance of spiking neural network with time-varying weights compared to STDP based learning algorithm.
4. On covertype dataset, SAM has significantly higher than the performance of Mc-EBM and Mc-Sefron (around 17% and 11% higher). This could be attributed to the generalization ability of SNN-t on class imbalance datasets. However, further investigation is required to make a definite conclusion.
5. On average difference between performance of SAM and SNN-t is only 0.3%.This indicates that the weight transformation method to generate SAM from SNN-t is very efficient and the performance of SNN-t is preserved in SAM. Similarly the average difference between Mc-Sefron and Mc-Sefron(GAM) is only 0.35%. This indicates the weight transformation method is independent of the learning algorithm used to train the SNN with time-varying weights.
6. As expected, Gradient Boosted Trees performed well on three out of the four datasets, except on Diabetes dataset. SAM performs slightly better than Gradient Boosted Trees on Diabetes dataset. Even though the improvement is accuracy is only 0.7%, it comes with an additional biggest advantage of interpretabilty.

In conclusion, SAM has better performance than Mc-EBM and Mc-Sefron(GAM). The performance of SNN-t is improved by the gradient descent learning algorithm. The weight transformation method retains the generalization ability of SNN-t. Improving the performance of SNN-t will improve the performance of SAM generated from them.

6. Interpretation of SAM for Real-world Data

In this section, we demonstrate the interpretation ability of SAM using two public data sets from Table 1. As the accuracy of classification is of paramount importance to derive robust interpretations, we choose the Sensorless and Shuttle dataset for this demonstration, as the SAM classifier classifies these dat sets with high accuracy ($> 98\%$). It must also be noted that the Sensorless data set has 48 features, which is the highest number of features among the data sets used in the study. On the other hand, the Shuttle data set has the lowest number of features (9) among the data sets used in the study. Thus, our interpretation study is conducted such that the sensorless data set with many features is used for interpretation through individual features. On the other hand, the shuttle dataset with fewer features is used to demonstrate interpretation of individual classes.

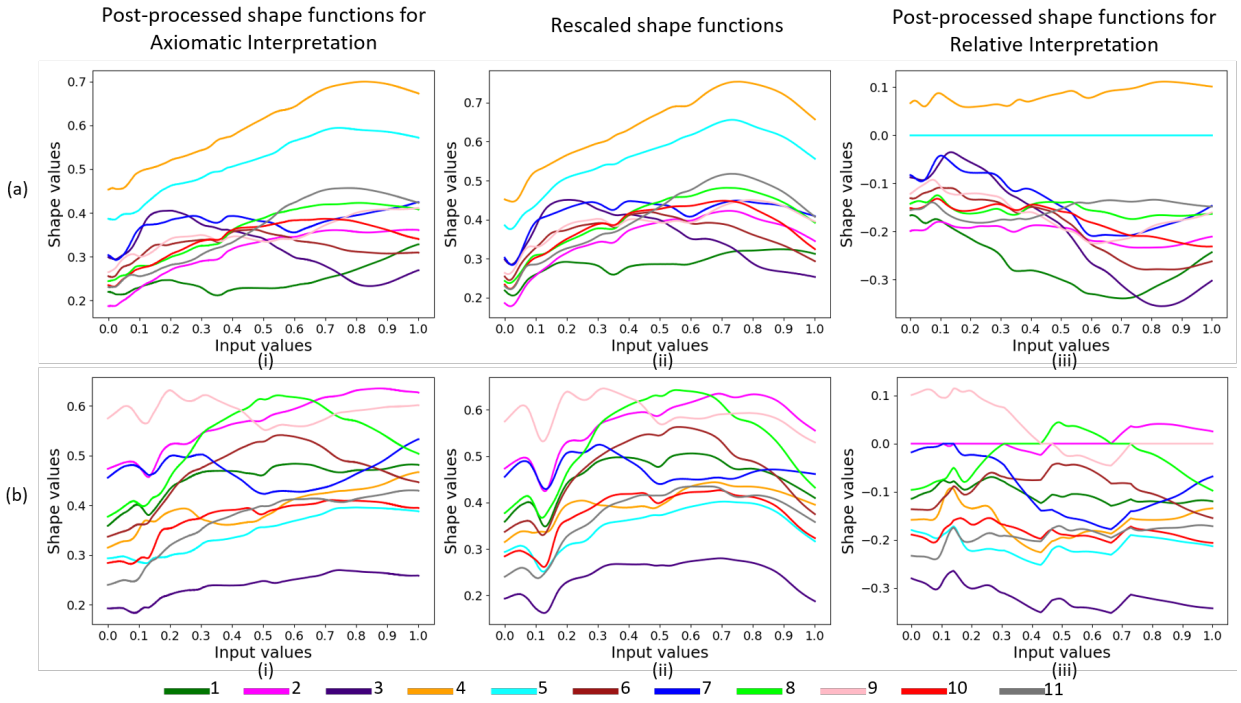

Figure 2: Comparison of axiomatic and relative interpretation methods on re-scaled shape functions for 2 features in the Sensorless data set. Each plot shows the shape functions of all the classes for a given input feature. Shape function of each class is represented by different colour code shown in the legend below.

6.1. Relative Interpretation to elicit importance of one feature across all the classes

We first study the relative importance of individual features across all classes, using the Sensorless data set. Figure 2 shows the comparison of axiomatic interpretation [13] and the proposed relative interpretation (section 4.2) for two randomly chosen input features of the Sensorless dataset. We use the re-scaled shape functions (Step 1 of Algorithm 2) to have a better visual comparison between the two interpretation methods. Thus, we apply the post-processing method in [13] on the re-scaled shape functions for axiomatic interpretation to have a better visual comparison with relative interpretation method. The individual rows in the Figure 2 represent individual features. Column 1 presents the postprocessed shape function to provide axiomatic interpretations. Column 2 presents the results after re-scaling the shape functions. Column 3 presents the postprocessed shape functions for relative interpretation.

The following observations can be made about the comparison between the axiomatic interpretation and the relative interpretation from the Figure 2.

- It can also be observed that the postprocessed shape functions for the axiomatic interpretation in Fig 2(a)(i) only smoothens the re-scaled shape function in Fig 2(a)(ii). However, it does not imply the importance of the feature for the classification.

- From the postprocessed shape functions for the relative interpretation in Fig 2(a)(iii), it can be observed that only the shape function for Class 4 has a positive value for this feature. The postprocessed shape functions for Class 5 is the second largest value that is used as the reference function, and is hence, zero for all input values of this feature. The postprocessed shape functions of all the other classes are negative. This helps us to infer that this input feature has consistently high importance for class 4 in the classification.

- Unlike the Fig 2(a)(ii), the re-scaled shape functions of the feature for all classes in Fig 2(b)(ii) are not visually separable. It can be observed that the postprocessed shape functions for the axiomatic

interpretations in Fig 2(b)(i) only smoothens this visually inseparable re-scaled shape function (Fig 2(b)(ii)), and do not elicit feature importance for individual classes.

- The postprocessed shape functions for the relative interpretation in Fig 2(b)(iii) shows that for lower values of this input ($x_i < 0.4$), this feature is of paramount importance to class 9. The feature is important to classify class 8 for the feature values $0.4 < x_i < 0.7$, while for values of this feature $x_i > 0.7$, this feature is important for class 2. It can also be observed that among the three classes for which this feature is important at different input values, its importance for class 9 at lower values is the highest.

In conclusion, it can be inferred from Figs 2(a) and 2(b) that the postprocessed shape functions for the axiomatic interpretations is always a smoothened pattern of the re-scaled shape function. This is the characteristic obtained by the post-processing method in [13], which aims at satisfying the two axioms, smoothness and monotonicity, to provide the axiomatic interpretation. However, the axiomatic interpretation does not establish the feature importance for individual classes, nor does it establish the degree of importance. On the other hand, the proposed relative interpretation establishes the importance of the different input values of the feature to the class labels, and their degree of importance.

6.2. Relative Interpretation to elicit importance of all the features for individual classes

Next, we use the Shuttle dataset to elicit importance of features for individual classes. This study can help to understand the interplay of features towards classification. Shuttle dataset is a highly imbalanced shuttle radiator data where class 1 belongs to the ideal working condition and the remaining classes belong to anomalies. It has 9 features representing the radiator valve positions.

Fig. 3 shows the importance of all the features to individual classes. Each plot shows the postprocessed shape functions for relative interpretation of all features for each class. Thus, we have 7 plots for the 7 classes in the Shuttle data set. Each plot has 9 segments, with each segment representing the shape functions for each feature. It should be noted that input values of the features are scaled in [0,1] (x-axis of individual segment), and the y-axis represents shape values. All the shape values in y-axis are in the range of $[-0.6, 0.4]$. Red and blue colours in the plots indicate the positive and negative shape values.

The following observations can be made about the relative interpretation method from the Fig. 3:

- A given input region of each feature is only important to one class. For example,different input regions of feature 1 are important for classifying classes 3, 4 and 5. However, input feature 4 is only important to class 4.

- More than one feature may be important for classifying each class. For example, input features 5, 6, 7,and 9 are important to classify class 2.

- All the features are important for classifying at least one class, with varying degree of importance.

- The degree of importance of individual features varies for classifying each class. For example, feature 1 has a higher degree of importance than other features, in classifying classes 4 and 5. This indicates that Feature 1 can be used to detect anomalies of class 4 and 5. On the other hand, certain input region of feature 7 has lower degree of importance for class 1, but it has a higher degree of importance for class 2. This shows that feature 7 can be used to detect anomaly class 2 with higher confidence.

- The SNN-t classifier may not be able to generalize a few classes. For example, only a small portion of shape values of input feature 2 are positive for class 7. This indicates that anomaly class 7 is very hard to be generalized by the classifier. This can also be verified by the fact that only 0.022% of the samples belongs to class 7.

In conclusion, the relative interpretability provides degree of importance of each feature for each class. Since the shape values of all the postprocessed shape functions are in the same range, the higher positive value directly correlates to higher importance of the feature. One should note that interpretations are always subjective and a domain knowledge is necessary to explain the interpretations.

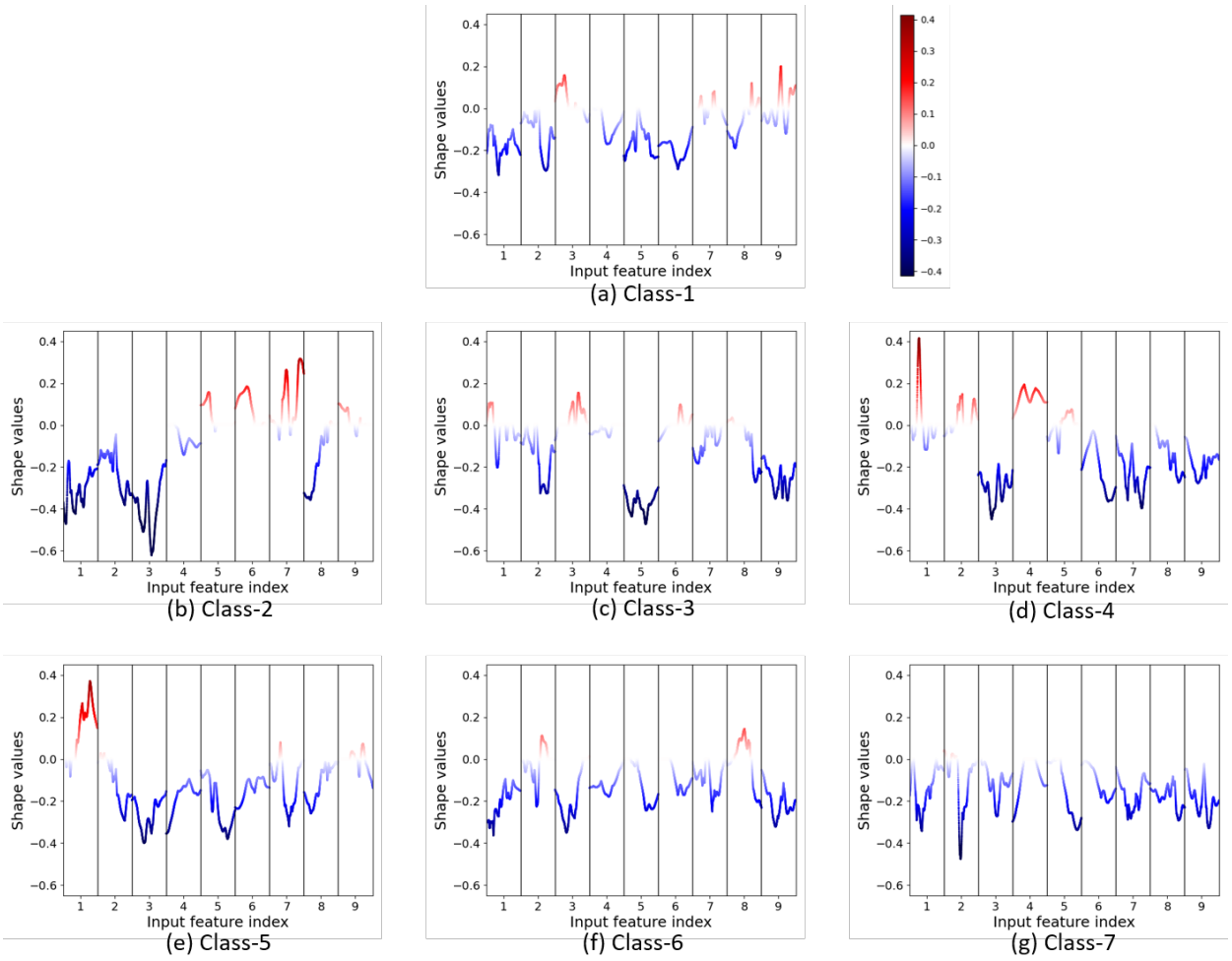

Figure 3: Relative Interpretation to elicit importance of all the features for Individual Classes using the Shuttle Dataset

7. Conclusion

In this paper, we propose a Spiking Additive Model (SAM) with a postprocessing method towards relative interpretation for multiclass classification problems. First, SNN-t is trained using a modified gradient descent learning algorithm. The functional similarity of a single layer SNN-t and GAMs is exploited to transform the time-varying weights from the temporal space of SNN-t to the feature space of GAMs. Thus, the SAM combines the generalization ability of SNN-t and the interpretability of GAM for multiclass problems. We evaluate the performance of the SAM on several multiclass problems, which show the superior classification accuracy of SAM. This shows that the gradient descent based learning algorithm improves the performance of SNN-t and the same improved performance is preserved in SAM after the weight transformation. Next, we postprocess the shape functions of SAM to obtain the relative interpretations. It is observed that the relative interpretations are richer than the other interpretation methods for multiclass GAM. It is observed that relative interpretation provides a better visualization to understand the importance of multiple shape functions, and helps to establish the importance of individual features to individual classes.

As a modified gradient descent algorithm is used to train a SNN-t with deep learning libraries, it can be easily scaled for high dimensional big data sets. With the improvement of the performance of SNN-t, the performance of SAM can be subsequently improved to derive richer interpretations.

Acknowledgement

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

References

[1] K. Simonyan, A. Vedaldi, A. Zisserman, Deep inside convolutional networks: Visualising image classification models and saliency maps, arXiv preprint arXiv:1312.6034.

[2] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, W. Samek, On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation, PLOS ONE 10 (7) (2015) 1–46.
URL https://doi.org/10.1371/journal.pone.0130140

[3] A. Shrikumar, P. Greenside, A. Kundaje, Learning important features through propagating activation differences, in: International Conference on Machine Learning, 2017, pp. 3145–3153.

[4] S. M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, in: Advances in Neural Information Processing Systems, 2017, pp. 4765–4774.

[5] M. T. Ribeiro, S. Singh, C. Guestrin, "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, ACM, 2016, pp. 1135–1144.
URL http://doi.acm.org/10.1145/2939672.2939778

[6] C. Rudin, Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead, Nature Machine Intelligence 1 (5) (2019) 206–215.

[7] A. Jeyasothy, S. Sundaram, S. Ramasamy, N. Sundararajan, A novel method for extracting interpretable knowledge from a spiking neural classifier with time-varying synaptic weights, IEEE transactions on cybernetics.
URL http://arxiv.org/abs/1904.11367

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

[9] R. Caruana, Y. Lou, J. Gehrke, P. Koch, M. Sturm, N. Elhadad, Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission, in: Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, 2015, pp. 1721–1730.

[10] A. Guisan, T. C. Edwards Jr, T. Hastie, Generalized linear and generalized additive models in studies of species distributions: setting the scene, Ecological modelling 157 (2-3) (2002) 89–100.

[11] Y. Lou, Y. Wang, S. Liang, Y. Dong, Efficiently training intelligible models for global explanations, in: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, 2020, pp. 2637–2644.

[12] A. Jeyasothy, S. Sundaram, N. Sundararajan, Sefron: A new spiking neuron model with time-varying synaptic efficacy function for pattern classification, IEEE transactions on neural networks and learning systems 30 (4) (2018) 1231–1240.

[13] X. Zhang, S. Tan, P. Koch, Y. Lou, U. Chajewska, R. Caruana, Axiomatic interpretability for multiclass additive models, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 226–234.

[14] H. Mostafa, Supervised learning based on temporal coding in spiking neural networks, IEEE transactions on neural networks and learning systems 29 (7) (2017) 3227–3235.

[15] P. H. Eilers, B. D. Marx, et al., Flexible smoothing with b-splines and penalties, Statistical science 11 (2) (1996) 89–121.

[16] H. Binder, G. Tutz, A comparison of methods for the fitting of generalized additive models, Statistics and Computing 18 (1) (2008) 87–99.

[17] Y. Lou, R. Caruana, J. Gehrke, Intelligible models for classification and regression, in: Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, 2012, pp. 150–158.

[18] Y. Lou, R. Caruana, J. Gehrke, G. Hooker, Accurate intelligible models with pairwise interactions, in: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, 2013, pp. 623–631.

[19] R. Agarwal, N. Frosst, X. Zhang, R. Caruana, G. E. Hinton, Neural additive models: Interpretable machine learning with neural nets, arXiv preprint arXiv:2004.13912.

[20] J. Chen, J. Vaughan, V. N. Nair, A. Sudjianto, Adaptive explainable neural networks (axnns), arXiv preprint arXiv:2004.02353.

[21] Z. Yang, A. Zhang, A. Sudjianto, Gami-net: An explainable neural network based on generalized additive models with structured interactions, arXiv preprint arXiv:2003.07132.

[22] S. M. Bohte, J. N. Kok, H. La Poutre, Error-backpropagation in temporally encoded networks of spiking neurons, Neurocomputing 48 (1-4) (2002) 17–37.

[23] D. Dheeru, E. Karra Taniskidou, UCI machine learning repository (2017).
URL http://archive.ics.uci.edu/ml

Appendix A. Supplementary Material

Appendix A.1. Hyperparameters

Appendix A.1.1. Population encoding:

The SNN-t is trained using a modified gradient descent algorithm. We normalized all the real-valued input feature values to be in range $[0, 1]$. Then we convert the real-values to spike pattern using population encoding scheme. We use, Gaussian functions as the receptive field neurons. The center (μ_r) and width (σ_r) of the r^{th} Gaussian receptive field used for encoding an input feature having range $[I_{min}, I_{max}]$, are initialized as,

$$\mu_r = I_{min} + \frac{(2r-3)}{2}\frac{(I_{max} - I_{min})}{q-2}, \qquad r \in \{1, \cdots, q\} \tag{A.1}$$

$$\sigma_r = \frac{1}{\gamma}\frac{(I_{max} - I_{min})}{q-2}, \qquad r \in \{1, \cdots, q\} \tag{A.2}$$

Based on the center and width of the receptive fields from Equations (A.1) and (A.2) respectively, the firing strength (ϕ_r) of the r^{th} receptive field for x is given as,

$$\phi_r = exp\left(-\frac{(x-\mu_r)^2}{2\sigma_r^2}\right) \tag{A.3}$$

We set $I_{min} = 0$, $I_{min} = 1$, overlap constant $\gamma = 0.7$ and the input spike interval $T = 3s$ for all the experiments. We used a precision of 0.01 to generate the spike times. Thus only 301 unique spike times will be produced for by each RF neuron for inputs in range $[0, 1]$. We set the number of receptive field neurons to 10 for Shuttle, Sensorless and diabetes dataset and set to 5 for covertype dataset.

Appendix A.1.2. Spiking Neural network with time-varying weights:

Only two parameters are required to model behaviour of the spiking neuron to produce an output spike. Time constant of the spike response (τ) determines how fast the postsynaptic potential increase for an input spike and firing threshold θ_j determines the sum of postsynaptic potential that requires to fire an output spike. θ_j is set to the same value for all the output classes. Generally, it is set to approximately $0.25 \times$ *sum of initial weight*

We use a batch size of 10 with stochastic gradient descent optimizer for all our experiments. We set a constant learning rate. We use Frobenius norm of the gradient of a weight matrix to determine if the gradients require normalization to avoid sudden jumps in the weight. If the Frobenius norm > 10, then we normalize the gradient of a weight matrix to have a Frobenius norm value equal to 10. The weight update range σ for the time-varying weight, determine the range of temporal neighbours that will be affected by the weight update. It is generally set to $5 - 25\%$ of the input spike interval. Lower than 5% over fits on the training dataset and a higher value of σ does not learn the variations in the input spike pattern and result in a constant weight function. Constant weight function is equivalent to single weight model.

After training SNN-t, SAM is obtained by the weight transformation method. The shape functions in SAM are obtained by tuning $\hat{s}_o$ to minimize the overall difference between $\hat{y}$ and $\hat{y}^*$ We use grid search in the range of $[3, 5]$s with precision of 0.01 to choose the $\hat{s}_o$ for each trained SNN-t. Table A.3 shows the hyperparameter setting for the datasets in table 1

Appendix A.2. Generating interpretable figures

We used SAM of Sensorless and shuttle data set to provide relative interpretation. For Sensorless dataset, we train an SNN-t and used $\hat{s}_o = 3.65s$ to obtain shape functions. We used those shape functions to generate the fig.2. We randomly choose two features from sensorless dataset. Fig.2(a) represents input feature 38 and the Fig.2(b) input feature 45. For shuttle dataset, we train an SNN-t and used $\hat{s}_o = 3.16s$ to obtain shape functions. We used those shape functions to generate the fig.3.

Table A.3: Hyper parameter setting of SNN-t for different datasets

Hyperparameter	Shuttle	Covertype	Diabetes	Sensorless
RF neurons	10	5	10	10
Time constant (τ)	1s	3s	1s	1s
Firing Threshold (θ)	25	50	150	50
Learning rate	0.5	0.5	0.5	0.8
Weight update range (σ)	0.4s	0.4s	0.4s	0.2s