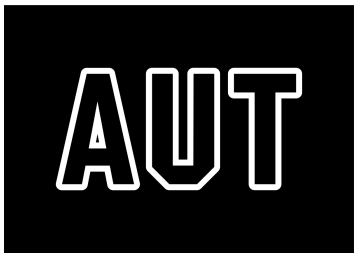


Explainable Artificial Intelligence Methods for Spiking Neural Networks

Jane Jung

Supervisor: Matthew Kuo
Nathan Allen

School of Engineering, Computer & Mathematical Sciences
Auckland University of Technology

A thesis submitted to Auckland University of Technology
in partial fulfilment of the requirements for the degree of
Master of Computer and Information Sciences

April 2026

Copyright

Attestation of Authorship

I hereby declare that except where specific reference is made to the work of others, the contents of this thesis are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgments.

Jane Jung
April 2026

COAUTHORSHIP CONTRIBUTION TEMPLATE

From the AUT Co-Authorship Protocol:

An author is an individual who has made a significant intellectual or scholarly contribution to research and its output, and agrees to be listed as an author. A significant intellectual or scholarly contribution must include one, and should include a combination of two or more, of the following:

- Conception and design of the project or output;
- Acquisition of research data where the acquisition has required significant intellectual judgement, planning, design, or input;
- Contribution of knowledge, ;
- Analysis or interpretation of research data;
- Drafting significant parts of the research output or critically revising it so as to contribute to its quality and interpretation.

For further details on the co-authorship guidelines and requirements, please refer to the AUT Co-Authorship Protocol.
For the definition of a 'manuscript' within a thesis please refer to the Postgraduate Handbook.

Co-authorship Contributions within this Thesis

Please copy the box below in to your thesis, repeated for each manuscript included in the thesis.

STUDENT AND SUPERVISOR APPROVALS

By signing you are confirming that the co-author contributions stated in the table(s) below are accurate.

Student Name	Jane Jung	**Signature**		**Date**	19/12/2025
Supervisor Name	Matthew Kuo	**Signature**		**Date**	19/12/2025

Chapter Number:	**5**
Manuscript Title:	**Surrogate Models of Spiking Neural Networks for Explainability**
Publication Status:	**Accepted for Publication**
Reference if published:	**https://rdcu.be/eVmLL**

AUTHOR SURNAME: (order as per manuscript)	CONTRIBUTION (May copy from the guidelines above)
Jung	**Conception and design of the research, analysis and interpretation of research data, development of the framework, and writing the research output.**
Kuo	**Conception and design of the project or output, contribution of knowledge, and proofreading and critically revising the research output.**
Allen	**Conception and design of the project or output, contribution of knowledge, and proofreading and critically revising the research output.**

Acknowledgements

I would like to acknowledge my supervisors Matthew Kuo and Nathan Allen for their unending patience with me, I really am grateful I was lucky enough to have the two best supervisors at AUT to be mentoring me the entire year. I'd also like to thank my parents, my sister, Kevin & Seoha for having my back outside of university – I bet none of you are ever going to see this though lol.

Abstract

As the demand for energy-efficient Artificial Intelligence (AI) grows, Spiking Neural Networks (SNNs) have emerged as a leading neuromorphic alternative to traditional deep learning. However, the complex, non-linear temporal dynamics of SNNs often result in a black box nature, hindering their adoption in high-stakes domains such as cybersecurity and clinical healthcare. This thesis proposes a novel framework, the SNN-based MLP (SNN-MLP), which utilises a non-spiking surrogate model to interpret the decision-making logic of a trained SNN. By mapping the high-dimensional activity of spiking ensembles to a differentiable architecture, the framework enables the application of post-hoc Explainable Artificial Intelligence (XAI) techniques, such as Shapley Additive ExPlanation (SHAP), to provide transparent feature-level explanations.

The framework is validated through two diverse case studies. The first case study evaluates network traffic for detection, and demonstrates a successful translation of single-dimensional SNNs into Multi-Layer Perceptrons (MLPs) and the resultant SNN-MLP successfully identifies malicious features with up to 87% accuracy, and also aligns in terms of interpretability with a baseline MLP. The second case study applies the framework to personalised depression modelling using more complex SNN architectures along with multimodal datasets. The results demonstrate that the SNN-MLP functions as a high-fidelity surrogate for multi-dimensional SNNs as well, identifying clinically relevant triggers– primarily anxiety and dietary factors – that align with established benchmarks. While the study notes challenges regarding data scarcity and class imbalance in clinical settings, the consistent overlap between the SNN and its surrogate proves that spiking architectures can achieve competitive predictive performance without sacrificing interpretability. Thus, this thesis provides a foundation for mathematically grounded, energy-efficient, and explainable AI, offering a pathway towards the deployment of explainable SNN systems in safety-critical and security infrastructures.

Publications

Jung, J., Kuo, M. M. Y., & Allen, N. (2025). Surrogate Models of Spiking Neural Networks for Explainability. In S. Wu, W. Li, X. Xu, & Y. Liu (Eds), Knowledge Management and Acquisition for Intelligent Systems (pp. 33-48). Springer Nature Singapore.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Artificial Neural Networks (ANNs) are used in the modern age in a multitude of industries. These models operate by attempting to recreate the decision-making process of the human brain with ANNs, which consist of many artificial neurons mathematically represent the architecture and dynamics of biological neurons. ANNs are useful, as they can learn underlying patterns in data without any prior knowledge of the data's distribution or characteristics (Steven Walczak et al., 2003). This makes them uniquely useful in situations where conventional algorithms struggle but the recording of large datasets is viable, and have demonstrated exceptional performance in areas like system control, intrusion detection, data classification, pattern recognition and more (Abiodun et al., 2018; Kaur and Kaur, 2012).

This momentum has extended into the industrial and medical sectors, where Internet of Things (IoT) technology and automated diagnostic systems are increasingly relied upon to manage complex, high-stakes processes. As society begins to rely more heavily on these automated systems, the focus of research is shifting toward self-learning agents capable of managing high-stakes environments, such as autonomous vehicles (Ni et al., 2020) and large-scale infrastructure control (Kayan et al., 2024). This widespread adoption has successfully established the foundational ability to create smart models; however, the current challenge lies in the refinement and sustainability of these technologies. As models grow larger to accommodate more complex tasks, the demand for servers and energy has intensified, making speed and power efficiency the primary bottlenecks for the next generation of digital products (Kariri et al., 2023).

In this context, Spiking Neural Networks (SNNs) have emerged as a significant third generation of neural network architecture that addresses these efficiency concerns by more closely emulating the event-driven computation of the biological brain (Eliasmith and Anderson, 2003). Unlike traditional Machine Learning (ML) models that rely on dense, continuous numerical activations, SNNs use sparse, temporally coded spikes. This allows for the highly

efficient processing of time-varying data – such as biosignals or neuromorphic sensor outputs – by only activating when a specific informational threshold is met (Pfeiffer and Pfeil, 2018). By moving away from the 'always-on' processing of traditional models, SNNs provide a biologically inspired alternatives that aligns with the needs of modern intelligent systems, which must operate under strict power and latency constraints at the edge, from mobile robots to wearable medical devices (Chu et al., 2022; Jiang et al., 2024).

Despite these advantages, SNNs present significant challenges regarding transparency. This issue does not necessarily apply to just SNNs though, as traditional ANNs have long been criticised for their 'black box' nature – where humans are unable to understand the decisions made by the model – due to the difficulty of tracing decisions through thousands or millions of continuous weight parameters. It is, however intensified for spiking networks, as instead of continuous weights, a single prediction within an SNN is generated through several asynchronous interactions that are nearly impossible for a human observer to trace or verify manually.

The research field for Explainable Artificial Intelligence (XAI) has been working towards addressing the issues for the opaqueness of these models, aiming to provide insights to the inner workings of ML models to humans in a way anyone can understand easily (Kumari et al., 2023).

In safety-critical domains such as cybersecurity, healthcare and autonomous decision-making, interpretability is essential for trust, transparency, and regulatory compliance (Linardatos et al., 2021a). While a wide range of XAI techniques have been proposed already, these methods are generally designed for continuous activation functions found in traditional ANNs like Multi-Layer Perceptrons (MLPs), and are not directly applicable to the spiking dynamics of SNNs. Additionally, due to SNNs being a relatively new field compared to traditional ANNs, the research done to explain SNNs is still quite minimal. However, considering the growing interest towards using SNNs for self-controlled systems, there is an urgent need for a way to explain the decisions of the SNN in a human-understandable way.

Therefore, this thesis addresses the 'black box' problem of SNNs by developing and validating a surrogate-based XAI framework. We propose an initial prototype which translates a trained SNN into an MLP, a type of ANN which retains the weights learned by the SNN as it trained as well as its behaviour using continuous-valued substitutes. By approximating spike-driven decision boundaries with continuous-valued analogues, it becomes possible to apply standard XAI, techniques to analyse the resulting MLP and infer characteristics of the original SNN. The effectiveness of this approach is demonstrated through two case studies: the first case study focuses on classification of Distributed Denial-of-Service (DDoS) attacks, and presents a demonstration for translating a simple, single-dimensional SNN as well as the

experimental results to find optimal configurations for SNNs that produce similar SNN-based MLPs (SNN-MLPs). The next case study uses the original dataset from Shah et al. (2021) focusing on mental health monitoring (depression classification) using more complex SNN architectures along with multi-class classification to further validate the proposed method on real-world applications.

This research is guided by the following research questions (RQs):

1. What are the current state-of-art studies on SNN with XAI?
2. What are the identified advantages/limitations for the identified studies on SNN with XAI?
3. How can an MLP be used as an explainable surrogate for complex SNNs whilst maintaining high performance?
4. How can the explanations derived from the XAI method be evaluated to ensure their validity and trustworthiness?

To address these research questions, the remainder of this thesis is structured as follows: Chapter 2 provides a foundational overview of ANNs and SNNs, as well as XAI domains. Chapter 3 answers RQs 1 and 2 by examining the current state-of-art XAI methods for SNNs and identifying the research gaps that we aim to address. In Chapter 4 we address the first part of RQ3 and details the methodology for our SNN to ANN conversion, as well as the resultant architecture of the proposed SNN-MLP surrogate model. The second part of RQ3 as well as RQ4 are presented in Chapter 5 and Chapter 6, where we present the results from the DDoS and depression classification case studies. Finally, we present our discussions on the results from the case studies to identify future areas of work as well as limitations as concluding remarks in Chapter 7.

Chapter 2

Background

2.1 Artificial Neural Networks

Artificial Neural Networks (ANNs) are computational models inspired by the human brain's structure and function to solve complex problems (Kaur and Kaur, 2012). Neural networks learn from examples and can adapt to changing conditions, making them effective for pattern recognition, classification, and other complex tasks They offer advantages over conventional computing methods in solving problems that are difficult to program using traditional techniques (AbouHassan et al., 2023).

2.1.1 Biological & Artificial Neurons

The fundamental building block of an ANN is the artificial neuron, which serves as a simplified abstraction of the biological neuron shown in Figure 2.1. In the brain, biological neurons receive electrical signals through highly branching structures called dendrites. The internal mechanism of the neuron consists of the membrane voltage (also called membrane potential). When no inputs are received, the neuron maintains this electric signal at a baseline level, which is known as the resting potential. As signals are received, the internal membrane potential increases; once this potential reaches a specific threshold, the neuron generates an electric pulse which are also referred to as 'spikes' or action potential, which travels along the axon to connected neurons. The output signal is then forwarded to the dendrites of the connected neurons via junctions called synapses, which are specialised junctions where the electrical signal is chemically or electrically transmitted to the next neuron. Synapses all have different levels of conductivity (or synaptic weights), which allows the brain to respond differently to various inputs. The highly parallel, weighted communication between biological neurons in the brain served as the foundational inspiration for the development of

ANNs. Researchers sought to develop artificial neurons which replicate this signal processing capability computationally. In artificial neurons, the conductivity of the biological synapses

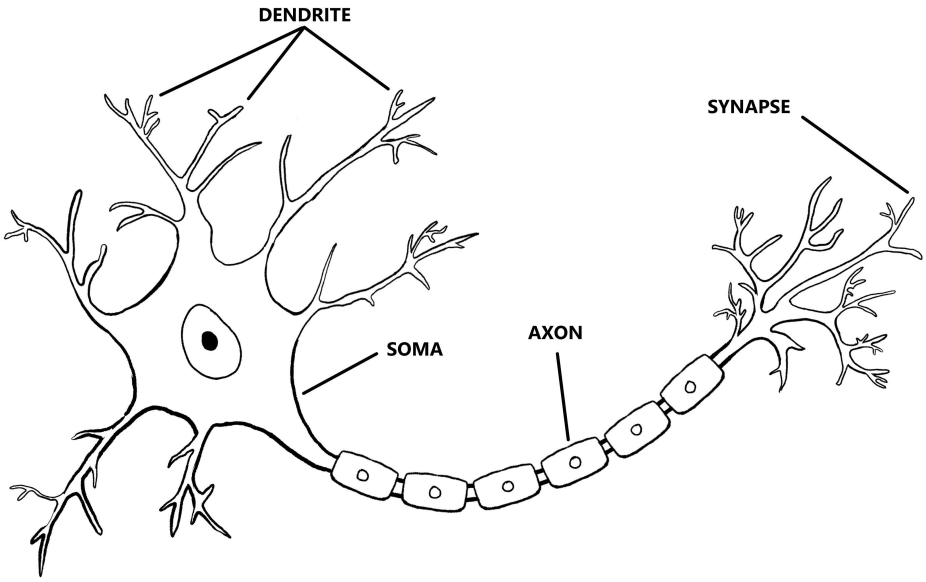


Fig. 2.1 Structure of a biological neuron.

is represented by numerical weight values which are assigned to each input connection. As illustrated in Figure 2.2, each of the input values x are multiplied with a corresponding weight w after being passed through the synapse. Then inside the soma (the cell body), these values are summed and added to a bias term (b). To replicate the biological decision to fire, the resulting sum is passed through an activation function $f(\cdot)$. The activation function is a non-linear (or sometimes linear) mathematical transformation applied to the summed input before generating the output. It introduces non-linearity to the network, which is necessary to solve complex problems and can be viewed as modelling the biological neuron's 'decision' to fire based on the membrane potential. Each artificial neuron thus receives inputs, processes them by calculating the weighted sum and applying the activation function $f(\cdot)$, and produces a single numerical output y. The example in the figure simply uses a linear activation function, but there are many more complex activation functions usable as well, shown in Table 2.1.

2.1.2 Neural Network Training

Human learning often begins with repeated exposure to examples, where an action's outcome is constantly compared against the desired result. The brain uses the difference (error) to subtly adjust the strength of synaptic connections – a concept known as synaptic plasticity –

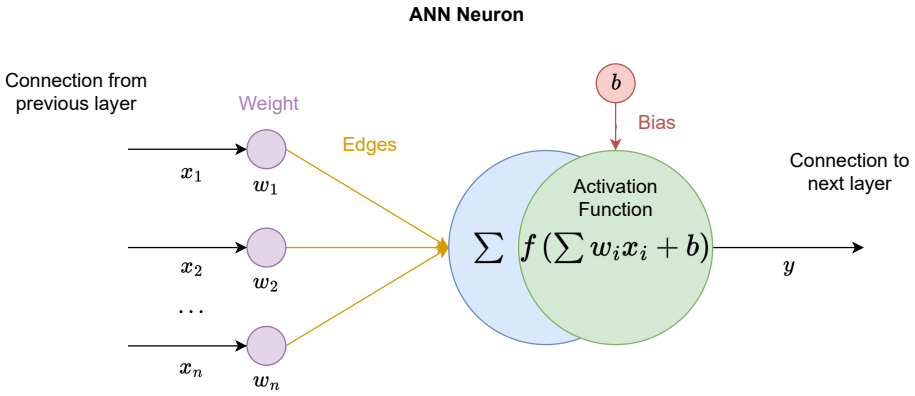
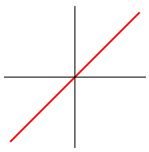
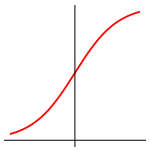
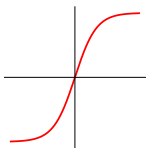
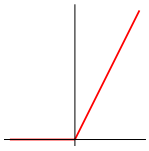
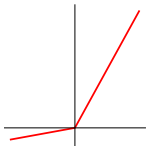


Fig. 2.2 Structure of a standard ANN neuron.

Table 2.1 Common activation functions for ANNs.

Name	Plot	Function
Linear		$f(x) = x$
Logistic Sigmoid		$f(x) = \frac{1}{1+e^{-x}}$
Tanh		$f(x) = \tanh(x)$
ReLU		$f(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}$
Leaky ReLU		$f(x) = \begin{cases} x, & x \geq 0 \\ 0.1x, & x < 0 \end{cases}$

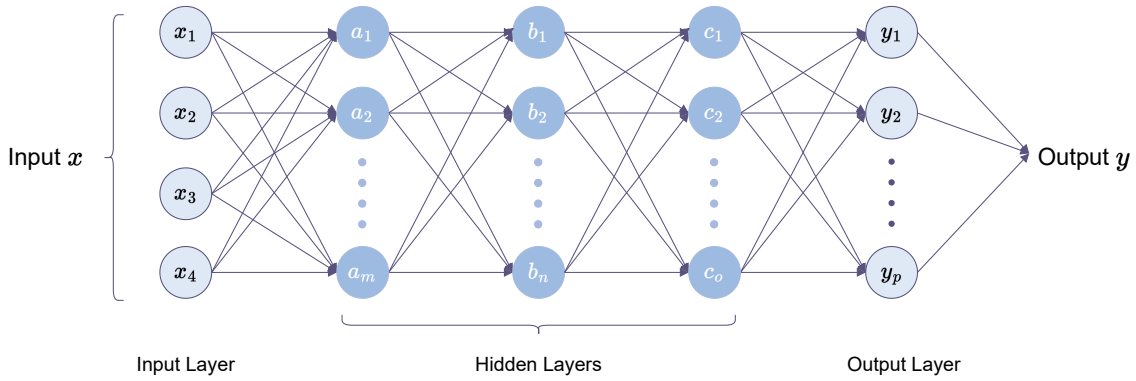


Fig. 2.3 Example of an ANN consisting of an input layer, several hidden layers and an output layer.

which gradually reinforces or weakens pathways associated with successful or unsuccessful outcomes (Abbott and Nelson, 2000). The ANN training process mathematically mimics this fundamental biological mechanism. The individual neurons described in Section 2.1.1 are created in large quantities and connected with each other by edges (which substitutes as synapses) to form complete neural networks. These networks are typically structured into distinct layers: some neurons are assigned to the input layer, while others serve as hidden or output neurons. Figure 2.3 shows the structure of an example ANN. The entire network is trained by iterating through large datasets, and updating its weights according to the learning rule used.

The most common learning approach supervised learning, which teaches the model to identify the pattern between input and output values (Nasteski, 2017). To train a neural network with supervised learning, datasets containing correct behaviour are divided into inputs (X) and the target outputs (Y). The network processes the input x to generate its predicted output y_{pred}, and compares it to the correct output – often called the target – from the dataset. The difference between the y_{target} and y_{pred} results in an error signal (σ), which is used to guide learning (Maglogiannis et al., 2007).

2.1.3 Backpropagation Algorithm

To reduce the error, the network uses a strategy called the backpropagation algorithm. This process consists of two phases: forward propagation and backward propagation. During the forward propagation, the input data travels through the network layers to produce a prediction and calculate the total error. Afterwards, during the backward propagation phase, this error is "propagated" back to the network, where it gets distributed across every connection weight within the network in proportion to that particulat weight's contribution towards the error.

This forward/backward propagation method adjusts the connection weights to reduce the error for the processed input-output pair, and is repeated while training the neural network until the desired level of accuracy is achieved. Because each weight adjustment is small, the network must process the entire dataset multiple times to effectively generalise. Each full pass through the training data is termed an epoch. Typically, datasets are divided into training and testing subsets – often in an (e.g., 80:20) ratio (Pargent et al., 2023). Additionally, to prevent the network from relying on the order of examples, it is common practice to shuffle the training set before each epoch.

2.2 Multi-Layer Perceptrons

The Multi-Layer Perceptron (MLP) is the most general type of ANN, and is widely used for tasks such as classification, regression, and function approximation (Markov, 2023). As a second-generation neural network, the MLP is characterized by its use of continuous numerical signals and differentiable activation functions. The architecture consists of an input layer, one or more intermediary hidden layers, and an output layer, as illustrated in Figure 2.3. The input layer receives the input data and parses the signals – which are numeric representations of the input data – to the intermediary layers. The intermediary layers, which are also commonly called hidden layers, process the information by passing through an activation function. The output layer then produces the predicted outputs based on the information from the hidden layers. In an MLP, each neuron in a hidden layer receives a weighted sum of all outputs from the previous layer. This sum is passed through a non-linear activation function – such as the ReLU or Sigmoid functions presented in Table 2.1 – to produce a continuous output value. These models are trained using the backpropagation algorithm, along with an optimisation algorithm known as gradient descent. Gradient descent functions by calculating the derivative of the error with respect to each weight, allowing the model to systematically adjust its internal parameters to reach the state of minimum error.

2.3 Spiking Neural Networks

Spiking Neural Networks (SNNs) are considered to be the third generation of ANNs, which mimic biological neurons more closely by incorporating the event-driven and temporal processing of the biological neurons. The biological process of neurons and their dynamics operating in continuous time (such as the changes in the membrane potential) are represented as ordinary differential equations. SNNs process inputs as voltage spikes, transmitting information through the precise timing of these spikes. Spikes, which are binary temporal

events, are transmitted across the network's connections (synapses). The network architecture allows SNNs to encode both spatial and temporal information. Spatial data is captured in the network topology (the weighted locations of synapses and neurons) that a spike traverses, while temporal data is encoded in the precise timing of their spiking activity (Kasabov, 2019). Figure 2.4 shows the inner workings of an SNN neuron. Typically, each neuron in an SNN

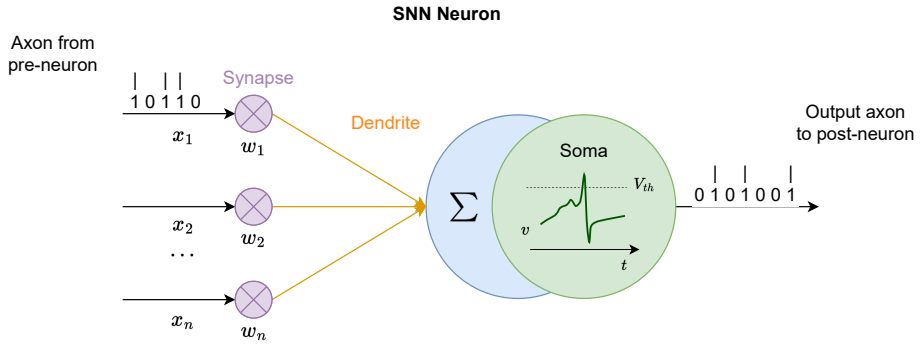


Fig. 2.4 Architecture of a standard SNN neuron.

has a neuronal post-synaptic potential (PSP), which is also referred to as the membrane potential (v). When the membrane potential crosses a set threshold (V_{th}), a spike is emitted and the membrane potential briefly drops, then slowly resets over time. These spikes are how neurons in SNNs communicate. Spikes are quick: voltage (potential) shoots up and then gets pulled down very quickly, enough so that they can be considered as discrete events, which avoids the need to calculate the differential value of the voltage across a series of time points, making them computationally efficient. Since SNN neurons only perform the primary computation (i.e., firing) when a spike is received or emitted, their computational activity is sparse. This contrasts sharply with ANNs, where all neurons and connections are computed at every forward pass, regardless of the input data. Furthermore, SNNs are intrinsically better suited for processing temporal and sequential data (Gerum and Schilling, 2021) because information is encoded in the precise timing of spikes, allowing them to extract features from time-varying input that are often lost or averaged out in rate-based ANNs (Yamazaki et al., 2022).

The mathematical representation of a spiking neuron can vary significantly depending on the required balance between biological accuracy and computational cost. Early research in computational neuroscience focused on high-fidelity models like the Hodgkin-Huxley (HH) model (Hodgkin and Huxley, 1952), which uses a set of nonlinear differential equations to describe the internal mechanisms in individual neurons. While the HH model provides a near-exact representation of biological processes, its extreme computational complexity makes it impractical for large-scale SNNs (Yamazaki et al., 2022).

To bridge the gap between biological realism and efficiency, models like the Izhikevich neuron (Izhikevich, 2003) were developed. The Izhikevich model simplified the complex dynamics of the HH equations into a two-dimensional system of ordinary differential equations (ODEs), allowing it to replicate a wide range of firing patterns with a fraction of the computational overhead. However, even this reduced complexity can be redundant for many machine learning tasks where the primary goal is pattern recognition rather than exact neurobiological simulation.

2.3.1 Leaky Integrate-and-Fire Neuron Model

The Leaky Integrate-and-Fire (LIF) Neuron Model type is another model which offers balance between performance and simplicity. LIF is a first-order spiking neuron model widely used in machine learning applications because it provides a simple way to model neuron properties and their action potentials (Braun, 2021; Gerum and Schilling, 2020).

The model characterises the neuron by its membrane potential ($v(t)$), which gradually increases due to incoming input currents ($I(t)$) until it reaches the threshold potential (V_{th}). Simultaneously, the 'leak' term causes the potential to decay exponentially back toward a resting potential (v_{rest}) in the absence of input. In a biological context, the cell membrane acts as a capacitor (C) that stores charge, while the ion channels act as a resistor (R) that allows charge to leak out. The membrane capacitance (C) determines how much charge the neuron can hold before its potential changes, effectively smoothing the input signal and influencing the speed of the neuron's response to the input current. The dynamics of a LIF neuron at time t can be represented by the following equation:

$$\tau \frac{dv(t)}{dt} = -(v(t) - v_{rest}) + RI(t) \tag{2.1}$$

where $v(t)$ represents the membrane potential at time t, and v_{rest} is the resting potential. τ represents the membrane time constant, where $\tau = RC$. A larger time constant causes the membrane potential value to decay faster towards the resting potential. As illustrated in Equation (2.1), once the membrane potential reaches the threshold V_{th}, the neuron fires a spike (i.e., the action potential) to reset the potential v back to its resting potential v_{rest}, which is held for a refractory period of τ_{ref}. Afterwards, the process presented in Equation (2.1) is repeated until there is another spike. This cycle of integration, firing, and resetting allows the LIF neuron to transform a continuous input current into a sparse, discrete temporal code.

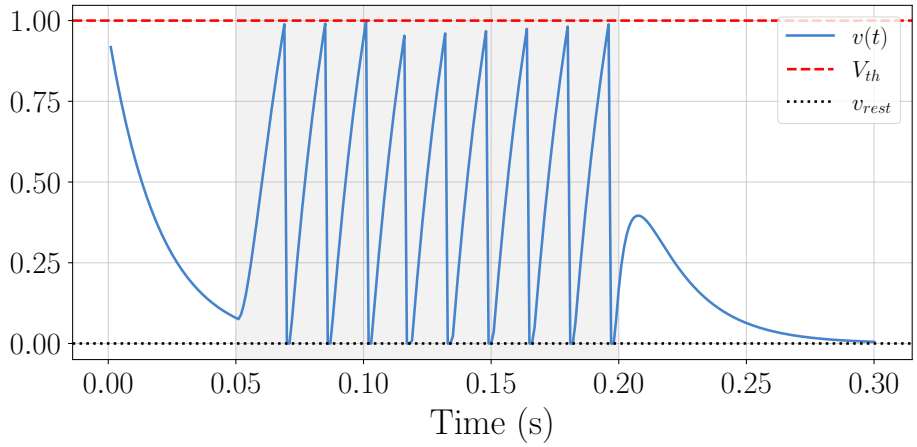
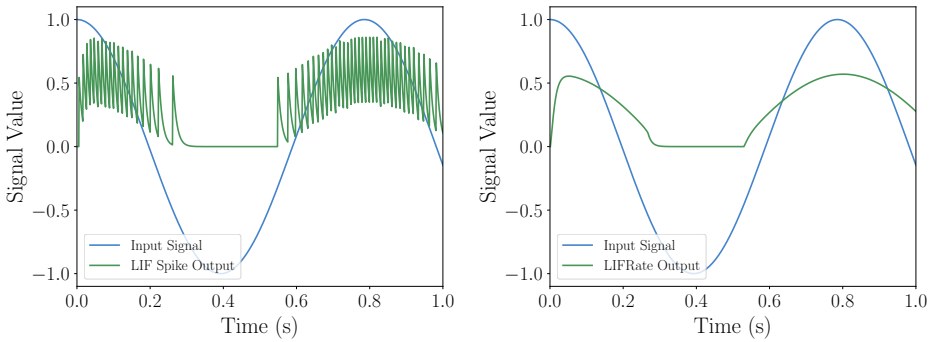


Fig. 2.5 Example of LIF neuron dynamics. The dotted red and black lines represent the threshold potential ($V_{th} = 1.0$) and the resting threshold ($V_{rest} = 0.0$). The grey area represents an active input current being fed into the neuron ($I(t) = 2.0$), and the blue line represents the membrane potential (v) at time t. The time constant of the neuron is $\tau = 0.002$.

2.3.2 Leaky Integrate-and-Fire Rate Model

The Leaky Integrate-and-Fire Rate (LIFRate) model is a rate-based approximation of the LIF model which focuses on the neuron's average firing rate rather than individual spikes (Braun, 2021). Instead of outputting either 0 or 1 like the LIF neuron model, the LIFRate model will calculate how often a neuron would 'fire" over time. This simplifies the discontinuous and event-driven behaviour of spiking neurons into a continuous, differentiable form more aligned with traditional ANNs. Figure 2.6a and Figure 2.6b show the differences in output between a LIF neuron and a LIFRate neuron respectively when presented with a sinusoidal input. We can observe that the LIFRate output value increases as the dendrite of the LIF spikes increases (i.e., higher firing rate).



(a) LIF neuron output (b) LIFRate neuron output

Fig. 2.6 Comparison of LIF and LIFRate neuron responses to sinusoidal input.

For a constant input current I, the minimum input to reach the threshold V_{th} when the potential decays back to the resting potential is $I_{\min} = V_{th}/R$ (derived from the LIF dynamics equation). Assuming that the membrane potential resets to zero ($v_{rest} = 0$), the firing rate $f(I)$ can be calculated with the following equation:

$$f(I) = \begin{cases} 0, & \text{if } IR \leq V_{th} \\ (\tau_{ref} - \tau \log(1 - \frac{V_{th}}{IR}))^{-1} & \text{otherwise} \end{cases} \tag{2.2}$$

The rate $f(I)$ is zero when the input current I multiplied by resistance R is less than or equal to the threshold potential V_{th}, as the input is insufficient to cause the neuron to spike. The non-spiking model is useful as the general LIF neuron model's discrete outputs and event-driven system make them incompatible for tasks requiring smooth, continuous representations of neural activity. With the non-spiking model, the spike timings can be abstracted into continuous firing rate values. This enables the use of surrogate models – such as MLPs – for purposes like explanation or translation, with the original characteristics of the spiking model as a basis.

2.4 Neural Engineering Framework

The Neural Engineering Framework (NEF) is a framework for representing collections of spiking neurons within neural models (Eliasmith and Anderson, 2003). While the NEF itself is an abstract set of principles, it is implemented in several software tools, among which Nengo (Bekolay et al., 2014) is a widely used Python library that facilitates building, simulating, and training SNNs based on NEF concepts. The NEF is built on three main principles: *Representation*, *Translation* and *Dynamics*.

2.4.1 Representation

The NEF defines how a population of neurons (also called ensembles) collectively represents a continuous, time-varying vector of real numbers through non-linear encoding and linear decoding.

Each neuron contributes to encoding a part of the input signal $x(t)$, using its tuning curve. The tuning curve is used to describe how the theoretical parameters such as the gain (g), bias (b), and the encoding weight $\mathbf{E}$ of the neuron translates the input vector $x(t)$ into the effective current I that drives the neuron. This relationship is formalised as:

$$I = g \cdot \mathbf{E} \cdot x(t) + b \tag{2.3}$$

where the gain represents how quickly the neuron's activity rises, and the bias represents the activity of a neuron with no given signal. The input current I is then passed through the neuron model's non-linear activation function to produce the neurons firing rate – how much the neuron will fire in response to the input. The original signal $x(t)$ is later reconstructed from the collective spike-driven responses of the entire ensemble using a set of linear decoding weights (**D**).

2.4.2 Transformation

As discussed in Section 2.1.1, neurons use synapses to communicate with each other. These communications are uni-directional – when a neuron spikes, it will transmit some amount of current to the post-synaptic (destination) neuron. The second principle is used to represent this communication a functional computations between two neural populations. Connections between neural populations implement the non-linear and linear vector transformation functions by combining the decoding weights **D** from the pre-synaptic population with the encoding weights **E** of the post-synaptic population. These weighted connections can be used to compute arbitrary functions set by the user, and enable feedforward computation in the network.

2.4.3 Dynamics

The final principle allows for modelling of systems that evolve over time, such as memory and integration. By applying the transformation principle to make recurrent connections to neuron populations (i.e., connecting its output back to its input), populations are able to sustain activity over time. This supports behaviours such as memory and temporal integration, and allows the network to evolve its internal state in response to inputs.

2.4.4 Nengo

Nengo is a python tool which takes the mathematical foundation of the NEF and translates it into a set of programmatic components, allowing researchers to build functional SNNs without needing to manually tune individual neuron parameters or synaptic weights (Bekolay et al., 2014). For this study, the SNNs will be built using the Nengo package, and thus the following section describes the key components of Nengo briefly.

Ensembles Neuron populations in Nengo are represented using ensembles (`nengo.Ensemble`). The representation principle of the NEF is embodied in Nengo ensembles, which represent

a collection of neurons which in turn represents a continuous vector. When creating a `nengo.Ensemble`, it must be initialised with the neuron type, the number of neurons in the population (n) as well as its dimensionality (d). The dimensionality is defined as the length of the vector that the ensemble is designed to represent. For instance, an ensemble representing a 2-dimensional vector would have a dimensionality of $d = 2$.

Nodes Nodes (`nengo.Node`) are not part of the NEF principles, but are helper components in the Nengo library that introduce external input into the network or to retrieve output from the network. A node can implement arbitrary non-neural processes, such as generating time-varying input signals (e.g., a constant value, a sine wave) or collecting the final decoded results of a computation. Nodes can interact with the ensembles using Connections.

Connections Connections (`nengo.Connection`) are the Nengo equivalent to synapses, and specify the flow of information between Nengo objects, such as between two Ensembles, an Ensemble and a Node, or two Nodes. The NEF transformation principle is implemented via Connections between Ensembles. When a connection is defined from a source Ensemble to a destination Ensemble, Nengo analytically calculates the necessary synaptic weights (C) to implement a user-specified function $f(x)$. This weight calculation combines the source Ensemble's decoding weights for $f(x)$ and the destination Ensemble's encoding weights, ensuring the connection implements the desired transformation. Recurrent Nengo connections (connecting a `nengo.Ensemble` to itself) are used to implement the NEF dynamics principle. Furthermore, Nengo supports biologically inspired learning rules within these connections. While the NEF typically solves for static decoders, Nengo also supports the modification of these weights through online learning rules. By specifying a learning rule on a connection, the initially calculated synaptic weights (or an initial weight matrix of zeros) can be adapted over time based on an error signal or local activity. This allows the network to learn unknown functions or adapt to changing input dynamics, similarly to an MLP. The learning rules will be discussed further in Section 2.4.5.

Probes A Probe (`nengo.Probe`) is another utility component in Nengo that is used to record the activity of any component during a simulation. A probe can record the raw spikes of an individual neuron or an entire ensemble, the decoded value ($x(t)$) that the ensemble represents, as well as the raw signals of entering/exiting a node. Probes do not affect the function or the internal dynamics of the model, and are used exclusively for data collection and analysis.

Network A Network (`nengo.Network`), is a container used to logically group and manage a collection of ensembles, nodes, connections and probes – essentially representing the entire model. Networks allow for the construction of complex, hierarchical models, promoting modularity and code reuse. The network is then passed to a Simulator (`nengo.Simulator`), which handles the execution of the model.

2.4.5 Learning in Nengo

Nengo provides built-in support for neuron models, synaptic connections, and learning rules, enabling efficient experimentation with spiking networks.

For this study, the SNN will be trained with the supervised learning approach. This methodology – as discussed in Section 2.1.2 – requires the model to use samples of correct input and target data pairs, and calculate the difference between its own outputs at the target value then adjust its weights accordingly to reduce the difference. The target label values ($Y\prime$) are provided by a target node (N_T) connected to the system, and represents the expected, correct network response (e.g., the one-hot encoded class label from Figure 4.3).

The signals from the (N_T) are fed to an error ensemble (E_σ), which is responsible for calculating the difference (error) between the desired (y_T) and predicted (y_{Y_pred}) outputs. This ensemble performs the vector subtraction $\sigma = y_{Y_\mathrm{pred}} - y_\mathrm{T}$. The output dimension of the error ensemble (d_{E_σ}) is configured to match the dimension of the network's output (d_Y). The error ensemble's decoded output (σ) is directly connected to the learning rule applied to a connection between ensembles (green dotted line), providing the signal that drives the weight adjustment for the model to learn.

Finally, an inhibition node (N_H) is included to manage the training and testing phases. This node outputs a scalar signal that is connected to the error ensemble (red connection), which can be used to force the error ensemble to output zeroes. This allows us to dynamically stop the error calculation during the testing phase of the simulation and ensure no learning occurs while performance is being evaluated on the test data.

Figure 2.7 shows the relationships between the error ensemble, target node and ensemble 2, which is considered to be the output in the scenario. Two commonly used rules in this context are the Prescribed Error-Sensitivity (PES) rule and the Recursive Least Squares (RLS) rule. These rules modify the decoder weights of neural populations in runtime, allowing networks to learn target functions or behaviours.

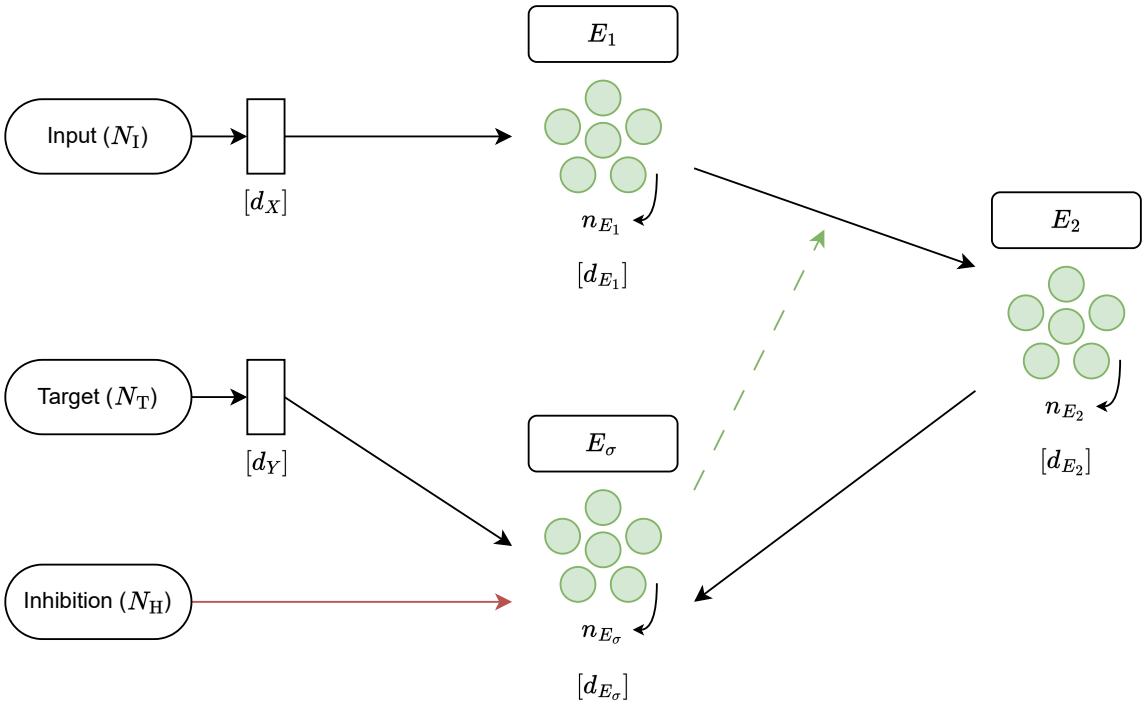


Fig. 2.7 Example structure of an SNN with an error ensemble that updates the weights between Ensembles 1 and 2.

Prescribed Error-Sensitivity Learning Rule

The PES learning rule is a supervised, biologically inspired method that updates decoder weights based on an external error signal (Banerjee et al., 2024). It operates online by adjusting weights at each timestep using the product of neural activity and error, as shown in Figure 2.7. A higher learning rate makes PES more responsive but risks instability or overfitting, especially when inputs change rapidly or remain constant for too long. PES is particularly suited for tasks requiring real-time adaptation, like motor control or function approximation.

Recursive Least Squares Learning Rule

The RLS learning rule is a more computationally intensive, but faster-converging alternative to PES. RLS aims to minimise the squared error between the predicted and target outputs by incrementally updating weights using a running estimate of the input-output correlation and the input covariance matrix. Unlike PES, RLS maintains a memory of previous activity, allowing it to adapt rapidly with higher accuracy. However, RLS is more computationally and memory-intensive than PES, which limits its scalability to large networks and real-time applications.

Table 2.2 Formal mapping of SNN components to NIF objects.

	Nengo Notation	**NIF Notation**
Node	N_{I}	$\mathrm{N}_{\mathrm{I}}^{NIF}$
Ensembles	$\{E_k \mid k \in [1,M]\}$	$\{\mathrm{E}_k^{NIF} \mid k \in [1,M]\}$
Connection Set	$\{C_{\mathrm{I},1}\} \cup \{C_{k,k+1} \mid k \in [1,M-1]\} \cup \{C_{M,P_M}\}$	$\{C_{\mathrm{I},1}^{NIF}\} \cup \{C_{k,k+1}^{NIF} \mid k \in [1,M-1]\} \cup \{C_{M,\mathrm{P}_M^{NIF}}^{NIF}\}$
Probe	P_M	P_M^{NIF}

2.4.6 Neural Interchange Format

Interchange Formats are standardised specifications used to represent models or data in a way that is independent of the original programming language, library, or hardware (e.g., XML, JSON). They act as a universal translator, allowing a model created in one environment to be seamlessly executed in another (Smith et al., 2007). The Open Neural Network Exchange (ONNX) is a common format used for machine learning, as it enables models from frameworks such as Keras or PyTorch to be ported to different inference engines (Ahmed et al., 2021).

Similarly, the Neural Interchange Format (NIF) is a format used to represent SNNs in a structured way, encapsulating essential information regarding nodes, ensembles, neural populations, connections, and probes. The NIF is compatible with Nengo, and can convert a Nengo SNN with a converter- namely, the `NengoNifConverter`. The `NengoNIFConverter` takes in the simulation object, and creates a dictionary to store the mapping of SNN components to NIF classes. Table 2.2 presents a formal mapping of the SNN components in Nengo to NIF objects, where each of the components are almost directly translated over as a key-value entry inside the `NIF.Model`.

The resultant `NIF.Model` can also be simplified (using `NIF.simplify()`) to only retain elements between two provided components (e.g., node, ensemble, probe etc) which represent the input and the output, and reduces the model's complexity (Nathan Allen, 2025). During the simplification process, Depth-First Search (DFS) is used to conduct a reachability analysis trace all possible paths between specified input and output components, and identified all the components which are either disconnected from the path, or do not contribute to the final result.

2.5 Explainable Artificial Intelligence

Explanations for AI are used to "verify the output decision made by an AI agent or algorithm" (Das and Rad, 2020) in a way that humans are able to understand and thus promote confidence in the AI agent's decision-making and its ability to make impartial decisions. The definition of "explainability" is still yet to be united, and depending on the researcher is often interchanged with "interpretability" as well (Linardatos et al., 2021b).

2.5.1 Explainable Artificial Intelligence Categories

The categorisation of the different Explainable Artificial Intelligence (XAI) methods also differ between researchers (Gamoura, 2023; Linardatos et al., 2021b). Figure 2.8 shows the taxonomy of the XAI methods. Most of the classifications between the suggested models are synonymous with one another and can be consolidated into three main classifications as follows:

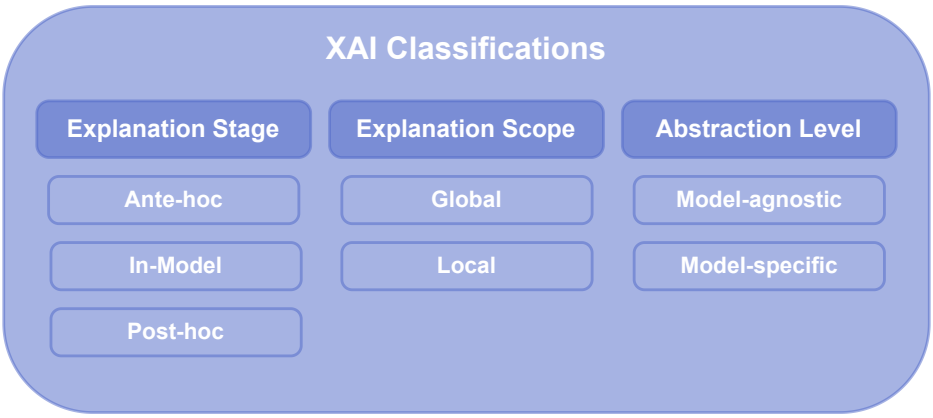


Fig. 2.8 XAI classifications.

Abstraction-based Classification

The XAI method can be classified on whether it can used for any model and is not restricted to specific model architectures. For example, if an XAI method can be used to explain both an MLP and a Convolutional Neural Network (CNN) model which have different architectures and parameters, it would be considered as a model-agnostic method. Examples of model-agnostic XAI methods include Local Interpretable Model-agnostic Explanation (LIME) (Ribeiro et al., 2016) and Shapley Additive ExPlanation (SHAP) (Lundberg and Lee, 2017). In contrast, model-specific XAI techniques are usually tailored for a specific model (e.g., ANN, Support Vector Machine (SVM)) and its parameters (Devireddy, 2025).

Scope-based Classification

The granularity of explanations in XAI methods is another distinguishing factor, categorised into local and global explanations. Local explanations focus on individual predictions or subsets of data, providing insights into how specific features influenced a particular output. For example, LIME offers a detailed, case-by-case explanation for individual model predictions. On the other hand, global explanation methods such as SHAP are used to provide a broader perspective across the entire data points and features, which makes them advantageous for identifying patterns, biases, or overall decision-making rules within the model (Abeyagunasekera et al., 2022).

Stage-based Classification

XAI methods can also be classified based on the stage at which explanations are generated: ante-hoc, in-model, or post-hoc (Ahmed et al., 2022). Ante-hoc (or pre-model) explanations occur before predictions are made. Preprocessing input data and feature selection falls under the ante-hoc explanations category, as by maintaining data quality and relevance they influence the model's training and can be used to analyse the model performance Kharal (2020). In-model explanations operate during the prediction process, utilising transparent models like decision trees or other white-box architectures to provide real-time insights. Lastly, post-hoc explanations analyse predictions after they are generated, employing techniques such as counterfactual analysis, perturbation-based methods or surrogate models to understand the influence of specific features on the model's outputs.

2.5.2 Example Explainable Artificial Intelligence Methods

While many XAI methods have proven effective in interpreting conventional ANNs such as MLPs, their success heavily relies on the underlying model assumptions, including continuous outputs and stable input-output relationships. The following section outlines generic XAI methods applied to traditional ANNs.

Shapley Additive ExPlanation

Shapley Additive ExPlanation (SHAP) is a model-agnostic and post-hoc XAI method, which can be used for both global and local explanations. The key characteristic of SHAP is its usage of cooperative game theory (Chalkiadakis et al., 2011), treating individual features as players to calculate the contribution of each feature for a particular outcome (Lundberg and Lee, 2017). SHAP assigns a value (the SHAP value, ϕ_i) to each input feature, representing

how much that feature contributes to pushing the model's prediction from the average baseline prediction to the current prediction using input values. The general variables to calculate SHAP are as follows:

- Let x represent a single instance (input features) to be explained.
- Let $f(x)$ represent the prediction function of the original black-box model for x.
- Let N represent the full set of input features.
- For a subset $S \subseteq N$ of features, denote f_S as the prediction function restricted to only the features in S.

First, the model's baseline prediction value E is calculated. This is the expected value of the prediction, calculated as the average of all predicted outputs from the training dataset (also called background dataset). Next, the marginal contribution (MC) for each feature i when it is added to a subset S is calculated:

$$\text{MC} = f_{S\cup\{i\}}(x_{S\cup\{i\}}) - f_S(x_S) \tag{2.4}$$

The final Shapley value ϕ_i for feature i is calculated by averaging the marginal contributions over all possible subsets S of features that do not contain i. This ensures fairness by considering the feature's contribution regardless of the order in which it is introduced.

$$\phi_i = \sum_{S\subseteq N\setminus\{i\}} \frac{|S|!(|N|-|S|-1)!}{|N|!} \left[f_{S\cup\{i\}}(x_{S\cup\{i\}}) - f_S(x_S)\right] \tag{2.5}$$

Here, $|N|$ represents the total number of features, and $|S|$ represents the total number of features in subset S. The additive nature of Shapley values means that they always sum up to the difference between the actual prediction and the baseline. If we put that in a machine learning context, the Shapley values for all features (players) will add up to the difference between the expected baseline prediction E and the actual model output $f(x)$.

$$f(x) = E + \sum_{i=1}^{|N|} \phi_i \tag{2.6}$$

SHAP classifies as both a local and a global level XAI method, as the SHAP value for individual predictions can then be aggregated to provide global level insights for the importance of features.

Local Interpretable Model-agnostic Explanation

Local Interpretable Model-agnostic Explanation (LIME) focuses on providing local explanations, by creating an "interpretable" surrogate model to approximate the original model (Ribeiro et al., 2016). Interpretable here means that the surrogate model will be a model that is mathematically understandable, such as linear models or decision trees which represent an estimated relationship between the inputs and the outputs. LIME, like SHAP is also model-agnostic, as it only requires the input and output data to generate calculations. For a single set of input and output values, the surrogate model attempts to find the impact (coefficient) of each input feature on the output by observing what happens to the output value when input values are perturbed.

- Let x represent a single instance (input features) to be explained.
- Let $f(x)$ represent the prediction function of the original black-box model for x.
- Let x' represent perturbed samples of x, in a set $X' = x_1', x_2', \ldots x_n'$. The perturbation process varies by data type (e.g., randomly sampling nearby features for tabular data) to create new, slightly modified inputs.
- The original model predictions are generated on these perturbed samples as $f(x_1'), f(x_2') \ldots f(x_n')$.

Then, the weight $\pi_x(x')$ of each perturbed sample x' is calculated. This weight measures the locality of the explanation. Higher weights indicate that $x\prime$ is closer to the original input value x. The weight is calculated as the difference of the distance between x' with the original instance x as $D(x,x')$, using an exponential kernel:

$$\pi_x(x') = \exp\left(\frac{D(x,x')^2}{\sigma^2}\right) \tag{2.7}$$

Where σ is a kernel parameter that controls the size of the neighbourhood around x which is considered 'local'. Note that the distance $D(x,x\prime)$ is squared to ensure that all distance values are positive. A simple linear regression model g (or any other interpretable model from the family G) is then trained as a least squares problem using the perturbed samples $x\prime$, their predictions from the original model $f(x')$, and the weights $\pi_x(x')$.

$$\underset{g \in G}{\operatorname{argmin}} \sum_{i=1}^{N} \pi_x(x_i')(f(x_i') - g(x_i'))^2 \tag{2.8}$$

Here, N represents the number of preturbed samples generated for the explanation. The interpretable model g's coefficients for each feature can then be used as the importance of

each feature to understand the local behaviour of the original black box model. Rather than grasping the model behaviour, it is more centralised in understanding the reasoning behind each prediction in context of the input features.

SHAP and LIME are particularly effective for interpreting MLPs because of the networks' relatively simple and deterministic structure with static input-output mappings. MLPs, which use the weighted sums of input features to produce continuous outputs are exploitable with SHAP and LIME due to the predictability of the model's actions. These methods rely on the assumption that small changes in input lead to proportionally interpretable changes in output, which holds true for MLPs, where each feature's contribution can be isolated and measured in a relatively straightforward manner. This allows for both local and global insights into the model's decision-making process. However, when applied to SNNs, these assumptions break down, presenting unique challenges for explainability. Unlike MLPs, SNNs use spikes and temporal patterns of neuron activity to communicate between neurons, and internal state vectors (e.g. membrane potential, refractory periods) contain temporal dependencies as well. SNN outputs are results from event-driven, non-linear interactions over time rather than a single static feed-forward pass, which means that small input perturbations can produce large, discontinuous shifts in spiking behaviour.

The traditional XAI methods like SHAP and LIME assume continuous, instantaneous input-output relationships and lack mechanisms to account for temporal dynamics or internal hidden states that drive the calculations for SNN models. Thus, they cannot reliably attribute importance to input features or timesteps in SNNs, nor capture the causal effects of spike timing and state evolution on the network's decisions.

Chapter 3

Existing Works

3.1 Systematic Literature Review Process

To establish a comprehensive overview of the current research landscape, a Systematic Literature Review (SLR) was conducted, focusing on existing studies that explored the use of Spiking Neural Networks (SNNs) combined with Explainable Artificial Intelligence (XAI) techniques of any kind. Figure 3.1 shows the SLR process used within the thesis. The objective of the review was to capture all published studies that explicitly combine SNNs with any form of XAI. The inclusion criteria was as follows:

- published between 2019 and 2024,
- peer-reviewed full articles or conference papers, and
- written in English.

The inclusion criteria for the date range was set to a five-year publication window to capture the state-of-the-art in XAI methods for SNNs. Given that the field for SNNs is relatively reacent and is rapidly evolving, studies prior to this period were deemed unlikely to reflect current best practices. However, a snowballing approach was included as a secondary method to ensure comprehensive coverage and to safely identify any foundational papers that may have otherwise been excluded.

The databases SpringerLink, IEEE Xplore, and ACM Digital Library were searched in December 2024, to capture the core technical contributions in neuromorphic engineering and explainable artificial intelligence. The search string used was:

(“explain*” OR “XAI”) AND (“spiking neur*” OR “snn*”)

These broad keywords were selected to ensure that all relevant works on the explainability of SNNs were captured. More specific terms such as "interpretability" or "explainable artificial intelligence" were trialled but proved too restrictive. The broader search using wildcards maximised coverage, although it also retrieved non-relevant results. The initial search yielded 2199 records (excluding duplicates). Screening was carried out in two phases. First, following inclusion criteria defined earlier, the records published before 2019, non-English works, non-articles (e.g., videos, images), and conference-ineligible entries from SpringerLink were excluded. This reduced the article set to 737 records. Next, the titles and abstracts of the remaining articles were screened against the inclusion criteria. At this stage, papers with unrelated definitions of "SNN" such as siamese/sequentialneural networks or statistical nearest neighbours (Frosio and Kautz, 2019; Huang et al., 2024; Schuler et al., 2023) for SNNs or those lacking genuine XAI components were removed, along with unnoticed duplicates. This left 38 eligible studies. Finally, backward snowballing of references from the 38 eligible studies yielded six additional relevant papers. This resulted in a final set of 44 studies, which are summarised and analysed in the following sections of this chapter.

Across this screening process, we found that "explainability" is not consistently defined across the literature. Instead, there are several overlapping domains of meaning, which are illustrated in Figure 3.2 and discussed in detail in the next section.

3.2 Overview of Explainable Artificial Intelligence Definitions Across Literature

Our analysis revealed that the concept of XAI is interpreted in three main domains across the reviewed works. These domains are not mutually exclusive, but rather reflect the different areas of interest that appear in the literature (Figure 3.2):

1. **AI-Reliant Explanations (AIXs):** Studies that use SNNs models primarily to explain data, phenomena, or non-Artificial Intelligence (AI) systems. Here, the explainability is not focused on making the model itself interpretable, but rather at producing explanatory insights about the task.

2. **Domains Adjacent to Explainability:** Studies that focus on techniquesto verify or validate their models. In these cases, explanation is implicitly achieved through accuracy, robustness, or trustworthiness, rather than explicitly investigated.

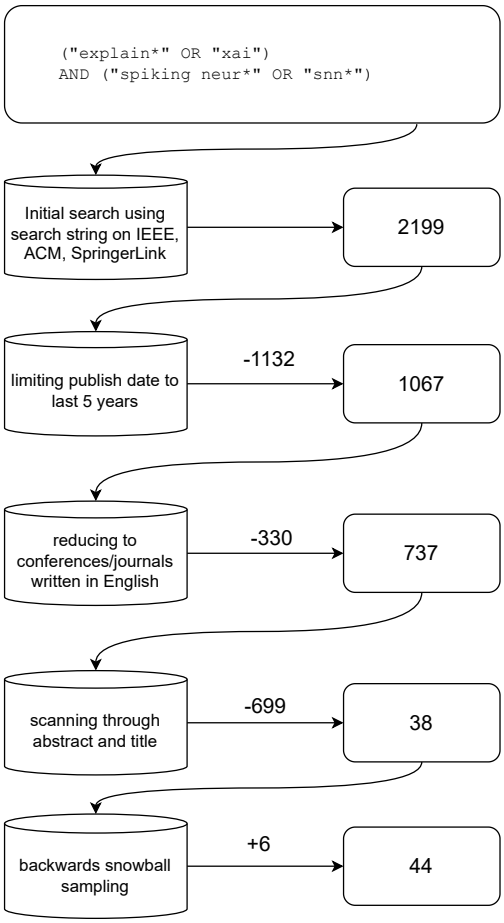


Fig. 3.1 Overview of the SLR process.

3. **True Interpretability:** Studies that directly propose or apply methods to make the internal mechanisms or decisions of SNNs interpretable to human users.

There are also works that explore XAI and SNNs in a mutually exclusive way– for example, by using some aspect or characteristic of SNNs for interpretation, or developing novel XAI methods which, although do not apply to SNNs directly, includes SNNs as part of their methodology. Although these articles do not directly contribute towards the focused domain, as they were included in the search strings, they are covered to capture the breadth of existing interpretations of explainability in the SNN context. The following sections of this chapter are structured according to these categories, providing a systematic synthesis of the 44 identified works.

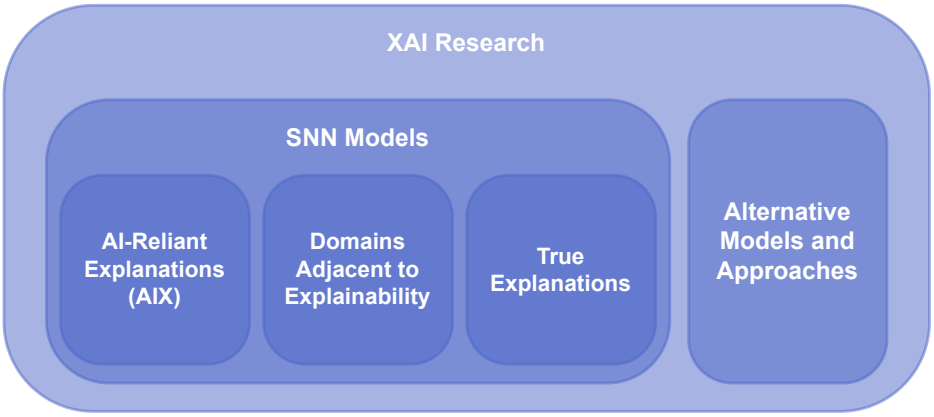


Fig. 3.2 XAI definition domains.

Table 3.1 Overview the studies in each domain.

Category	Paper References
AIX	Abouhassan et al. (2022); AbouHassan et al. (2023); Cachi et al. (2023); Chandra et al. (2024); Harbour et al. (2024); Kasabov et al. (2023); Kulkarni et al. (2013a); Xie et al. (2022); Yue et al. (2024)
Explanation Adjacent Domains	Banerjee et al. (2023); Jia et al. (2022); Khoei and Kaabouch (2024); Lin et al. (2023); Sun et al. (2023); Wu et al. (2024)
True Explanations	Bitar et al. (2023); Doborjeh et al. (2021); Hassan and Kasabov (2025); Jeyasothy et al. (2019a, 2024); Kamal et al. (2023, 2022); Kim and Panda (2021); Mikulasch et al. (2023); Nguyen et al. (2023); Perdigão (2024); Runyu et al. (2022); Sai et al. (2024); Szczęsny et al. (2023)

3.2.1 AI-Reliant Explanation

This category includes papers that consider the model itself as the explanation for their method where outlining/visualising the training process or discussing the model's architecture serves as the "explanation" justifying the model's outputs.

In neuroinformatics, NeuCube (Kasabov, 2014), was introduced to visualise brain-neuron activity using unsupervised learning with SNNs to "estimate" active neural regions of the brain in response to neuroimaging data (e.g. Electroencephalogram (EEG), Functional Magnetic Resonance Imaging (FMRI)). The goal was to more accurately mimic human brain learning and reasoning processes. As the model's behaviour becomes more human-like, the patterns of brain activity recorded from the model are suggested to become suitable replicas for biological brains, and thus make the human brain more understandable even if the model's internal decision process is not directly observable.

For example, Kasabov et al. (2023) demonstrated the use of fuzzy logic in a Brain-inspired SNN (BI-SNN) architecture to extract and evolve fuzzy spatio-temporal ruless (fSTRs) from

spatio-temporal brain data. Neuron clusters were associated with different brain region labels, and the fSTRs described the activities of these clusters at specific time points based on the outcome of the activity at the time point. Then, with transfer learning the study enabled the incremental adaptation of these rules and visualised them NeuCube to show the activity of the different neurons over time as the model learned. The use of NeuCube to visualise neuron cluster activity trends, combined with textual rule-based explanations, was subsequently evaluated to provide a sequential analysis of high- and low-performing brain regions in test subjects performing different activities (e.g., drinking from a glass of water, throwing a ball). While this method did explain the sequential events inside the brain, the level of explanation provided pushes it closer towards an analytical approach, as it only visualises the neuron activities, and the estimated activation paths are neither reasoned nor empirically verified. For example, there is no validation for the linking of the brain regions to the associated activities.

Abouhassan et al. (2022) proposed the use of evolving neuro-fuzzy (i.e., neural networks combined with fuzzy logic) SNN models for time-series forecasting. Here, evolving referred to neural networks that utilised both supervised and unsupervised learning to retain information from previous data whilst adapting (evolving) to new inputs. Three models were compared: First, an evolving fuzzy neural network (EFuNN) which used a fuzzy approach to incrementally learn by forming and evolving clusters in a "supervised clustering way" and represented the data in the form of fuzzy rules; Second, the DENFIS model which was a neuro-fuzzy model that evolved clusters incrementally from data in an unsupervised way and calculated a function for each cluster; and Third, the a dynamic evolving SNN (deSNN) which refined connection weights using spike-driven synaptic plasticity, incrementally adapting to incoming data. The fuzzy rules created by the neuro-fuzzy models were claimed to support the understanding of the model's decision process, but the "explanations" provided in the research again not identify the significance of the input variables or model features which impact these weights. Additionally, as the rules themselves were generated by the neuro-fuzzy models, the proposed method was essentially a case where a black-box was used to explain another.

Similarly, fuzzy logic was proposed with these models to visualise the behavioural patterns of the model (AbouHassan et al., 2023; Harbour et al., 2024). However, like the earlier neuro-fuzzy approaches, these methods still failed to provide clarity on the relative importance of input variables or model features in shaping the outputs.

Alternatively, an LSTM model was proposed to predict the neural spike activity within an SNN to "explain" it– which again raised the issue of the misinterpretation of explainability (Kulkarni et al., 2024).

There were also cases where the work involving SNNs ended after only using only to learn the relationship between the input and output, rather than expanding on the explainability method– for example with anomaly detection tasks using image recognition (Chandra et al., 2024; Xie et al., 2022; Yue et al., 2024), short-term forecasting (Kulkarni et al., 2013b), or sequential multi-task classification (Cachi et al., 2023). In these cases, the training of the SNN model to recognise input-output relationship was considered to be the explanation– i.e. when there is input ABC then the output is D because the SNN predicts it. The model's decision-making process was only highlighted at a high-level, and there were no explicit methods discussed to ensure that the model was reaching the conclusion through the "correct" path. Thus, the articles under this paper were unable to resolve the goal of uncovering "black box" models.

AIX table

Table 3.2 Summary of all papers in the AIX category.

Citation	Method	Contributions	Limitations
Kasabov et al. (2023)	Fuzzy Rule Extraction	Demonstrated a fuzzy logic-based BI-SNN approach for extracting and evolving spatio-temporal brain activity rules with NeuCube visualisation.	Explanations focused on visualising connections between brain regions and output, not the reasoning behind decisions. Limited BI-SNN architecture, which has a strict mapping of the neurons' locations. This makes the method less meaningful for general SNNs, as it they are not always guaranteed to have spatial assumntions for the neurons. Thus, the connections between modules are also not guaranteed to accurately reflect underlying relationships.
Abouhassan et al. (2022)	Fuzzy Rule Extraction	Comparative analysis of three different evolving neuro-fuzzy SNNs on their learning capabilities for time series data and how well they can detect and explain anomalies.	Rule extraction used another black-box model, which itself is not explainable. Explanations focused on output-region visualisations, not the reasoning behind decisions.
AbouHassan et al. (2023)	Fuzzy Rule Extraction	Presented a neuro-fuzzy learning method to visualise interactions between time-series variables and their impact on the model predictions.	Relationships were visualised, but the impact on decisions was not explained.
Harbour et al. (2024)	Fuzzy Rule Extraction	Proposed fuzzy logic for explainable SNN in drone control.	Work was theoretical and no implementations were presented.

Table 3.2 *Cont.*

Citation	Method	Contributions	Limitations
Kulkarni et al. (2024)	Surrogate Modelling	Used LSTM as a low-dimension representation of an SNN for explainability (understanding neuron/time-step significance). Implemented using an SNN simulation.	The methodology or process for training the model to correctly learn the relationship between input-output pairs does not equate to an explanation of the SNN model's behaviour. Explanations were created based on another black box (LSTM). Since an SNN simulation was used instead of an actual model, difficult to prove that results are generalisable.
Xie et al. (2022)	Model Learning Method	Presented a case study on using 'explainable' SNNs with traffic sign recognition.	The methodology or process for training the model to correctly learn the relationship between input-output pairs does not equate to an explanation of the SNN model's behaviour.
Yue et al. (2024)	Model Learning Method	Presented a case study on using 'explainable' SNNs with industrial process fault detection.	The methodology or process for training the model to correctly learn the relationship between input-output pairs does not equate to an explanation of the SNN model's behaviour.
Chandra et al. (2024)	Model Learning Method	Presented an Artificial Neural Network (ANN)-SNN hybrid model (traditional ANN with some intermediary layers using spiking LIF neurons) proposed for faster real-time glaucoma detection using image data.	The impact of the Leaky Integrate-and-Fire (LIF) neurons are proposed to produce faster results due to their spiking mechanisms. However, this is not presented in the papers' results. Although the ANN model structure is compatible with many general XAI methods, explainability is not explored with the model. The accuracy of proposed model was 60% and was outperformed by other benchmarks.
Kulkarni et al. (2013b)	Model Learning Method	Used SNN to explain the linear and non-linear relationships between like temperature, humidity, day-of-week and the forecasted load.	The SNN is used as an explanative approach, explanations on how the SNN learnt the relationships were not included. Limited discussion of model performance and also omitted comparative results from other benchmark models.

Table 3.2 *Cont.*

Citation	Method	Contributions	Limitations
Cachi et al. (2023)	Model Learning Method	Proposed novel SNN architecture enabling knowledge transfer between tasks.	Although the SNN demonstrated successful transfer learning for different tasks, the transfer learning ability does not equate to an explanation of the SNN's behaviour.

3.2.2 Domains Adjacent to Explainability

Machine Learning (ML) accuracy, robustness, and trustworthiness are all themes which recur within XAI literature, and are sometimes used interchangeably with explainability. Although these terms are not identical to explainability, they are often invoked as adjacent concepts because improvements in these dimensions are interpreted as indirect evidence of an interpretable or trustworthy model. For example, in some cases, if modifying an input consistently improves model accuracy, it can be treated as an explanation of which inputs drive the model's decisions. An extensive analysis of all the adjacent domains is beyond the scope of this thesis. However, they are not completely misaligned with explainability– thus, rather than providing an exhaustive analysis, this section outlines how these adjacent domains intersect with explainability and how they are used within the reviewed works.

XAI through Accuracy. These techniques focused on analysing the changes of accuracy through variations in model parameters to explain the model's behaviour. For example, the impact of the training data/testing data was examined: Seras et al. (2022) proposed an Out-of-Distribution (OOD) detection method based on spike count patterns in hidden layers, using heatmaps on the MNIST dataset to visualise anomalous samples. Alternatively, Zhou et al. (2023) introduced a data filtering strategy to remove the impact of malicious inputs from a trained SNN and strengthen the model's classification for benign samples, though the approach offered only limited gains in classification accuracy. Similarly, Ghosh et al. (2023) demonstrated that event-based downsampling could enhance classification performance on camera data. Beyond data-driven approaches, Elbrecht and Schuman (2020) explored compositional pattern production, showing that fixed neuron positions could exploit input-channel relationships during training and enhance the performance of the model, albeit with sensitivity to hyperparameter choices. Finally, Ferrand et al. (2023) proposed a two-compartment spiking neuron model inspired by pyramidal cells, where input from an apical dendrite (dendrite from the apex of a pyramidal cell) modulated basal activity, and showed that this mechanism improved the accuracy of context-dependent computations in recurrent SNNs. In all these cases, changes in accuracy were interpreted as indicating which inputs,

data conditions, or structural factors "explain" the network's behaviour, even though no explicit explanations such as feature-level attributions were provided.

XAI through Robustness. Robustness in Machine Learning (ML) refers to the *insensitivity of a model's performance to miscalculations of its parameters* (Tocchetti et al., 2024). Improvements in robustness are often linked to gains in explainability because identifying conditions under which the model continues to perform well (or fails) provides insight into what drives its behaviour. Several approaches have been proposed to enhance robustness– for example, Wu et al. (2024) explored the trade-off between robustness and accuracy, showing that Randomised Smoothing Coding (RSC) can significantly increase robustness compared to Poisson coding, though sometimes at the cost of slightly lower performance. Additionally, Jia et al. (2022) demonstrated that combining multi-topology networks with reward learning improved the model performance as well as its robustness. With reward learning, the model accuracy improved 0.03-0.43% across different classification tasks. The motif topology structure was further proven to improve the performance of the model by 0.76%-7.81% when the input data was incorporated with noise, which demonstrated the robustness of the model. Both studies focused primarily on improving the stability and reliability of SNNs rather than clarifying their internal reasoning processes. Thus, while they offered mechanistic insight into when and how models remained consistent under perturbation, they fell short of providing human-interpretable explanations of why those behaviours occurred.

Alternatively, Banerjee et al. (2023) introduced a formal verification framework to enable systematic analysis of SNNs behaviour and robustness. The framework was based on the satisfiability modulo theory (SMT), which are problems of determining whether a logical formula in a model is satisfiable if it adheres to a specific theory. In the approach, the entire internal mechanism of a trained SNN (e.g., firing mechanism, spike propagation) was encoded into a set of logical and arithmetic rules (e.g., a neuron does not spike if its potential does not cross its threshold), also called Quantifier-Free Linear Real Arithmetic (QF-LRA) constraints. This encoding captured the exact behaviour of both individual neurons and the full network as they evolved, allowing the SMT solver to reason symbolically about the network's computed spike trains and how they procedurally connected to the SNN output. Verification problems were then defined as logical formulas combining the SNN model and user-specified properties, which the solver was then able to use to prove whether the model satisfied a given safety or robustness condition for a set of inputs, or, if not, produced a counterexample input where it failed. By creating an explicit logical mapping from every spike event and synaptic weight to the resulting neural output, the method unveiled how internal neuron states evolved over time and what spike patterns lead to specific classification outcomes. However, the reasoning as to why certain spike patterns led to specific classification outcomes were not included

in the approach. Furthermore, another key limitation of the approach was scalability; as the number of variables and constraints grew rapidly with network depth, time steps, and perturbation bounds, the practical application of this approach was restricted to relatively small models.To summarise, although each of these studies contributed indirectly to model transparency by revealing when and how spiking networks maintain or lose stability, their primary emphasis lied in quantifying resilience to perturbations (robustness) rather than explaining model reasoning.

XAI through Efficiency. This domain includes techniques which focus on reducing model complexity (Sun et al., 2023) or minimising input data requirements (Lin et al., 2023) while maintaining performance. By creating simpler and more streamlined architectures, these methods indirectly enhance interpretability by making the relationships between inputs and outputs more traceable, allowing researchers to better isolate the factors influencing network behaviour. Furthermore, the verification of the maintained model performance despite feature reduction or model functions overlaps with the feature attribution-based explanation methods like Local Interpretable Model-agnostic Explanation (LIME) and Shapley Additive ExPlanation (SHAP).

XAI through Trustworthiness. In the context of explainable artificial intelligence, trustworthiness refers to the perceived reliability, consistency, and confidence of a model's outputs, which enable users to assess how much trust can be placed in its decisions–even when full interpretability is not achievable (Kumar et al., 2020). These techniques consider the consistency and reliability of model outputs, even when accuracy is low. A model that consistently produces the same result may still provide useful insights–such as the level of uncertainty within a system, processing time, probability of misclassification– for parameter tuning and adjustment, thereby contributing indirectly to explainability. For example, in the research done by Khoei and Kaabouch (2024), an BSNN was proposed for a safe and reliable method to detect abnormal behaviours in Industrial Control System (ICS). The proposed BSNN as well as a baseline SNN was evaluated across 3 different domains; the detection accuracy, its safety and finally its computational cost. For safety, 4 metrics were used: the aleoratic and epistemic uncertainties, the safety score and the Mean Average Percentage Error (MAPE). The aleoratic uncertainty measured the inherent randomness present in the data, such as noise or environmental fluctuations that could not be mitigated through model refinement. This quantified how much the model's confidence was affected by unpredictable conditions in the industrial control setting, with lower aleatoric uncertainty implying that the model produced consistent outputs even in noisy or variable environments. The epistemic uncertainty represented the reduceable amount uncertainty through additional training data, model tuning or enhanced model architecture, and evaluated how well the

BSNN had learned the underlying industrial system dynamics and how confident it was in extrapolating to unseen inputs. By identifying the inputs and/or decision regions the model found ambiguous/unfamiliar, the epistemic uncertainty could be reduced, thus strengthening users' trust in the model's consistency across operating conditions. The safety score measured the reliability of the model's output (i.e., how sure it was of its own prediction), and was calculated as a weighted ratio of the True Positive (TP), True Negative (TN), False Positive (FP) False Negative (FN) values, where each weight represented the confidence assigned to the correct/incorrect decisions. A high safety score was an indication that the network's decisions are reliable, which in safety-critical domains, are crucial to reinforce human operators' confidence in automated decision-making. The results showed that the BSNN achieved a safety score of 89%, compared to 74% for the baseline SNN, along with markedly lower aleatoric and epistemic uncertainties (1.83 and 0.5 vs.5.82 and 0.68). These improvements indicated that the BSNN not only identified anomalies more accurately but also assigned a higher level of confidence to its outputs, reducing the likelihood of unsafe misclassifications or undetected faults in real-time environments. Although the paper does not explicitly focus on explainability, the use of quantitative safety and uncertainty metrics indirectly supports XAI objectives. By revealing how the model quantifies and responds to uncertainty, these measures provide interpretable insights into the confidence, stability, and reliability of its outputs–which can then later be used as a foundation to improve the model itself.

Table 3.3 Summary of all papers in the 'Domains Adjacent to Explainability' category.

Citation	**Focused Domain**	**Contributions**	**Limitations**
Seras et al. (2022)	Accuracy	Developed a novel OOD detection method for SNNs, improving performance when combined with other techniques.	Claims superiority over existing OOD methods are not empirically demonstrated.
Zhou et al. (2023)	Accuracy, Trustability	Proposed data filtering to enhance accuracy and trustworthiness by removing malicious training samples.	Relies on prior knowledge of malicious data and does not address interpretability directly.
Ghosh et al. (2023)	Accuracy	Showed that event-based downsampling with LIF neurons reduces noise and simplifies outputs in camera data streams.	Papers focuses on accuracy, without explicit discussion of how downsampling impacts explainability.
Elbrecht and Schuman (2020)	Accuracy	Introduced CPPN encoding for SNNs, enabling scalable, spatially-aware training and improved performance.	Does not assess or verify if spatial structuring leads to greater interpretability.

Table 3.3 *Cont.*

Citation	Focused Domain	Contributions	Limitations
Ferrand et al. (2023)	Accuracy	Proposed a dual-compartment neuron model for context-dependent processing, inspired by biological cortical circuits to align with human cortical information processing.	Performance benefits are discussed, but verification on the likeness between the biological process versus the proposed version, and interpretability contributions remain unspecified.
Jia et al. (2022)	Robustness, Accuracy	Combined visual and auditory SNN inputs and reward learning to boost noise robustness and accuracy.	No insight into internal decision reasoning is given.
Wu et al. (2024)	Robustness, Accuracy	Demonstrated that RSC-SNN coding yields higher robustness than Poisson coding, with minimal accuracy loss.	XAI is not discussed.
Banerjee et al. (2023)	Robustness, Accuracy	Formulated a formal SMT-based verification method to systematically test SNN robustness and behaviour.	Explains when robustness holds or fails but not why specific outputs arise from spike patterns.
Sun et al. (2023)	Efficiency	Proposed efficient uncertainty calculation method for SNNs with reduced computational costs using Monte Carlo dropout.	Uncertainty estimation techniques do not directly explain decisions.
Lin et al. (2023)	Efficiency	Used dimensionality reduction to lower feature count for SNN classification while maintaining accuracy.	Does not explore how reduced input improves accuracy, which features are most relevant to a prediction and also how the feature reduction impacts the interpretability.
Khoei and Kaabouch (2024)	Trustability	Presented a BSNN for anomaly detection in ICS, outperforming baseline SNNs in safety and reliability metrics.	No discussion of XAI, difficult for scalability with larger models.

3.2.3 True Explanation

The papers of this category are the those that actually explore different explanation techniques for spiking neural networks, and are explored in the next section.

3.3 Overview of all Explainable Artificial Intelligence Techniques With Spiking Neural Networks

Szczęsny et al. (2023) introduced an XAI for SNNs that emphasises current signals rather than traditional voltage-based or weight-based interpretability approaches. Their proposed approach is achieved through a custom SNN model created with a transparent architecture founded on cusp catastrophe theory– a mathematical framework that explains systems exhibiting sudden behavioural shifts due to gradual parameter changes.

The model is characterised by its hysteresis behaviour, where system responses depend on both current and past states, enabling it to model complex neuronal dynamics such as activation and silencing in response to subtle input variations. The network architecture combines elements of traditional multi-layer neural networks with features of spiking networks. The input and first processing layers are fully connected, allowing flexible weight adjustments to capture specific input pattern features. The second layer aggregates these into representations of individual features with reduced synaptic connectivity, reflecting biologically-inspired efficiency. A final decision layer then processes aggregated feature responses to generate output. A distinguishing characteristic of the architecture is the formation of feature-focused neural clusters, or nerve ganglia, with internal neuron cooperation but no cross-ganglia communication, which enhances interpretability by isolating feature-specific processing. Each ganglion contains two neuron types, while separate decision neurons synthesise responses from these regions. The explanation mechanism leverages this architecture by tracing current signals across the network to identify meaningful correlations between input patterns and output decisions. These current signals are converted into features, from which the most influential ones are extracted and matched with input representations to provide interpretability for the network's predictions.

While the approach offers a novel perspective by focusing on current dynamics for explanation, a key limitation is the fact that the approach is model-specific, as it only works with the custom SNN model using cusp catastrophe theory. Additionally, the evaluation within the paper also lacks quantifiable evidence (e.g. comparing to other SNN-XAI methods) to verify the correctness of its explanations. The proposed solution is also only restricted to a narrow case study to detect specific ions in water samples, which raises concerns about the generalisability of the approach to broader or more complex application domains. Consequently, while conceptually innovative, the method's current formulation and scope limit its applicability as a general-purpose XAI framework for SNNs.

Jeyasothy et al. (2019a) proposed a global XAI method called Feature Strength Functions (FSFs) to explain the Multi-Class Synaptic Efficacy Function based leaky-integrate-fire

neuRON (Mc-SEFRON) model applied to multi-class classification tasks using image data. The Mc-SEFRON model, based off a previously presented model by Jeyasothy et al. (2019b), is a variant of typical SNNs with no hidden layers and a time-varying weight neuron model. Additionally, unlike conventional SNNs where synaptic weights are fixed constants, the Mc-SEFRON weights are functions of time, which allows them to dynamically modulate the Post-Synaptic Potential (PSP) accumulation based on the precise temporal window of spike arrival. This reliance on time-dependent efficacy makes the model highly sensitive to spike precision.

When training, the network operates through several distinct steps. First, the real-valued input data (e.g., feature intensity x_i) is converted into the time domain. This is done using a specific kind of input units called Receptive Field (RF) neurons (r), which convert x_i into a set of precise spike times (s_i^r) with a population encoding function ($G(\cdot)$). This encoding ensures that a higher feature intensity translates to an earlier spike time. The model is trained using a custom Spike-timing Dependent Plasticity (STDP) rule tailored to optimise these time-varying weights. Finally, the classification is determined by the first-to-spike classification derived from the encoding process, where the j-th output representing the correct class is trained to accumulate PSP fastest and fire earliest, utilising the neuron's firing threshold (θ_j) as the decision boundary. Once the Mc-SEFRON is fully trained, the FSF method is then applied on top to extract explanations following a similar set of steps to provide a global level of explanations, as illustrated in Figure 3.3. The FSF relies on

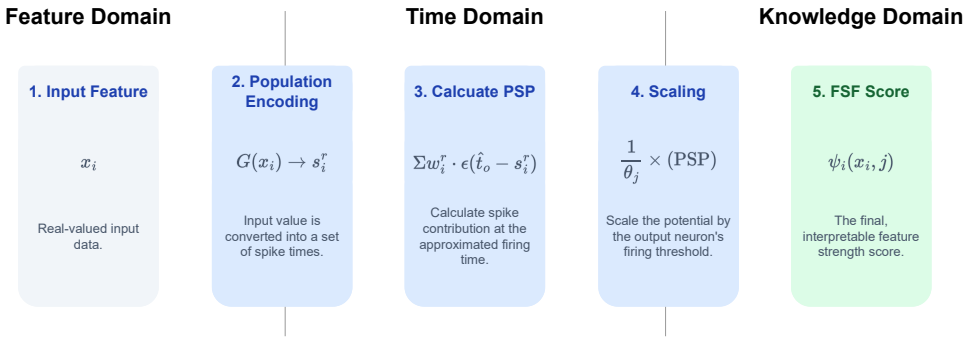


Fig. 3.3 Overview of the FSF process.

understanding the relationship between the feature domain (input data) and the time domain (SNN activity) from the trained Mc-SEFRON model. First, the process isolates a single i^{th} input feature from a sample (x_i) in the feature domain, and follows the same steps as the Mc-SEFRON to convert it into a train of spike times (s_i^r) to move the data into the temporal

domain. Then, the core temporal calculation involves summing the PSP contribution as the weighted impact of all spike times (s_i^r), which utilises the learned synaptic weights (w_i^r) during training and the spike response function (ε), measured at the *approximated* earliest firing time ($\hat{t}_o$) of the output neuron j. This PSP is then scaled/normalised by dividing it with the output neuron's firing threshold (θ_j). This scaling step performs the necessary inverse population encoding, resulting in the final FSF score ($\psi_i(x_i, j)$) as the interpretable value, which illustrates how variations in feature values influence model decisions globally.

When tested on the Tested on the MNIST dataset and the UCI image datasets, the model achieved a competitive 92.3% accuracy. FSF was then used to produce heatmaps that visually highlighted which parts of an input image predominantly contributed towards the classification outcomes, making the decision-making process interpretable. Compared to local methods such as LIME or SHAP, FSF provides a complementary global perspective, which further deepens the insight attainable with the methods. Furthermore, it actually considers internal dynamics of the SNN when generating explanations, rather than just static input-output mappings for a more relevant explanation.

The main limitation of the proposed method lies in the architecture– the main advantage of SNNs lies in their ability to more closely mimic biological operations of the human brain. However, due to the unique method of population coding and the modified STDP rule, the Mc-SEFRON model is less biologically feasible as it requires more – which then makes the advantage it holds over other ANNs such as Multi-Layer Perceptrons (MLPs) or Convolutional Neural Networks (CNNs) less clear (Hussain and Thounaojam, 2024). Additionally, although the FSF method is compatible with image datasets, the direct interpretability is lost when the model is applied to common non-spatial datasets, such as tabular data found in medical or financial domains. In these cases, features could represent abstract attributes, such as gender, or loan amount. While the FSF could technically still calculate a strength score ψ_i for these attributes, the output is merely a list of numerical contributions, offering no intuitive spatial context as with a heatmap. Also, the population encoding method used for the model is designed for continuous numerical features (i.e. summation for PSP), may not be intuitively accommodate the categorical features that are often prevalent in complex tabular datasets. Furthermore, due to the unique internal classification method that assigns labels based on the earliest output spike, the model is restricted in cases where a sample belongs to more than one class (Mishra et al., 2025).

The Mc-SEFRON model was also used in another approach. Jeyasothy et al. (2024) introduced a surrogate modelling XAI method called the Directly Interpretable Multiclass Additive model (DIMA) model. This model was derived from the previously defined Mc-SEFRON architecture, and was created by transforming the temporal classification

function of the trained Mc-SEFRON into the real-valued feature space with an inverse population encoding scheme to continuous values. The resulting weights are expressed as shape functions, which describe how each feature influences the model's output. These shape functions are then combined into a single composite function that mirrors the decision-making process of the original model as a surrogate. The final composite function takes the form of a Generalised Additive Model (GAM), which is referred as the DIMA model. These shape functions explicitly describe how individual features contribute to the classification decision, thereby eliminating the need for additional postprocessing to interpret the model's predictions.

The DIMA model was evaluated on ten standard benchmark datasets for tasks such as image classification, including datasets like iris flower recognition, wine quality, and breast cancer diagnosis. The results showed that the performance of DIMA closely matched that of Mc-SEFRON, with around 1% difference in true positive and true negative values on average. In addition, DIMA was tested on three tabular credit fraud detection datasets that varied in size and number of features. In these experiments, DIMA achieved up to a 12% improvement in accuracy over existing interpretable models, particularly when dealing with highly imbalanced data.

Due to the shape functions of the DIMA model being extracted directly from the Mc-SEFRON, the DIMA model can directly provide interpretability through feature contribution for multi-class classification without impacting the performance of the model. It also integrates the spiking dynamics to generate explanations, instead of a simple input-to-output mapping approach. However, as mentioned before, due to the Mc-SEFRON using a customised encoding and learning rule, the biological plausibility of the SNN model itself is unclear. Additionally, although the DIMA has demonstrated promising results on imbalanced datasets, its performance across more balanced or diverse datasets remains unclear, raising questions about its broader reliability.

Kim and Panda (2021) presented Spike Activation Map (SAM), a local explanation method for convolutional SNNs. SAM generates visual heatmaps based on a calculation of input feature importance (in the image classification case, these are pixels) and was studied on deep convolutional SNNs with LIF neurons on image data.

Inspired by neurobiology, SAM is based on the observation that short Inter-spike Interval (ISI) spikes carry more information as they are more likely to induce post-synaptic spikes by increasing the membrane potential of a neuron. SAM quantifies spike contributions using two metrics: Temporal Spike Contribution Score (TSCS) and Neuronal Contribution Score (NCS). The TSCS rates the contribution of a previous spike at time t' with respect to current time t, with an exponential decay to reflect the biological principle that spikes

occurring closer together are more impactful. The individual TSCS values then are summed up to compute the NCS for a neuron. A high NCS indicates that a neuron has been firing frequently in a short period. To generate the explanation, SAM computes the TSCS per spike, aggregates them into NCSs for each neuron and – for the current timestep t– multiplies the active spikes by their NCS. Summing these values across all the feature maps creates a heatmap that highlights the input regions most responsible for the current output.

In comparative experiments using Tiny-ImageNet data, SAM outperformed a SNN-based adaptation of Gradient-weighted Class Activation Mapping (GRAD-CAM) (XAI method for CNNs also by generating heatmap), yielding sharper and more accurate visualisations. GRAD-CAM calculates the gradients of a target class score with respect to the feature maps in the last convolutional layer. These gradients represent the importance of each feature map for that class. By weighting and combining these feature maps according to their gradients, Grad-CAM produces a heatmap that highlights the spatial regions of the input that most contributed to the network's decision. However, as SNNs operate with discrete spikes and binary events it makes direct gradient backpropagation ill-defined or only approximate through surrogate gradient techniques. As gradients are propagated backward through multiple layers in deep acpSNN, these approximations accumulate, leading to a non-discriminative heatmap where the beginning and the ending time-steps have little spike activity– resulting in heatmaps with zero values.

In contrast, even without target labels to specify the output class, SAM was able to generate meaningful explanations which visualises the feature (pixel) contribution across time. A key limitation for the proposed approach, however, is that it is restricted to classification tasks with image data using convolutional SNNs (model-specific), and furthermore can only be considered a local XAI method as the visualisations can only be generated for one image at a time.

Nguyen et al. (2023) presented a local XAI method called Temporal Spike Attribution (TSA) for explaining time series prediction which aggregates the information of a Fully Connected Spiking Neural Network (FC-SNN) model and identifies the contributing features.

TSA quantifies the contribution of input features based on spike timing relationships, leveraging the NCS that models how spikes temporally influence postsynaptic neurons through an exponential decay function. This is further refined by incorporating Absence of Spikes (AS), accounting for the negative contribution of absent spikes, scaled according to previous layer size for biological and computational relevance.

TSA was evaluated over three FC-SNN architectures of varying depth, testing on both synthetic binary time series data (synthetic OR-class patterns) and the "Activities of Daily Living Recognition using Binary Sensors (ADL)" dataset (Ordóñez et al., 2013), which

contained real-world multivariate sensor data tracking human activities over extended periods. The FC-SNN models achieved high accuracy on synthetic data ($\geq$90%) but a reduced performance (50%) on the ADL, indicating practical challenges in more complex temporal prediction scenarios.

For the explanation component, the authors evaluated the effectiveness of two TSA techniques: TSA using just the spikes (TSA^{S}) and TSA using the absence of spikes (TSA^{NS}). Both were evaluated against SAM using several metrics defined in the paper: correctness, output-completeness, continuity and compactness. Correctness referred to whether the explanation reflects the true behaviour of the model, which was measured by incrementally deleting the most contributing features and observing the model performance. In the paper, a low score is desirable for correctness as the model performance should drop significantly if its most significant features are removed.

Another metric proposed in the paper was "Output-completeness", which was defined in the paper as the extent a set of identified important features F was sufficient for prediction y. A perfect output-complete explanation was posited to cover all important features relevant to the prediction, but may include more features than required. Output-completeness was measured by deleting unimportant feature segments and reporting the model performance upon deletion. Continuity measured the generalisation capability of the explanation method. An explanation method is continuous if it generates similar explanations for similar input. Continuity was measured by inspecting the stability for slight variations in the input data. Lastly, Compactness referred to the size of an explanation, where a compact attribution map was desirable (to save computational resources).

TSA^{S} did not show a significant improvement of correctness than SAM, however, for both correctness and output-completeness TSA^{NS} outperformed both TSA^{S} and SAM, demonstrating that the additional incorporated of AS strengthens the explanation support for the model. Generally, TSA achieved significantly higher scores across all fields compared to SAM. TSA^{NS} achieved the strongest results for correctness and output-completeness, confirming that absent spikes play an important role in shaping model predictions, while TSA^{S} produced the most compact explanations. In terms of continuity, both TSA^{S} and TSA^{NS} surpassed SAM, with TSA^{S} performing slightly better on synthetic data and TSA^{NS} stronger on ADL data. An additional advantage of TSA discussed was its ability to distinguish between positive and negative feature attributions, providing deeper understanding of the model's reasoning for a prediction rather than highlighting only the most important features.

Despite these contributions, several limitations remain. The low accuracy (50%) of the FC-SNN models on the ADL dataset makes the case study appear less reliable for real-world applications, even if the primary focus was on explanation quality rather than

prediction performance This low performance is notable, especially since other studies (Hamad et al., 2021; Ordóñez et al., 2013) utilising the ADL dataset reported significantly better performances after testing with several models where the average f-score of all models were at 70%. Moreover, as the metrics to measure the quality of the explanations were newly defined in the paper, there is no guarantee that they are equally usable when comparing TSA to other XAI techniques. Even within the paper, there are contradictions. For example, in terms of correctness, explanations on the synthetic dataset scored higher than those on ADL data, despite better prediction accuracy with the former. Reliable explanations require an accurate neural network as a foundation, as there is no purpose on having a "correct" explanation generated if the model's accuracy is incorrect as there is no way to guarantee that the relationship learned for the explanation is accurate. This weakens the credibility of correctness as an evaluation metric, as no clear correlation was observed between model accuracy and explanation quality. The choice of SAM as a baseline is also problematic, since it was designed for image classification rather than time-series tasks; broader benchmarking against other XAI techniques would have provided stronger validation of both TSA and the proposed explanation metrics. Finally, although TSA produces biologically grounded explanations that leverage spike timing and absence, they remain difficult for non-experts to interpret, limiting their accessibility and practical uptake.

Bitar et al. (2023) proposed two gradient-based feature attribution methods for SNNs trained with a surrogate gradient backpropagation function: SNN Gradient (SNN-Grad), a saliency map-based approach and SNN Integrated Gradient (SNN-IG), an Integrated Gradients-based approach.

Each method is implemented in two forms: 3D for event-based data and 2D for real-value inputs converted to spikes. Saliency maps are used to explain traditional ANNs by taking the gradient of an output neuron with the highest activation score with respect to the inputs, and highlighting the most influential features towards a prediction (Simonyan et al., 2014). However, this definition does not directly apply to SNNs, which do not produce continuous output activations but rather sequences of discrete spikes. To adapt this, the authors propose redefining the class score as the number of spikes emitted by the output neuron corresponding to the predicted class within a specified time window. SNN-Grad then computes the surrogate gradient of this score with respect to the input spikes, producing an attribution map that highlights which spikes (or pixels) contribute most to the model's decision. The method comes in two forms: SNN-Grad3D (SNN-Grad3D), which operates on raw spiking event data, and SNN-Grad2D (SNN-Grad2D) which applies to real-valued inputs that have been converted into spikes.

SNN-IG extends the idea of Integrated Gradient (SNN-IG) to SNNs, where the gradient of the model's output is integrated along a path between a baseline input (such as a completely black image) and the actual input (Sundararajan et al., 2017). This integration process captures how input perturbations affect the output over the entire input space, providing a more complete attribution map than simple saliency methods. Since spiking event data does not have a natural continuous interpolation, SNN-IG adapts this by defining a path through time rather than pixel intensity. Here, pixel intensity simply refers to the brightness of the image pixel, as greyscale images will always be between a spectrum between black and white. For real-valued inputs that have been converted to spikes, SNN-IG2D (SNN-IG2D) follows a standard SNN-IG approach by linearly interpolating between a spike-encoded baseline and the full spike-encoded input, then integrating the gradients across these steps. For raw event-based data, SNN-IG3D (SNN-IG3D) constructs a path by gradually increasing the sampling rate of the input spikes over time, effectively interpolating between an empty event stream and the full input sequence.

Both techniques improve explainability for SNNs by leveraging gradients to compute precise attributions, addressing the shortcomings of previous methods such as SAM, which only highlight where spikes occur without revealing how specific inputs influence the output. Moreover, they allow for a more scalable and efficient approach compared to model-agnostic methods like LIME, which require perturbations of the dataset to remove the temporal vectors. However, they face the limitation that their primary focus is on convolutional SNNs, and that they're only compatible with 2D image inputs thus far. Furthermore, the qualitative assessments rely substantially on visual inspection, which can be subjective despite being supported by quantitative metrics.

Kamal et al. (2022) proposed a two-stage glaucoma prediction framework combining an SNN with an Adaptive Neuro-Fuzzy Inference System (ANFIS) to leverage both image and clinical data. In the first stage, greyscale fundus images of size 256×256 were processed by an SNN that encoded image pixel intensities into spike patterns. This was achieved by summing the pixel intensities in each image column, where spiking neurons encoded these intensities as firing frequencies within a fixed time window of 250 ms. Two synaptic models were evaluated: current-based (CUBA) and conductance-based (COBA). The extracted spike density features were then used as inputs to the ANFIS classifier, which integrates neural network learning with fuzzy logic for prediction.

Compared to standard CNN models, the proposed hybrid SNN-ANFIS system demonstrated improved accuracy as the sample size increased, suggesting its effectiveness in multimodal data integration. For explainability, the authors employed two XAI techniques: Pixel Density Analysis (PDA) to identify differences in RGB channel distributions between

glaucomatous and non-glaucomatous images, and Submodular Pick LIME (SP-LIME) to provide global feature importance summaries across predictions.

A notable limitation is that the explainability methods were applied only to the ANFIS component, not the SNN. Additionally, as PDA and SP-LIME operates on input-output mappings, its interpretations are based on the spike-based features produced by the SNN, regardless of the accuracy of the SNN. Consequently, the internal decision-making processes within the SNN remain unexamined, limiting transparency at the level of spike encoding and neuron behaviour.

Similarly, SNNs were combined with XAI techniques to recognise between cerebral micro-bleeding (CMB) in brain MRIs (Kamal et al., 2023). The authors use a two-step process: first, identifying CMB in MRI images using SNNs, then determining the specific genes in individual patients that contribute to AD development.

Like the previous paper, the SNN takes in a greyscale image with size 256×256, and the sum of the pixel values in every column is encoded by the Spiking Neurons (SNs) as firing frequencies. Then, the encoded value is used to create a pixel density map and analysed with PDA. Regions with microbleeds typically exhibit lower pixel densities compared to healthy tissue because microbleeds are small, hypointense areas that produce fewer pixels in the MRI, and these appear as darker or lower-density regions in the mapping. PDA effectively captures these low-density areas by quantifying the pixel distribution and highlighting regions where pixel density falls below certain thresholds.

The SNN model used in the paper demonstrated on average 97% accuracy, which suggests high credibility of the subsequent explanations. However, the paper doesn't include the training process of the SNN and justifications for the selection for experimental parameters, nor does it discuss the translation process of the pixel value to the weights, which reduces the validation for using the SNN in the first place. Additionally, the proposed methodology for *explaining* the model is slightly misaligned– the proposed XAI method PDA is used more so as a tool towards diagnosing CMB, rather than being used to verify the SNN model. For example, extracting the most significant feature from the outputs and the inputs does not necessarily verify that the model made the "correct" decision.

Another study explored both global and local XAI methods for interpreting LIF neuron-based SNNs applied to consumer healthcare classification tasks involving tabular datasets (Sai et al., 2024). The authors trained SNNs on a diabetes diagnosis dataset with features such as age, BMI, and blood pressure, and on a mobile health dataset containing accelerometer and heart rate sensor data to predict user activities, achieving classification accuracies of 85.0% and 97.8% respectively. Explanations were then generated for the model's output using 3 different techniques: LIME, SHAP and Partial Dependence Plot (PDP).

LIME generated instance-specific explanations– for example, glucose levels and BMI were identified to have strong influences on a positive diagnosis of diabetes, while the right wrist and left ankle were identified to contribute most effectively towards predicting the type of activity by the user. This was further supported by the analysis done with SHAP and PDP, where glucose, BMI and age were established to be the most important features for diabetes analysis, and the right wrist and left ankle were the most important features to predict the user activity.

While the study demonstrated the applicability of general XAI techniques for SNNs, it is difficult to consider a reliable approach. The pre-existing methods (LIME and SHAP), are not inherently compatible with the unique temporal and event-driven characteristics of SNNs. To force compatibility, the study first had to flatten the temporal output data from the SNN into a one-dimensional, static input format before processing it with LIME and SHAP. Consequently, the resulting explanations only estimated the relationship between flattened inputs and outputs, failing to reliably capture the crucial time-dependent dynamics of spike-timing, which is essential for understanding SNN behaviour. Additionally, there is no validation to confirm the feature importance allocations (e.g. re-testing with important/unimportant feature subsets to compare difference in performance) which also undermines the reliability of the generated explanations.

Doborjeh et al. (2021) proposed a novel explainability framework for BI-SNNs trained on Electroencephalogram (EEG) data by applying dynamic spatiotemporal clustering and rule extraction via STDP to capture underlying neural activity patterns and extract spatiotemporal rules which explain model decisions. The dynamic spatiotemporal clustering method grouped neurons based on their activation patterns while learning from streaming EEG data.

The proposed methodology begins by mapping SNN neurons to EEG electrodes based on the physical structure of the brain using the Taliarach atlas (Talairach and Szikla, 1980). Unsupervised learning is used for the model's training– as the SNN processes incoming EEG data, neurons with similar spike-driven responses form clusters dynamically. These clusters evolved throughout training, revealing spatiotemporal dependencies between different brain regions. Since the clusters emerged naturally from neuronal interactions rather than being predefined, they provided insight into how the model adapted to the data. The spatiotemporal rule extraction method then identifies sequential spike-based patterns that contribute to model predictions decisions. The authors defined these rules using spiking activity thresholds, where certain clusters fired in a specific order during EEG task processing. After extracting dynamic clusters and spatiotemporal rules via unsupervised learning, the framework applies supervised learning using a deSNN for the final classification task.

The study applied this method to EEG recordings from a GO-NOGO task, comparing brain activity between healthy individuals and opiate users. By analysing the size, shape, and membership of dynamic clusters, the authors demonstrated that SNN models could identify distinctive biomarkers differentiating the two groups with 85% accuracy. Additionally, explainability improved classification accuracy: by selecting the 8 most informative EEG features based on cluster evolution, the deSNN achieved 92% classification accuracy, outperforming traditional machine learning methods such as Support Vector Machine (SVM) and MLP. This highlighted the potential of explainable SNNs in medical and cognitive neuroscience applications, where both accuracy and interpretability are crucial.

A key contribution of the paper was the novel method for visualisation of neural clusters evolving over time, allowing tracking of interactions between different brain regions during specific cognitive states or tasks and enhance interpretability of the model's behaviour. Furthermore, the explanation method was also verified with a comparison of performance between using all features and the 8 most contributing features, strongly supporting the reliability of the approach. However, the core explainability centres on what brain regions are associated with model outcomes, rather than *why* the model made a particular decision. Additionally, while the rule extraction process provides some transparency into spike patterns leading to outputs, it lacks deeper causal reasoning or mechanistic insight– especially for SNNs not grounded in an explicit brain architecture.

Another approach for visualisation is suggested as NeuDen, a framework which integrates deSNNs anddynamic evolving neuro-fuzzy systems (deNFSs) for spatio-temporal learning, predictive modelling and explainable rule extraction from streaming data (Hassan and Kasabov, 2025).

NeuDen consists of 3 parts: First, input time series are converted into spike trains using a threshold-based temporal contrast algorithm and encoded into a three-dimensional SNNcube structure, implemented via the NeuCube architecture. In this cube, neurons are mapped to features and interconnected, allowing the unsupervised learning of temporal associations based on STDP. Once the temporal patterns are captured, the trained SNNcube is connected to a deSNN for supervised learning. From the trained deSNN, frequency-based feature vectors are extracted; these vectors summarise the spike rates and network activity over temporal windows, effectively transforming the event-based data into a form suitable for vector-based processing. Finally, the extracted feature vectors serve as input to DENFIS, which clusters the data and generates Takagi-Sugeno fuzzy rules through evolving online learning. With each new data sample, DENFIS updates clusters and regression rules, producing interpretable "if-then" fuzzy logic formulas that link input features to predicted outcomes.

Experimental evaluation of NeuDen was conducted on several real-world and benchmark datasets, including the Gas Furnace (industrial) dataset, financial stock indices, and Lebanon's economic data (NEER), which in the study were normalised into values between 0 and 1. Across all cases, NeuDen outperformed classical regression, neural, and traditional evolving connectionist system (ECOS) models in predictive accuracy, achieving a RMSE of 0.02 on stock prediction and 0.03 on NEER, compared to 0.08 for DENFIS alone and > 0.13 for other neural network models.

Aside from the superior performance, the framework also demonstrated the ability to extract evolving clusters and interpretable fuzzy "if-then" rules linked to input variables and time-windows for all datasets. Furthermore, with visualisation tools such as correlation matrices and feature interaction networks, NeuDen provided insights into how temporal and spatial features collectively influence predictions. For example, in financial data, the model revealed strong mutual influences between feature pairs (like NASDAQ and Yahoo stocks), which were visualised and traced through spiking activity and rule weights.

The biggest limitation for the proposed framework is the fact it is model-specific and tailored for integration of NeuCube and specific neuro-fuzzy architectures. Additionally, for explanations to be generated within the framework, there needs to be 3 different model training sessions, which also restricts the scalability of the method as well.

There were also cases where a new method proposed towards explainable SNN models or explanation methods but the implementation of the approach was mostly theoretical or via simulated predictions. For example, causal reasoning has been considered for SNN explanations to learn associations in patterns via unsupervised learning to apply what was learned for task A onto task B (Mikulasch et al., 2023; Perdigão, 2024; Runyu et al., 2022).

However, the work with causal reasoning were mostly still in the theoretical domain, or in cases where there were experiments, the justification for the associations learned via causal learning contributing towards explainable models were still relatively unsound as there was no clear method to evaluate how reliable the learned patterns were for task A, and how they impacted the model's behaviour for task B.

Table 3.4 Summary of all papers in the True Explanations category.

Citation	Method	Abstraction Level	Scope	Stage	Application Domain	Contributions	Limitations
Szczęsny et al. (2023)	Tracking current signals within the model as it travels through SNN	Model specific–proposed new model with interpretable architecture.	Local	In-model Post-hoc	Real-time tabular classification	Suggested an interpretable SNN for real-time classification.	The method only works with the proposed model, under constrained conditions.
Jeyasothy et al. (2019a)	FSF	Model specific–Mc-SEFRON model	Global Local	Post-hoc	Image classification	Suggested an XAI method that continuously creates heatmaps to map the spiking frequency of a neuron to its relative pixel. The changes in the heatmap can then show the impact of the pixel to the decision over time.	Works only with the Mc-SEFRON SNN model, which is not thought to be biologically plausible.
Jeyasothy et al. (2024)	DIMA	Model specific–Mc-SEFRON model	Global Local	In-model	Tabular and image classification	Proposed a method to build a surrogate model SNN as a mathematical model, which can then be used for explanation. The surrogate model does have high accuracy, which indicates that it accurately mimics the SNN behaviour.	Works only with the Mc-SEFRON SNN model, which is not thought to be biologically plausible.

Table 3.4 *Cont.*

Citation	Method	Abstraction Level	Scope	Stage	Application Domain	Contributions	Limitations
Kim and Panda (2021)	SAM	Model specific–Convolutional SNNs (CSNNs)	Local	Post-hoc	Image classification	Suggested generating heatmaps on top of images which highlight key input features contributing towards a prediction.	Works only with convolutional SNNs, restricted to local classification only.
Nguyen et al. (2023)	TSA	Model agnostic–SNNs	Local	Post-hoc	Time-series classification	Utilised SNN model weights and spike times to generate explanations for individual predictions through feature contribution over time. Novel suggestion included for metrics to evaluate explanations.	Model's accuracy is low for the case study, so there's no guarantee the explanations are correctly explaining the decision-making process of the model. Suggested metrics are also only applicable to the proposed XAI method, which limits comparability to other XAI approaches.
Bitar et al. (2023)	SNN-Grad, SNN-IG, SNN-Grad3D, SNN-IG3D, SNN-Grad2D, SNN-IG2D	Model specific–CSNNs	Local	Post-hoc	Image classification	Extended saliency maps for SNNs to be compatible with static and temporal data and explain the importance of input spikes or pixels to SNNs' output classification	Only usable with CSNNs, and only at a local scope which restricts generalisability.
Kamal et al. (2022)	ANFIS, SP-LIME, PDA	Model agnostic	Global Local	Post-hoc	Image classification	Discussed the contribution of the explanations towards the accuracy of the model (showing that the explanation method is correct).	XAI not used directly with SNN–used with ANFIS instead.

Table 3.4 *Cont.*

Citation	Method	Abstraction Level	Scope	Stage	Application Domain	Contributions	Limitations
Kamal et al. (2023)	PDA	Model agnostic	Local	Post-hoc	Image classification	Demonstrated use of SNN for image classification.	XAI not used directly with SNN– instead used more as part of the diagnostic tool alongside SNN.
Sai et al. (2024)	SHAP, LIME, PDP	Model agnostic	Global Local	Post-hoc	Tabular classification	Example application of generic XAI models onto SNNs.	Generic LIME and SHAP are used even though they are incompatible with the temporal dynamics that drive SNNs behaviour, which reduces the credibility of the explanations.
Doborjeh et al. (2021)	NeuCube – Rule extraction from neuron clustering patterns.	Model-specific– BI-SNNs	Global Local	In-model Post-hoc	Time-series classification	Used the spatial and temporal features of the brain and the native processing process to visually represent the evolving neural clusters in brain regions during model training. Discussed the contribution of the explanations towards the accuracy of the model by showing how the important features selected by the XAI method improves model accuracy.	The focus of the explanations are on visualising what brain regions are connected to the output, not why the decision is made. The proposed architecture is also only applicable for SNNs with brain-inspired topologies.

Table 3.4 *Cont.*

Citation	Method	Abstraction Level	Scope	Stage	Application Domain	Contributions	Limitations
Hassan and Kasabov (2025)	NeuDen – Predictive modelling and explainable rule extraction.	Model-specific–DENFIS framework	Global Local	In-model Post-hoc	Time-series prediction	Proposed a new framework which integrates explainability for SNNs (interpretable fuzzy rules + visualisation methods) and outperforms classical models for accuracy.	Framework is highly model-specific (evolving SNN (eSNN), deSNN, DENFIS) and can only be used with SNNs that have brain-inspired topplogies. The combination of the three models also requires several training sessions for explanations which reduces the scalability of the approach.
Runyu et al. (2022)	Rule-based Pattern Finding	Causal Inference				STDP is used to try and create a topolgy map for SNN model neurons to implement casual reasoning in the models (assosciating A with B)	Experiment from case study not discussed in detail– does not specify whether the SNN model is replicating the actions of the subject, or the EEG readings form subject.
Mikulasch et al. (2023)	N/A	Causal Inference				New suggestion that visuomotor mismatch responses in visual cortex arise as a process of "explaining away" in causal inference, rather than from dedicated error neurons in a predictive coding model.	Work is theoretical and no implementations have been presented.

Table 3.4 *Cont.*

Citation	Method	Abstraction Level	Scope	Stage	Application Domain	Contributions	Limitations
Perdigão (2024)	Suggestion on how to implement BCI into SNNs and outlines advantages of BCI-models	Causal Inference				Bayesian casual inference + SNN combination suggested to make more explainable and accurate models. Also suggests framework to provide insights into decision-making process.	Work is theoretical and no implementations have been presented.

3.4 Additional Works

While the main focus of this review is on approaches that directly integrate XAI with SNNs, several studies identified in the search addressed either interpretability or spiking computation in isolation, and are briefly summarised in the following section.

Some of the studies focused on designing new SNN architectures aimed at improving interpretability. For example, research by Jeong et al. (2024) proposed a novel neural architecture called Hierarchical Level-implemented Architecture attaining Part-Whole Interpretability (HiLite) that aimed to achieve part-whole interpretability in computer vision tasks by mimicking the hierarchical structure and information exchange mechanisms of the human cortex. Although the proposed model did use LIF neurons, the model itself was based off a completely different architecture called column networks, and the contribution of explainability was done using the characteristics of column networks, instead of the standard SNN network.

Alternatively, some research proposed novel applications for SNNs, and indirectly included some contributions towards explainability. Research by Krichmar et al. (2022) introduced a flexible path planning approach using SNNs for maze navigation. While their work did not explicitly target reasoning on why the model made certain decisions, the training process was visualised within a grid-based environment, allowing the model's actions to be tracked loop by loop, thus enabling detailed analysis of how the agent's behaviour evolved over time. Other approaches involved using counterfactual explanations with ML models at an attempt to produce an interpretable model, but similarly to the papers experimenting causal inference methods, at the time of the review the proposed approach remained in the preliminary stages and did not include a clear methodology for implementation with SNNs (Dandl et al., 2020).

In the medical domain, explainability methods have been applied to various models, occasionally alongside spiking networks but without direct integration. For Covid-19 detection, LIME was used to highlight influential regions in chest X-rays (Kamal et al., 2021; Servanshi et al., 2021). Here, SNNs were used only for classification, while explanatory analysis was performed with CNNs or SVMs, leaving the SNNs themselves uninterpreted. Other works developed novel interpretable architectures, such as the DCA model for cancer classification, which incorporated attention with discriminative feature constraints to identify biologically meaningful features (Zhang et al., 2024). Similarly, Khan et al. (2024) proposed a fusion of explainability and formal methods for healthcare decision-making, using LIME and SHAP along with the formal method Coloured Petri-Nets (CP-Nets) to analyse different ML model's predictions.

Beyond healthcare, explainability has also been explored in industrial and environmental domains. For example, Alqadhi et al. (2024) developed a hybrid deep learning model with SHAP explanations for landslide susceptibility modelling. In a different context, Machlev et al. (2022) proposed a framework to quantify the reliability of XAI methods for CNN-based monitoring of power quality disturbances. They introduced "hard" and "soft" explainability scores to evaluate occlusion sensitivity, GRAD-CAM, and LIME, with mixed results depending on disturbance type. Although their models were not spiking-based, their scoring framework suggests possible approaches for validating explanation quality more broadly.

3.5 Discussion

XAI for SNNs is still an emerging field, and so the research on XAI with SNNs is still fairly limited. The literature review for this thesis identified only 35 relevant studies, and even then only 14 had a strict focus on XAI methods for SNNs.

The lack of studies can also be attributed to the fact that the definition of explainability itself has not been consistently defined, and so many studies which claim to focus on explainability drift off onto misaligned or looser definitions. For example, a significant portion of the studies used the SNN model itself as a way to explain the relationship between input and output datasets, or a description of their model's architecture or training processes a way to 'explain' their models' behaviours. Other studies often incorrectly equated XAI with adjacent concepts such as improved accuracy and efficient, implementing robustness or increasing trustworthiness, treating enhancements in any of these domains as an indirect evidence for interpreting a model. While these themes are related, they are not fully aligned to the actual definition of XAI, and leads to shallow or incomplete explanations, failing to provide genuine insight into the model's reasoning process (the why behind the decision).

For the purpose of this thesis, XAI will be defined as a method (or methods) which aims to directly make the internal mechanisms or decisions of SNNs interpretable to human users. The rest of our discussion will therefore focus on the findings from Section 3.2.3, which align with our definition.

Some works for XAI with SNNs make use of the general XAI methods developed for traditional ANNs, such as LIME and SHAP. However, these studies only make a shallow use of these methods, often by applying them to a flattened version (average or sum) of the temporal spike trains from the SNN. While the flattening of the temporal data makes it compatible with the XAI methods, this approach ignores the core differences in how the models operate. Unlike traditional ANNs, SNNs rely on use spikes and temporal patterns of

neuron activity, and their outputs are the result of event-driven, non-linear interactions over time. Therefore, traditional XAI methods which assume a consistent scalar valued input-output relationships are not suitable for SNN as they lack mechanisms to account for these central temporal dynamics. As a result, they cannot reliably attribute importance to input features or timesteps, nor capture the causal effects of spike timing and state evolution on the network's decisions, rendering the explanations they provide incomplete or misleading.

Where explicit methods have been proposed for SNNs, they are usually tied to specific tasks, and cannot be generalised across different applications (e.g., different data types). For example, many of the examined works focus heavily on visualisation for image-based tasks, which restricts explanations to a local level and prevents them from addressing global model behaviour. Few works explore the broader potential of SNNs, which are uniquely well-suited to handle time series, neuromorphic sensing, and other crucial temporal domains where explainability could be equally, if not more, important than in image tasks. Some more comprehensive explanatory models exist, but these tend to embed the explanation mechanism directly into the network structure itself by altering some aspect of an SNN. While this can produce convincing results in specific cases, it limits the flexibility of the method and complicates scaling for larger models (e.g., adding more neurons, layers, or connections), as the additional calculations for explainability, on top of the native mechanisms for the SNN makes the method computationally expensive to execute or prone to failure, thereby limiting its usability. Additionally, as mentioned before, the main advantage of SNNs lies in its biological plausibility. This concept is central to the field, as SNNs are designed to mimic the energy-efficient, event-driven processing of the brain. If the architecture or features of the SNN has to be modified significantly to provide interpretability, then it becomes uncertain as to whether the model can still be considered as an SNN, which then brings up the question as to whether it even becomes worth using over traditional ANNs. Thus, methods which propose some alterations to the SNN structure, or its internal mechanisms for explainability are also less desirable since not only do they need to demonstrate competitive performance and high-quality explanations, they also need to justify how the approach can still be biologically plausible, which are omitted in most of the papers reviewed in this thesis. Many existing literature also lack rigorous evaluation and rely on qualitative visualisations or case studies, with little use of quantitative metrics to examine how useful the explanation method itself is. Especially in cases with local XAI techniques, as it is difficult to examine each outcome result efficiently. This makes it difficult to compare methods directly, and forces the methods to rely on the personal metrics proposed within the paper, which reduces the reliability of the methods as well.

Table 3.5 Comparison of XAI methods for SNNs evaluated against key criterion.

Citation	Dynamics-Aware	Biologically Plausible	Task-Generalisable	Evaluation of Approach
Szczęsny et al. (2023)	✓			
Jeyasothy et al. (2019a)	✓		✓	
Jeyasothy et al. (2024)	✓		✓	
Kim and Panda (2021)	✓	✓		
Nguyen et al. (2023)	✓	✓		✓
Bitar et al. (2023)	✓	✓		
Kamal et al. (2022)		✓	✓	✓
Kamal et al. (2023)		✓	✓	
Sai et al. (2024)		✓	✓	
Doborjeh et al. (2021)	✓	✓		✓
Hassan and Kasabov (2025)	✓			✓
Runyu et al. (2022)	✓			
Mikulasch et al. (2023)	✓			
Perdigão (2024)	✓			

Table 3.5 summarises the reviewed "True Explanation" XAI methods against four key criteria derived directly from the limitations identified above. The *Dynamics-Aware* criteria addresses the problem of static data flattening by indicating whether a method explicitly respects the time-dependent, spatiotemporal nature of SNNs. *Biologically Plausible* denotes methods that generate explanations without compromising the network's foundational neuromorphic mechanisms, preserving the primary advantage of SNNs. *Task-Generalisable* techniques overcome domain restrictions (such as being limited to image-classification or local classifications) by ensuring applicability across varied datasets and network architectures. Finally, *Evaluation of Approach* identifies studies that address the lack of rigorous evaluation by providing quantitative validation or empirical metrics to verify the correctness of their explanations, moving beyond purely qualitative visual evaluations. Crucially, as shown in the table, none of the reviewed methods successfully satisfy all four criteria simultaneously, highlighting a significant gap in the current literature.

In this thesis, we will address these research gaps by firstly providing a novel XAI method that maintains the temporal dynamics of the SNN model, and simultaneously provide both local and global level insights for the model's decision-making process. Furthermore, the rest of the thesis will provide validation of the XAI method proposed in the paper by experimenting with different hyperparameters and various application scenarios to evaluate the model's performance, and therefore justify the generalisability of our approach.

Chapter 4

Modelling Spiking Neural Networks Using Surrogate Multi-Layer Perceptrons

To achieve explainability for Spiking Neural Networks (SNNs), we explore the potential of converting an SNN used for classification into an interpretable Multi-Layer Perceptron (MLP) (namely, SNN-based MLP (SNN-MLP)), for models using single dimensional ensembles and multi-dimensional ensembles.

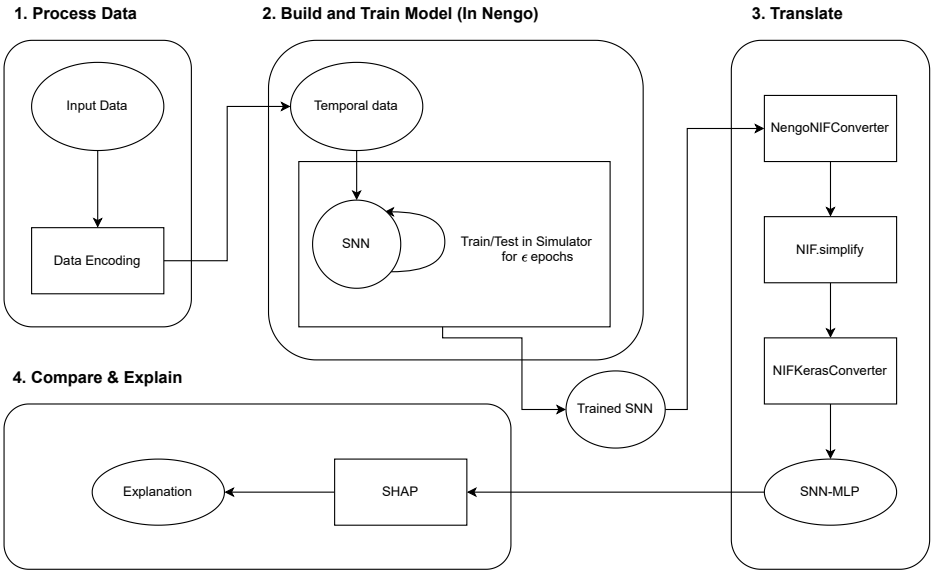


Fig. 4.1 Overview of the SNN-MLP translation methodology.

Figure 4.1 outlines the proposed process, where we first encode the input training data for classification (Phase 1 of the figure), then train an SNN model in a Nengo simulator (Phase 2) for up to n epochs. Then, in Phase 3, we take the trained SNNs, and extract information (weights) from them to build equivalent MLPs in the Keras library while maintaining the

same overall structure of the network in Phase 2. Finally, we use the generated SNN-MLP on the testing dataset, and apply Shapley Additive ExPlanation (SHAP) explanations onto the output results. If the behaviour of the SNN-MLP aligns with the SNN, we can thus presume the SNN-MLP to be an explainable surrogate of the SNN. Each of the phases are described in more detail in the following sections. Section 4.1 describes the preliminary data processing approach and Section 4.2.1 explains the configurations for building and training the SNN model. Section 4.3 presents our methodology for generating the SNN-MLP through translation, followed by our approach for applying Explainable Artificial Intelligence (XAI) methods on the surrogate SNN-MLP for meaningful interpretation in Section 4.3.3.

4.1 Temporal Data Augmentation

To enable the SNN to process the static classification data, the static inputs (X) and targets (Y) must first be transformed into a temporal spike-based format that is compatible with the SNN. For each sample in the datasets X and Y, a linearly increasing time point value with step

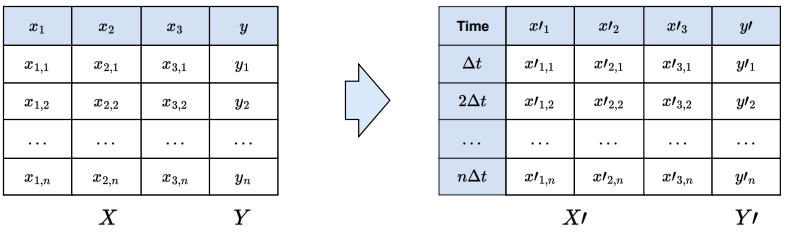


Fig. 4.2 Adding temporal vectors to input and target data.

size of Δt is assigned to each input sample in the datasets. Figure 4.2 outlines how the initial input and target datasets are translated into encoded datasets $X\prime$ and $Y\prime$ respectively. The time step (Δt) determines the duration for which a sample is being held for the SNN. For example, if $\Delta t = 0.5$, then each sample would be presented to the model for 0.5 seconds. A longer duration would enable the SNN model to have a stronger response to the input, and thus spend more time to converge its weights towards the input data. For a single-dimensional (e.g. binary) output y in the encoded target dataset $Y\prime$, the categorical labels are converted into numeric values to represent the range the model output can be interpreted for (e.g. 0 and 1 or -1 and 1).

4.1.1 Multi-class Data Encoding

For binary classification, a threshold can be used to classify the SNN output into two categories. However, for datasets with k output classes like in the depression dataset (7

output classes), using a single continuous value for the output becomes less reliable due to the ambiguity of setting the thresholds between multiple classes. To address this, in multi-class classifications each of the class labels in $Y\prime$ are converted into a k-dimensional one-hot encoded vector before being presented to the SNN's target node, as shown in Figure 4.3. Each vector is assigned the same time step value (Δt) as shown before in Figure 4.2.

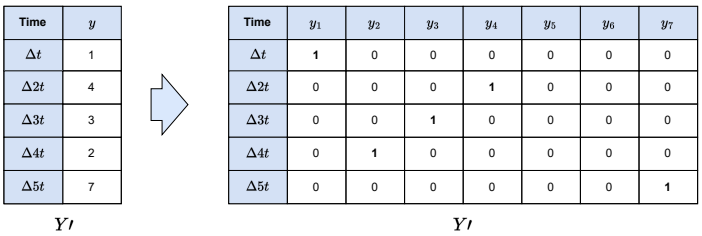


Fig. 4.3 Updating target dimensions with one-hot encoding.

4.2 Spiking Neural Network Configuration

The SNN for this study is built using the Nengo package in Python. As discussed in Section 2.4.4 in Nengo, an SNN is constructed from several computational elements, including input nodes and a set of connected ensembles (configured either in series or parallel) that facilitate information transfer.

The input node N_I passes through the values of the input dataset $X\prime$. In our architecture, the node simply acts as a linear pass-through, and outputs a signal with the size of the feature dimension, d_X.

Each ensemble (E) represents a population of n spiking neurons designed to collectively encode and decode d-dimensional vectors. The dimension d_E is customisable, and defaults to a value of 1. The neurons collectively encode the input vector i into their spiking activity, and this encoded activity can then be decoded back into another vector o, where both i and o have the size of d_E.

Connections are used for communication between the SNN components, defining the data flow, and controlling how signals are transformed and learning occurs. A Connection C applies a weight matrix $\mathbf{W}$, whose size is dictated by the output size of the source object and the input size expected by the destination object, giving it the shape $(d_\mathrm{source}, d_\mathrm{dest})$.

Probes are used to collect the data from the SNN components during the simulation (i.e., output value of the SNN), and has their input size matching the output size of the targeted component. Probes are used exclusively to record the activity (e.g., spikes, decoded vector outputs) over time for analysis.

4.2.1 Building the Spiking Neural Network

The SNN consists of 3 nodes representing the input N_I, target N_T, and inhibition N_H. The input node feeds in the feature vector x into the network. Its output size is the input feature dimension ($d_{X\prime}$). The target node provides the true label vector y used for supervised learning, and its output size is the target label dimension ($d_{Y\prime}$). The inhibition node uses the current time t of the simulation to check whether model is in the training phase or testing phase, and outputs a signal with shape $d_{Y\prime}$.

The SNN also contains a sequence of M ensembles, E_1, E_1,... E_M where each ensemble E_k has d_{E_k} dimensions and n_{E_k} neurons.

The first ensemble in the sequence E_1, receives the input signal i from the input node N_I, matching the input feature dimension d_{N_I}, such that $d_{E_1} = d_{N_\mathrm{I}}$. The last ensemble in the sequence, E_M, outputs the predicted values Y_pred with shape $d_{Y\prime}$. The dimension for the last ensemble thus matches the dimension of the target dataset $d_{Y\prime}$. Additionally, there's also the error ensemble E_σ, which is connected to both the last ensemble E_M (providing predicted values) and the target node N_T (providing actual values). This error ensemble is configured to have the same dimensionality as the target signal, d_{N_T}, meaning $d_{E_\sigma} = d_{Y\prime}$. Finally, the inhibition node is connected to the error ensemble, and applies a time-dependent signal $\gamma(t)$ to restrict learning during the testing phase of the simulation, as detailed in Section 2.4.5.

To collect the decoded output from the SNN model, the final ensemble E_M in the model is assigned a probe P_M, which collects the decoded output from the ensemble. The input dimension of the probe is automatically configured as the output dimension of the connected object, which in this case would be d_{E_M}. Thus, the values recorded of the probe can be used to interpret the final output decision of the SNN.

The SNN has a variety of tuneable parameters which impact its learning– such as the learning rate (η), the simulation timestep (Δt), and learning rule (i.e., Prescribed Error-Sensitivity (PES), Recursive Least Squares (RLS)). The learning rate determines how quickly the connection weights are adjusted in response to the error signal, while the simulation timestep defines how long each input sample is presented to the network during simulation as discussed in Section 4.1.

4.2.2 Training the Spiking Neural Network

To train the model, the entire input dataset ($X\prime$) as well as the corresponding targets ($Y\prime$) is presented to the model in a simulation loop. This single pass through the dataset is called an epoch (ε). The total duration of the epoch is calculated using the number of samples in the

input dataset ($N_{X\prime}$), where each sample is presented to the model for Δt seconds.

$$T_\varepsilon = N_X \times \Delta t$$

The simulator in Nengo needs to be configured with the total time duration T_{sim} before starting. For a single epoch, the total duration would simply be calculated as $T_{\text{sim}} = T_\varepsilon$

With smaller datasets, a single epoch may not be enough for the model to learn the complex relationships, and adjust its weights accordingly. To address this, models are typically trained iteratively, where model processes the training data for several epochs. This exposes the model to each sample several times, which allows it to have a more comprehensive understanding of the entire dataset. Additionally, in each epoch once the model is trained, it can then be tested on the testing dataset. The changes in the performance at every epoch can then be monitored to track the learning progression and identify stages where the model starts to overfit (memorising the training data rather than learning the relationships).

Traditional Artificial Neural Network (ANN) training environments are thus equipped with helper tools to reload batches of the datasets, and also allows the model to be adjusted dynamically between training and testing modes.

However, the Nengo simulation environment requires a static model configuration throughout its duration. If the model were to be trained for several epochs ε, the simulator's runtime duration needs to be configured and set before the simulation begins.

$$T_{\text{sim}} = \varepsilon \times T_\varepsilon = \varepsilon \times N_{X\prime} \times \Delta t$$

Additionally, in Nengo, it is not possible to dynamically pause the simulation to reload the dataset with new timesteps. Furthermore, it also prevents enabling or disabling of the learning rule by updating the model parameters (such as the inhibitory weights controlling the error signal) mid-simulation to stop the learning during the testing phase at the end of an epoch, then enable it again at the start of the training phase in the next epoch.

Thus, if the simulation were to run for several epochs, the most straightforward approach, would be to replicate the input and target data ε times, where each sample is annotated with a timestep $\varepsilon \times \Delta t$. Similarly, the inhibitory value would also be encoded in the same way for the training/testing periods and repeated ε times. However, this results in a substantial increase in memory allocation.

To implement epoch-based training across ε epochs without replicating the data, we designed custom, time-dependent functions that utilise the modulo operation to loop through

the original dataset. As mentioned earlier, the model is fed in the encoded values of $X\prime$ and $Y\prime$ to its input node N_{I} and the target node N_{T} respectively.

The values from the each of the nodes are retrieved through custom time-dependent functions, $f_{X\prime}(t)$ and $f_{Y\prime}(t)$, which are set in the nodes internally. Both functions rely on a single, shared mapping, $i(t)$, which converts the continuous simulation time t into a discrete index i in each of the datasets. This ensures the dataset is presented repetitively in sequence. This mapping is defined by the following formula:

$$i(t) = \left\lfloor \frac{t}{\Delta t} \right\rfloor \bmod S$$

Where Δt refers to the duration of the sample being presented to the model, and S represents the total number samples in the datasets in $X\prime$ or $Y\prime$. The final input and target signals presented to the input node N_{I} and target node N_{T} are then represented by indexing the respective datasets, where $f_{X\prime}(t) = X\prime[i(t)]$ and $f_{Y\prime}(t) = Y\prime[i(t)]$.

The inhibition signal function $\gamma(t)$ in the inhibition node is enabled in each epoch only after the training period (τ_{train}) ends during the designated testing duration that starts immediately after until the end of the simulation epoch (τ_{total}).

$$\gamma(t) = \begin{cases} 0.0 & \text{if } t \bmod \tau_{\text{total}} \leq \tau_{\text{train}} \\ W_{\mathrm{I}} & \text{if } t \bmod \tau_{\text{total}} > \tau_{\text{train}} \end{cases}$$

4.3 Spiking Neural Network to Multi-Layer Perceptron Translation

After the training phase for an epoch, the SNN model is first converted into the Neural Interchange Format (NIF) format using the `NengoNIFConverter` presented in Section 2.4.6. By doing so, we can produce a standardised, intermediary representation of the SNN model, which is decoupled from the simulator-specific data structures of Nengo whilst maintaining the internal information about the model and its parameters such as weights and parameters.

At this stage, we restrict our SNNs to be non-cyclic by removing the components used for training the model – i.e., the target and inhibition nodes as well as the error ensemble – which create a feedback loop (see: Figure 2.7) as the feedforward network topology of MLPs do not support the cyclic connections. Once the SNN-MLP is generated, we don't expect to train it further, as that would make the model update its weights in a different way from the original SNN, and prevent it from being a viable representation of the SNN. Thus, even if the aforementioned components were redacted, the remaining components would be

sufficient to create an MLP that can represent the dynamics of the SNN. Therefore, we call the `NIF.simplify()` function (see: Section 2.4.6), passing in the input node and the probe connected to the E_M ensemble to remove all the components which are not connected to the input node and the final output probe.

This leaves the NIF data to only retain information about the input node, the chain of M ensembles, the final output probe, and the associated connection weight matrices. Table 2.2 shows the formal mapping of the SNN components' symbolic notation from the Nengo framework to their respective symbolic notation within the Neural Information Flow (NIF) representation. It should be noted, that when training with longer epochs, to monitor the performance of the model at each epoch, a new MLP model is created using the weights learned during the training phases up until that epoch to ensure that the SNN-MLP maintains a faithful representation of the SNN over time.

4.3.1 Converting the Spiking Neural Network to a Keras Multi-Layer Perceptron

Once the SNN is represented with NIF, we can then map the relevant parameters such as weights, gains and biases to the corresponding components of the MLP. For this, we use another NIF converter `NIFKerasConverter` we developed to create a simple MLP model with the Keras library in Python. The process begins by iterating through the structural SNN components stored in the NIF model.

Figure 4.4 shows an example weight mapping after converting the simplified SNN into an SNN-MLP. For simplicity, the ensembles are represented together as the gain values for each ensemble.

Input Layer The input `NIF.Node` of the original SNN is converted into a `keras.Input` layer (labelled as $\mathbf{L}_\mathrm{I}$ in the example figure), initialised with $d_{X\prime}$ neurons, which is equivalent to the number of input features in $X\prime$. Note that, after running the `NIF.simplify()` call, there will only ever be one node present within the SNN, which is automatically selected as the input node.

Intermediary (Hidden) Layers The hidden layers in the MLP are built sequentially following the connection path found from the input node to the M-th ensemble. For every `NIF.Ensemble` E_k^{NIF}, a `keras.Dense` layer $\mathbf{L}_k$ is generated. Each of the neurons from the E_k^{NIF} is translated into a neuron in the $\mathbf{L}_k$ layer, and the bias of the `NIF.Ensemble` ($b_{\mathrm{E}_k^{NIF}}$) is also transferred directly over to the $\mathbf{L}_k$ layer. Additionally, to ensure that the SNN-MLP is

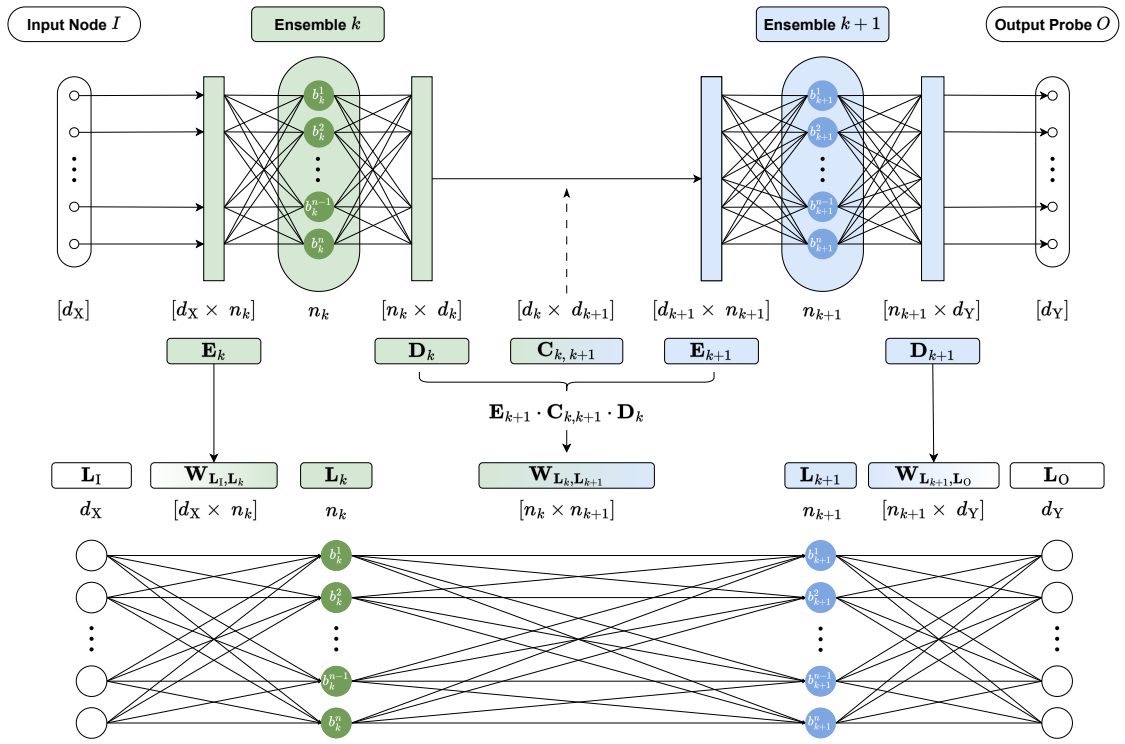


Fig. 4.4 SNN to MLP weight translation.

able to mimic the internal mechanism of the SNN, a custom activation function is applied to every neuron in the hidden layers. More details on the custom activation function will be discussed in Section 4.3.2. Each new `keras.Dense` layer is connected to the output of the previously processed layer, which gives it the expected input shape of $n_{E_{k-1}}$, and incrementally forms the sequential architecture.

Connection Weights In order for the MLP to represent the learnt behaviour of the SNN, the weights of the SNN need to be transferred over. To achieve this, we iterate through all the connections of the SNN to calculate and apply the weights for each keras layer in a second pass. An MLP contains a set of weights **W**and a set of biases **B**, where the bias is mapped to each neuron individually. The bias is self contained in every layer, with a size corresponding to the number of neurons within the layer. The weight matrix for a layer $\mathbf{L}_k$ connecting to its subsequent layer $\mathbf{L}_{k+1}$ has the shape $[n_{\mathbf{L}_k} \times n_{\mathbf{L}_{k+1}}]$.

In contrast, SNNs do not have a direct weight matrix connecting between ensembles. Instead, signals between two sequential ensembles (E_k and E_{k+1}) are translated by first decoding the neural activity of the source ensemble (E_k) into a high-level vector, and then encoding that vector back into the input current for the neurons of the destination ensemble (E_{k+1}).

For example, in Figure 4.4, the weight matrix of the connection (C) between ensembles E_k and E_{k+1} (labelled as $C_{k,k+1}$ in the figure) has a shape of $[d_k \times d_{k+1}]$, where d_k and d_{k+1} represent the dimensions for each of the ensembles. This matrix specifies the linear function that maps the decoded output of E_k to the input of E_{k+1}. Ensemble k provides its decoders $\mathbf{D}_{E_k}$ (shape $[n_{E_k} \times d_{E_k}]$), which map its neural activity to its decoded output dimension. Ensemble $k+1$ provides its encoder $\mathbf{E}_{k+1}$ (shape $[d_{E_{k+1}} \times n_{E_{k+1}}]$) and gain g_{k+1}, which map its input vector back into current signals for its neurons.

To assign weights to the MLP, the functional connection between ensembles must first be transformed into a single weight matrix that maps neuron activity to scalar values. For a connection between ensembles E_k and E_{k+1}, the associated weight matrix $\mathbf{W}$for the SNN-MLP (labelled as $\mathbf{W}_{\mathbf{L}_k,\mathbf{L}_{k+1}}$ in the figure) is calculated by multiplying the connection weight matrix $C_{k,k+1}$ with the encoders and decoders of the connected ensembles:

$$\mathbf{W} = \mathbf{E}_{E_{k+1}} \cdot C_{k,k+1} \cdot \mathbf{D}_{E_k}$$

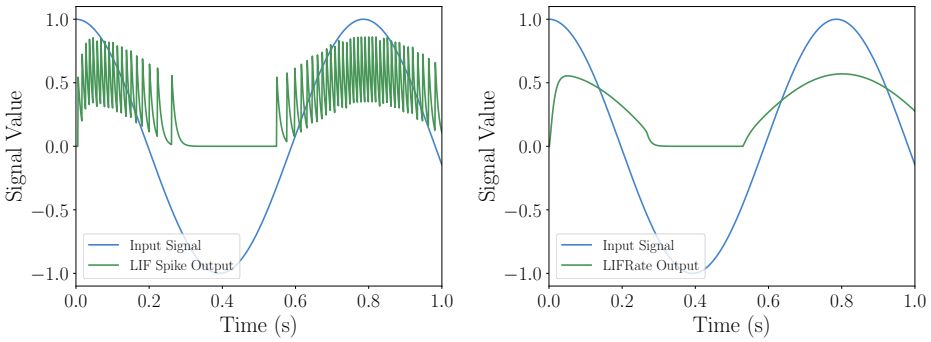
Which maps the activity of the $n_{\mathbf{L}_k}$ neurons in $\mathbf{L}_k$ to the $n_{\mathbf{L}_k}$ neurons in $\mathbf{L}_k$ of the SNN-MLP.

After, the transformed weight matrix $\mathbf{W}'$ is scaled using the gain of the $k+1$ ensemble, (see: Section 2.4.2) then set as the kernel of the `keras.Dense` layer corresponding to $\mathbf{L}_{k+1}$. The bias vector for the `keras.Dense` layer's neurons is applied directly using the biological bias defined within the corresponding SNN ensemble.

Output Layer Finally, an additional `keras.Dense` layer ($\mathbf{L}_Y$) is generated to represent the MLP's output decision Y_{pred}. The connection between the last ensemble E_M and the output probe P_M is used to create a final, non-activated output `keras.Dense` layer. The final ensemble's dimensionality d_{E_M} is used to set the number of neurons in this additional layer, which are all initialised with a bias of 0. The weights of the final layer are initialised directly using the connection's decoding weights $\mathbf{D}_{E_M}$, and the output value from this layer represents the predicted output value for the SNN-MLP (Y_{pred}). Depending on the dimensionality of the target label dataset, the predicted output value can be interpreted in two ways. In the case where $d_{N_\text{T}} = 1$ for binary classification tasks like the Distributed Denial-of-Service (DDoS) case study, the classification can be decided by a rounded value using an intermediary threshold. Otherwise, if $d_{N_\text{T}} > 1$ like in the depression case study, the final output can simply be found by retrieving the argmax of the output vector.

4.3.2 Activation Function

As mentioned earlier, a custom Leaky Integrate-and-Fire Rate (LIFRate) activation function (Listing 4.1) was defined to implement the spiking behaviour into all the neurons in the hidden layers of the SNN-MLP model. The activation function is derived from the spiking



(a) (b)

Fig. 2.6 Comparison of Leaky Integrate-and-Fire and Leaky Integrate-and-Fire Rate Neuron Responses to Sinusodial Input (reproduced from page 12)

behaviour from each of the SNN neurons' tuning curve (Eliasmith and Anderson, 2003) and uses the refactored firing rate Equation (2.2) from Section 2.3.2.

$$f(I) = \begin{cases} 0, & \text{if } I \leq V \\ \left(\tau_{ref} - \tau_{RC} \log\left(1 - \frac{V}{IR}\right)\right)^{-1} & \text{otherwise} \end{cases} \tag{4.1}$$

```
def lif_activation(x):
    dx = x-1 # threshold at 1.0

    # LIFRate activation function
    amplitude = 1
    tau_rc = 0.02
    tau_ref = 0.002
    fr = amplitude / (tau_ref + tau_rc * tf.math.log1p(1.0 / dx))

    # If dx leq 0, return 0.0 otherwise the LifRate activation value
    return K.switch(dx>0, fr, 0.0)
```

Listing 4.1 Custom Leaky Integrate-and-Fire (LIF) activation function implemented for the translated MLP

This enables the MLP to approximate the SNN's spiking dynamics effectively. Although the LIFRate neuron discards the explicit temporal vector of spiking activity, the temporal

sequence of spikes is effectively compressed into an average rate that preserves the essential nonlinear transfer function of the spiking model, as discussed in Section 2.3.2. Thus, this scalar representation can be used with the MLP as a substitute for the spiking behaviour of the original SNN model.

4.3.3 Applying Explainable Artificial Intelligence to the Spiking Neural Network-based Multi-Layer Perceptron

Once the SNN-MLP is constructed from the pre-trained SNN, SHAP can be applied to interpret its predictions based on the testing dataset. The SHAP explainer uses the model's predictions along with a baseline (expected) output to compute feature attribution scores for each input sample. These scores indicate the extent to which each feature influenced the predicted class. The resulting explanations can then provide interpretable insights into the surrogate model's decision-making process and, by extension, offer an interpretable approximation of the original SNN.

Chapter 5

Results: Single-Dimensional Spiking Neural Network Classification

To confirm the effectiveness of the proposed method for converting a single-dimensional Spiking Neural Network (SNN) (where all ensembles in the SNN only have a dimension size $d_E = 1$) into a Multi-Layer Perceptron (MLP), we will be utilising a case study focused on Distributed Denial-of-Service (DDoS) attack classification. The results from this case study is presented in three parts. First, we outline the background for this case study, and explain the dataset used in Section 5.1. Next, we outline the preprocessing methods, the model configuration parameters as well as the metrics to be used for evaluating the SNN model performance in Section 5.2. Then, we present the results using the different SNN configurations in Section 5.3, and discuss the impact of key parameters (learning rate, timestep, threshold, and learning rule) on the model accuracy and stability. Finally, using the most stable SNN configuration, we compare the Shapley Additive ExPlanation (SHAP)-based explanations generated across a benchmark MLP (Wei et al., 2023), a rebuilt MLP, and the converted SNN-based MLP (SNN-MLP) to evaluate the quality and consistency of feature attributions in Section 5.4.

5.1 Background

DDoS attacks are a malicious attempt to disrupt the normal traffic of a targeted server, service, or network by overwhelming the target or its surrounding infrastructure with a flood of internet traffic to reduce the availability of a system. The attacks are generally aimed towards using the resources such as bandwith or memory (Batchu and Seetha, 2022). Modern DDoS strategies are typically categorised into volumetric attacks, which focus on signals

saturation; protocol attacks, which exploit vulnerabilities in network layers; and application layer attacks, which target specific server functions to deplete application-specific resources (Raghupathi and Teja, 2024).

In response to these threats, traditional Artificial Neural Networks (ANNs), have been established as the foundation of modern Intrusion Detection Systems (IDSs). These models are typically employed as supervised classifiers that distinguish between 'Benign' and 'Attack' traffic based on extracted network flow features (Javeed et al., 2024). This process involves grouping raw network packets into flows and calculating statistical metrics, such as packet counts, flow durations, and mean packet sizes. The primary challenge for these systems is the requirement for immediate reactions to attack traffic. As large-scale volumetric attacks can compromise a network within seconds, detection must be rapid to allow for mitigation before total service failure occurs. However, this speed must be balanced with high reliability to avoid false positives that could inadvertently block legitimate traffic.

A prominent benchmark used to evaluate these detection systems is the CICDDoS2019 dataset, developed by Sharafaldin et al. (2019) from the Canadian Institute for Cybersecurity. This dataset is widely utilised in network security research because it encompasses a diverse range of contemporary reflection and exploitation-based attacks, such as DNS, LDAP and MSSQL reflection attacks. These vectors reflect real-world conditions where protocols are leveraged to amplify traffic volume against a target. While various machine learning and deep learning techniques have achieved high accuracy using this dataset, they are almost always restricted due to the usage of black box models. The opaque nature of these architectures makes it difficult for network administrators to understand the underlying logic behind a classification, which is a significant drawback in high-stakes security environments. Recent work by Wei et al. addressed this by applying SHAP to traditional MLPs, achieving an accuracy of 99% while providing feature-based explanations for the detections. This previous study establishes the baseline for our research. For the SNN-MLP translation to be considered a viable Explainable Artificial Intelligence (XAI) method, its generated explanations must be validatable. Thus, if the explanations generated the proposed SNN-MLP aligns with the baseline MLP used in the prior study, it can confirm that the SNN model has learned the relationships between the input and target data correctly, and that the explanations for the SNN-MLP are accurate by association, even if the models are different.

5.2 Experimental Setup

To maintain similarity with the previous approach (Wei et al., 2023), the paper follows the same procedure for the feature extraction, data balancing, cleaning, label encoding, and

normalisation methods described in the article and are described in the rest of this section. The complete source code and the SNN configurations files in this section are available at https://github.com/janenotjung-hue/xaisnn/tree/main. The dataset used for this experiment was developed and made publicly available by Sharafaldin et al. (2019).

5.2.1 Evaluation Dataset

The dataset includes large samples for attack (represented as 0) and benign (represented as 1) traffic which have been captured in a modern simulation to provide a realistic representation of network traffic flow. Data was collected across two days, with 50 million samples collected on the first day, and $>$ 20 million samples collected on the second day. The dataset contains 12 different attack types (DNS, LDAP, NetBIOS, SNMP, MSSQL, SSDP, UDP, UDP-Lag, SYN, TFTP, WebDDoS, PortMap), and every sample is recorded along with 88 additional features (Sharafaldin et al., 2019). For this research, the scope of the classification will be reduced to classification between benign and DNS attack traffic, which were captured on the second day from 10:52AM to 11:05AM.

Feature Selection Feature extraction was performed by Wei et al. (2023) using a combination of Permutation importance SHAP, Extreme Gradient Boosting (XGBoost) and Random Forest (RF) with SHAP analysis to find the top 20 most important features for classification of DNS attacks. Following their suggestion, the top 20 important features from the dataset were extracted to be used for the model. These features are as follows: *Protocol, URG Flag Count, Flow Duration, Init Win bytes forward, Fwd Packet Length Min, Min Packet Length, Fwd Packets/s, Max Packet Length, Flow IAT Min, Fwd Packet Length Max, Average Packet Size, Bwd Packets/s, Inbound, Init Win bytes backward, ACK Flag Count, Total Fwd Packets, Total Backward Packets, Flow IAT Mean, Packet Length Std, Bwd IAT Total*.

Data Balancing In the reduced dataset there was a significant imbalance between the target labels, with 507608 samples labelled as attack (DNS) traffic and only 3374 benign samples. In the original study, to maintain fairness, all benign traffic was first extracted, and combined with a random sampling of 0.1% from the attack traffic. However, for this study, only 1000 data points were selected to have a balanced dataset and act as a preliminary study for XAI with SNNs. This was in consideration of the time required to train and test the model with different parameter configurations, as the simulation is required to run $\Delta t \times$ the number of samples within the dataset (2000) as each sample is presented to the model for Δt seconds. While this is an arbitrary number, the duration of the simulation increases significantly with larger Δtvalues, and also requires more resources to store the simulation

information. Therefore, for this case study we used a reduced sample size containing 1000 samples for each class – future work would involve enhancing the scalability of the trainable SNN.

We then cleaned the dataset to remove any missing or invalid values, which left us with 891 samples labelled as benign traffic and 900 samples for attack traffic. Although this created a slight imbalance of classes, rather than further reducing the dataset to achieve perfectly equal distributions, this marginal imbalance was preserved to evaluate the SNN's performance in a more representative environment. This therefore allowed for a preliminary observation of how class disparity interacts with the chosen spike encoding schemes, if at all.

Label Encoding In the original approach from Wei et al., the categorical target values "BENIGN" and "ATTACK" were substituted with numeric values so that they would be interpretable by the model. The benign traffic was represented as 1, and attack traffic was encoded as 0. However, SNNs are generally configured with a radius value ($r = 1$) which means that the decoded signal output can range anywhere between -1 and 1, rather than just between 0 and 1. Having the data encoded to just 0 and 1 only utilises half of the network's representable space, and may have a negative impact the learning dynamics of the SNN. In consideration of the dynamic range of the SNN to output negative values but also stay faithful to the original approach, 4 distinct datasets were created, which are summarised in Table 5.1. Note that in all cases, the datasets were split into train/test data with a 90:10 ratio. By testing all four combinations we can systematically determine the most robust and efficient label encoding scheme for the proposed SNN architecture, which hypothetically should then be able to correctly replicate the results of the benchmark model. Achieving superior performance alongside consistent explanations across different encodings would also demonstrate that the SNN is capturing the true underlying patterns of the dataset rather than merely adapting to the numerical range of the targets, demonstrating even more flexibility.

Table 5.1 Summary of data encoding variants.

Dataset	Benign (Sample #)	Attack (Samples #)
1 (used by Wei et al.)	1 (891)	0 (900)
2	0 (900)	1 (891)
3	1 (891)	-1 (900)
4	-1 (900)	1 (891)

Data Normalisation The radius of SNN ensembles represent the optimal range value for SNN ensembles to interpret. When met with values beyond the threshold (e.g., > 1 or < -1),

although it doesn't entirely break the model it does cause the neurons to fire excessively in response, which ends up saturating them so that they cannot dynamically process smaller features. In the original CICDDoS2019 dataset, several features such as *Min Packet Length* and *URG Flag Count* had a high variance between their minimum and maximum values.

To address this, MinMax-based normalisation was applied to scale the features to reduce the range of the values to 0-1 using Equation (5.1), where Z_i represents all normalised numeric values ranging between 0 and 1, and *max* and *min* represent the maximum and minimum values from all data points.

$$Z_i = \frac{Z_i - min}{max - min} \tag{5.1}$$

The same approach was also used in the previous study as well.

5.2.2 Model Configuration

As mentioned in the previous chapter, the SNN model was built using the Nengo library in Python, with two ensembles containing 20 (to match the number of features in the input dataset) and 5 Leaky Integrate-and-Fire (LIF) neurons respectively. Although this was slightly different to the MLP used by Wei et al. (2023) which had 3 hidden layers (23, 15, 10), considering that the models have different processing methods and that SNNs are expected to be capable of achieving the same results as MLP with less neurons, we determined that having a different model topology would not have a major impact. Thus, we decided to try a model with a reduced amount of neurons compared to the benchmark MLP to verify this capability. A combination of parameters were tested with the SNN while maintaining the same ensemble sizes, to determine which configuration was the most suitable to use for SNNs with classification tasks. Details about the parameters are shown in the Table 5.2. The SNN model was set with static seeds for all components to maintain a controlled experiment. In all cases, the SNN model was run with a 90:10 split of training and testing data. After the SNN model was finished with training, the SNN-MLP model was built following the same structure as the SNN model.

Table 5.2 SNN model parameters.

Parameter	Values
Learning rate (s)	$a \times 10^{-x}$, where $a = 1, 2, \ldots, 5$ and $x = 2, 3, \ldots, 6$
Timestep (s)	$[0.1, 0.2, \ldots, 1]$ and $[1, 2, \ldots, 10]$
Learning Rule	Prescribed Error-Sensitivity (PES) and Recursive Least Squares (RLS)

5.2.3 Evaluation Metrics

To evaluate the performance of the SNN model, the following metrics will be used: the extended evaluation metrics True Positive (TP), True Negative (TN), False Positive (FP), False Negative (FN) as well as the overall accuracy. TP refers to an attack sample correctly classified as an attack, while TN denotes a benign sample correctly classified as benign. Conversely, FP represents a benign sample misclassified as an attack, and FN represents an attack sample incorrectly classified as benign. The accuracy is a measure of the total number of correct classifications in the entire sample group (Wei et al., 2021).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.2}$$

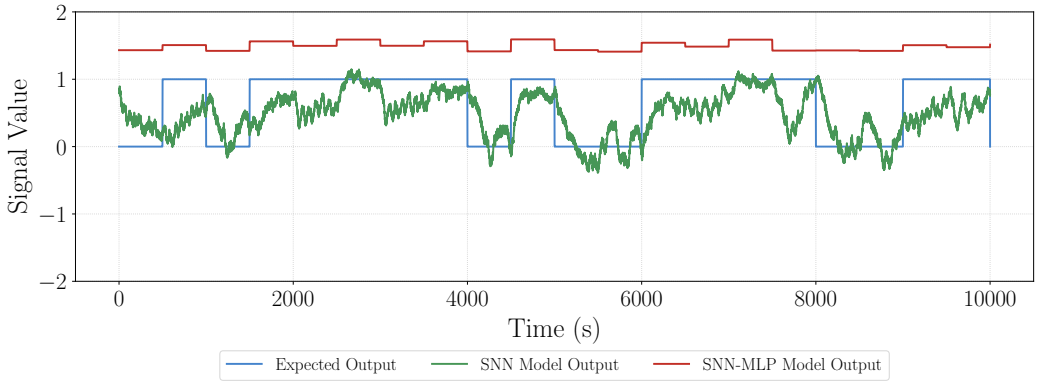
5.3 Comparing Spiking Neural Network Learning Rules

To ensure the SNN was correctly completing classification tasks and confirm that the translated SNN-MLP was accurately mimicking the behaviour of the SNN, some initial tests using different combinations defined in Section 5.2.2. This section presents the initial performance results of the SNN-MLP and identifies the best combination of parameters to proceed with for further explanation.

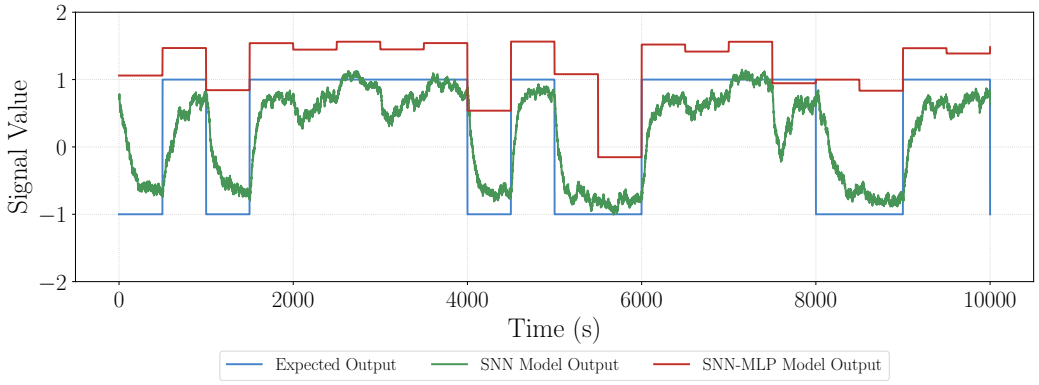
In conventional classifiers, a fixed decision threshold of 0.5 is typically applied for binary classification, as the target values are expected to be 0 or 1. However, for SNNs, the output depends on spiking activity and neuron dynamics, which enables the model to produce continuous values that may be outside the ranges, resulting in the optimal threshold shifting and not necessarily aligning with the median threshold of 0.5. This behaviour is also more likely to occur when the label values are encoded as the ranges from the radius values – such as benign = 1, attack = -1or benign = -1, attack = 1(datasets 3 and 4 respectively).

For example, in Figures 5.1 and 5.2 the outputs of the testing data on datasets 1 and 3 for the SNN and its surrogate SNN-MLP models are traced along with the expected target value using Prescribed Error-Sensitivity (PES) and Recursive Least Squares (RLS). Between datasets 1 and 3, the benign label is the same (1), but the attack label switches from 0 to -1 respectively. From the traces, we can confirm that the SNN has been successfully classifying the test data, and that the SNN-MLPs are generally following the behaviour of the SNN, and that the SNN almost always stays within the threshold set by the encoded values.

With PES, regardless of the data encoding method there is a certain amount of positive bias for the SNN-MLP, and all the predicted values exceed the range of the neurons, sitting at (≈ 1.5) with dataset 1 (Figure 5.1a). The offset between the SNN and SNN-MLP is likely due to the bias learned by the SNN during training, which was carried into the SNN-MLP



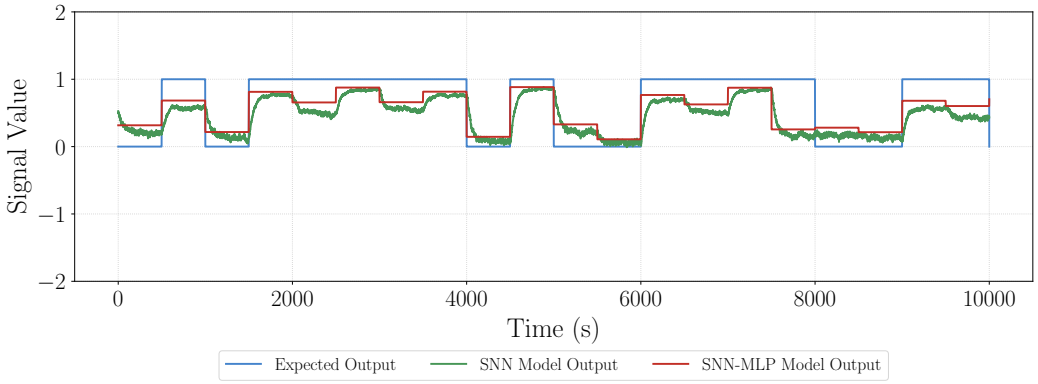
(a) Dataset 1 (benign = 1, attack = 0)



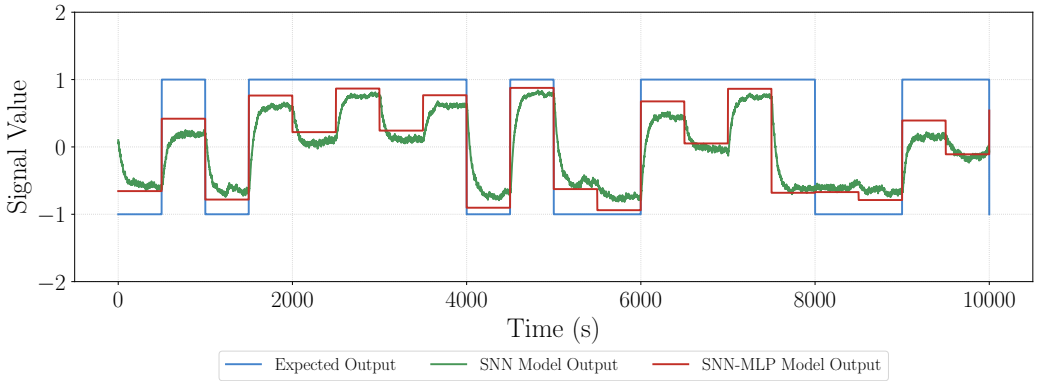
(b) Dataset 3 (benign = 1, attack = -1)

Fig. 5.1 Initial traces of the SNN decoded output and the SNN-MLP output against the expected target output using PES with the test data ($\eta = 0.001$ and $\Delta t = 1.0$).

when it was generated. However, even with the slight offset the SNN-MLP still follows the general trend of the SNN, increasing and decreasing in line with the SNN. With dataset 3 (Figure 5.1b), the output of the SNN-MLP does still stay closer to 1, and overall the points of increase/decrease match that of dataset 1. However, the magnitude of changes are much larger with dataset 3 and less consistent. For example, at 5000-6000 seconds, in Figure 5.1a the output of the SNN-MLP drops twice, and then again at around 7500s to a similar value. However, in Figure 5.1b, although the drops are at the same time point, the drop is much larger at 5000-6000s compared to the 7500s point. This could be due to the target labels in dataset 3 having a larger range, which likely impacted the weights of the SNN during training, as we can see that the traces of the SNN model on the testing data do not align at some points (e.g., at times 4000-6000s, 8500-9000) between the two figures. With RLS, the SNN-MLP follows the SNN, and we can see that there is less offset between the two models.



(a) Dataset 1 (benign = 1, attack = 0)



(b) Dataset 3 (benign = 1, attack = -1)

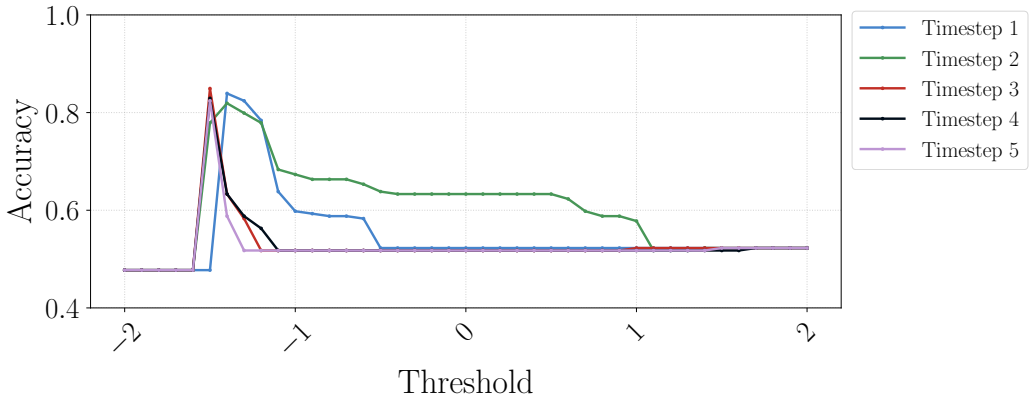
Fig. 5.2 Initial traces of the SNN decoded output and the SNN-MLP output against the expected target output using RLS with the test data ($\eta = 0.001$ and $\Delta t = 1.0$).

The trend of the SNN outputs from datasets 1 (Figure 5.1a) and 3 (Figure 5.1a) also align more closely with each other as well, although again, with dataset 3 the range of the output values are wider- between -1 and 1 due to the encoded target values.

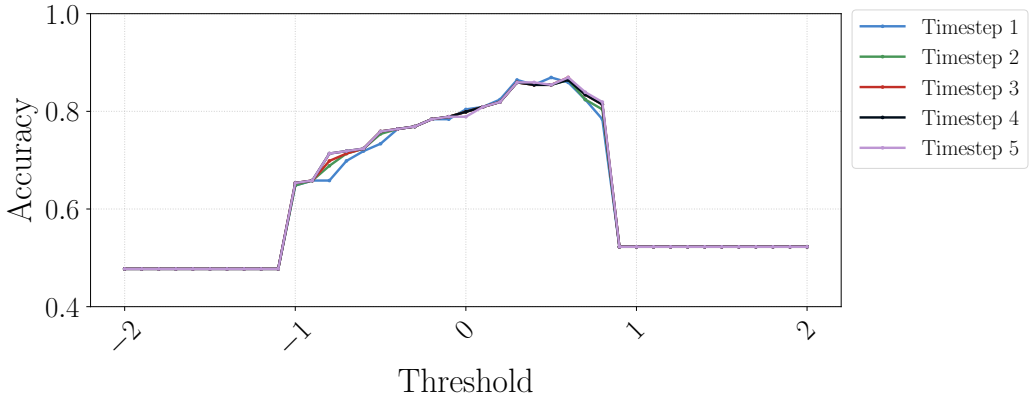
Given that the SNN outputs are continuous and can exceed the neuron's set representation range of 1 to -1, and that different encodings affect the magnitude of these outputs, we also examined the performance across a wider range of classification thresholds for a more thorough evaluation of the model's classification performances. To account for the possibility that the particular configuration of the learning rate and timestep were causing irregular responses, we tested the SNN-MLP models across a small range of timesteps ($\Delta t = [1, 2, \ldots, 5]$) with a static learning rate ($\eta = 0.001$) as the control variable.

We present the results with using Dataset 1 (benign = 1, attack = 0) as an example to explain the Insensitivity Factor (IF) values– the results for the rest of the encoding methods

can be found in Appendix A. Figure 5.3 shows the accuracy of the SNN models trained PES and RLS learning rules as well as different and timesteps (Δt) across a range of threshold values from -2 to 2.



(a) PES accuracies across thresholds



(b) RLS accuracies across thresholds

Fig. 5.3 Accuracy variations across threshold ranges $-2.0-2.0$ using PES and RLS learning rules across $\Delta t = [1,2,\ldots,5]$, and $\eta = 0.001$ for both models.

With PES, having a longer timestep shifts the curve towards the right, with the peak accuracy reaching 0.84 when the threshold is at 1.6, at a timestep of 5 seconds. The curve for the RLS model is consistent regardless of the timestep value used, where for all the timesteps the peak accuracy is reached (0.87) when the threshold value is around 0.6. From this, we can confirm that just having a static threshold of 0.5 for all the different encoding methods and learning rules would not represent the SNN-MLP model's decisions correctly, especially when using PES.

However, simply reporting the peak accuracy attained from the optimal threshold is insufficient for evaluating the model's quality, as this approach ignores the model's behaviour

for all situations. Having a model that performs at a peak accuracy but only at a certain point makes it unreliable, especially when there is no concrete method to set the threshold objectively. Ideally, the model should have stable performance across a wide range of thresholds, as it would mean that the model is more robust to fluctuations in the output values (i.e., values beyond the range set by the neuron radius).

Therefore, we introduce a new metric IF to measure the sensitivity of the model accuracy to variations in the threshold. The IF is a value used to measure how much impact the threshold value has on the accuracy. A higher IF indicates that a wider range of thresholds can be used with minimal impact, and therefore the model is more robust. The IF is calculated as follows:

$$IF = \delta \times \sum_{x=0}^{\lceil \frac{b-a}{\delta} \rceil} \frac{f(\delta \times x + a) + f(\delta \times (x+1) + a)}{2} \tag{5.3}$$

where a and b represent the minimum and maximum threshold values considered, x represents the current threshold, δ is the step size between thresholds, and $f(\delta)$ denotes the accuracy at a given threshold.

It should be noted that the IF score also needs to have a baseline established to act as a reference point for the minimal performance. For the baseline, we consider a model accuracy of 0.5 across the threshold range. As this is a binary classification task, having a model accuracy of 0.5 indicates the average accuracy of the model is on the same level as guessing randomly. Therefore, it can be used as a reference point to indicate the model has essentially learned nothing. In this example, a baseline IF score of 2.0 corresponds to a model that achieves a flat accuracy of 0.5 across the entire threshold range (i.e., from $a = -2$ to $b = 2$). Any IF score above therefore indicates that the model's accuracy, on average, exceeds 0.5 over the total threshold range. In contrast, an IF score of 2.0 or less implies the model is performing no better than random guessing across the entire threshold spectrum. Therefore, a higher IF value confirms that the model is more robust, as it maintains higher accuracy across a wider variety of classification thresholds.

Using the range as displayed in the figures of Figure 5.3 (-2 to 2), the IF values from the PES model for each of the timesteps (Δt) are: $[2.14, 2.10, 2.10, 2.11, 2.11]$, which align with the width of the curves shown in Figure 5.3a where using timestep 1 produces the widest peak, followed by timesteps 5 and 4. With RLS, the IF values are more consistent: $[2.22, 2.22, 2.23, 2.22, 2.23]$, as the curves are roughly the same. Notably, when the threshold is at 0.1 the accuracy for timesteps 5s and 3s are slightly higher than the rest, which reflects the higher IF values in the list. From this, we can see that both PES and RLS learning rules were effective for the model to learn about the datasets using varying thresholds, with

RLS displaying slightly higher and more consistent IF scores and thus suggesting stronger threshold robustness.

Since the range of the SNN output and the optimal threshold shift between different configurations (as noted by the different peak locations for PES) and also learning rules, relying only on maximum accuracy is misleading. Therefore, we continue to examine the IF score in subsequent experiments, as it allows us to consistently compare the overall stability and reliability of PES and RLS performance across all four datasets.

The following sections further evaluate the results of the model configured with different combinations of the parameters previously defined. Considering that the initial models' performance demonstrated the impact of the threshold, two additional evaluation cases are considered: the performance of the models with the average of the two target label values used as the threshold, and the performance of the model using the threshold with the highest accuracy.

5.3.1 Evaluating Impact of Learning Rates & Timesteps

To optimise the performance of the SNN-MLP, appropriate parameters must be set for the baseline SNN. For the preliminary case study, the two main parameters explored for model optimisation were the learning rate (η) and the timestep (Δt). For SNNs, there are no globally optimal parameter values – instead, the optimal parameter settings depend on the specific SNN structure, training dataset as well as the learning rule used. Thus, it was necessary to conduct a systematic evaluation. The following section presents the performance results of the SNN-MLPs models which were trained with the PES and RLS learning rules, and examines the impact of using different learning rates and timesteps to configure the model.

Impact of the learning Rate and timesteps on the Prescribed Error-Sensitivity Learning Rule

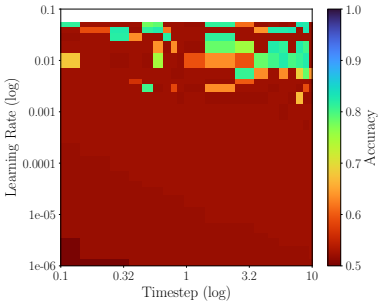
With the PES learning rule, there is no consistent pattern observable between the learning rate and timestep in terms of the accuracy. Figure 5.4 shows the accuracies of the SNN-MLP when tested with the 4 different datasets using the median threshold (0.5 for datasets 1 and 2, 0.0 for datasets 3 and 4). Between datasets 1 (Figure 5.3a) and 3 (Figure 5.3c) the only difference is the label for the attack traffic (1 and -1 respectively). As the upper threshold remains the same, we can observe a similar pattern in both figures, that shows a linear growth of accuracy in proportion to the learning rate and timestep. However, with dataset 3, there is also a peak in the top left corner of the graph, with the highest accuracy being 0.86 when learning rate is at 10^{-6} and timestep is 0.6. This contrasts with the results of dataset 1, which shows the lowest accuracy in the same region.

Meanwhile, with datasets 2 and 4 where the labels are reversed altogether, the accuracy is generally higher for a greater range of parameters, and interestingly the accuracy seems to be inversely proportional to the learning rate. For example, in figure (Figure 5.3b) the highest accuracy (0.86) is recorded when the timestep is 0.1 rather than the longer periods, and immediately after there is a dip in accuracy from when the timestep is between 0.2 and 2.0. This behaviour is replicated with dataset 2 (Figure 5.3d), but with a more gradual dip in the accuracy with larger learning rates and timesteps, as well as a much higher overall accuracy in the banded area.

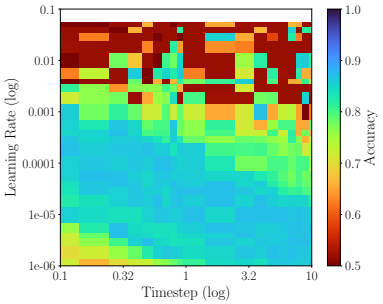
It can also be noted that datasets 1 and 2 seem to be the mirror versions of each other, and the same for 3 and 4. For example, the red ‘area‘ in the middle of dataset 3’s figure is the area of best performance for dataset 4, and vice versa. Figure 5.5 visualises the accuracy for the models using PES, but with the threshold that gave the best accuracy. Here, we can see the same diagonal band as the figures using the median thresholds, but a more consistent pattern between all the figures, where the accuracy decreases with either: smaller learning rates and short timesteps, or with larger learning rates and longer timesteps.

Impact of the learning Rate and timesteps on the Recursive Least Squares Learning Rule

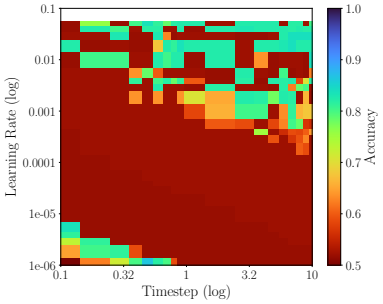
With RLS, the accuracy is more consistent across all the different datasets regardless of its parameters. When using the best threshold (Figure 5.7), there are almost identical results across all graphs, where the accuracy steadily increases while the learning rate is $< 10^{-4}$ and the timestep is below 1s, then after flattens out at around 0.85, with mild fluctuations depending on the configuration with datasets 1, 3 and 4.



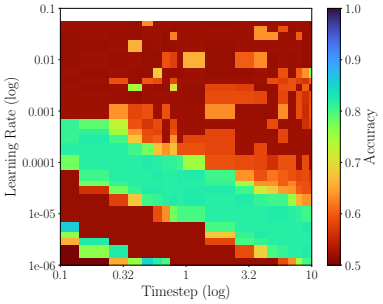
(a) Dataset 1 (benign = 1, attack = 0)



(b) Dataset 2 (benign = 0, attack = 1)

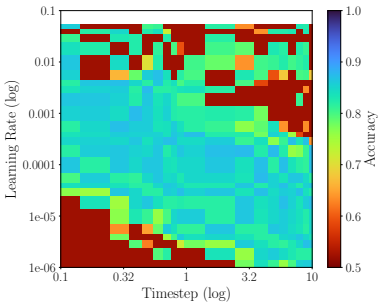


(c) Dataset 3 (benign = 1, attack = -1)

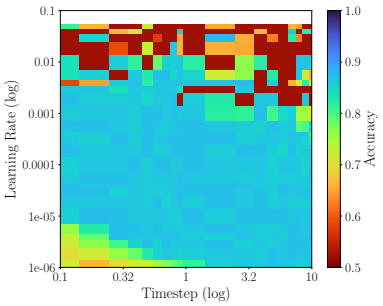


(d) Dataset 4 (benign = -1, attack = 1)

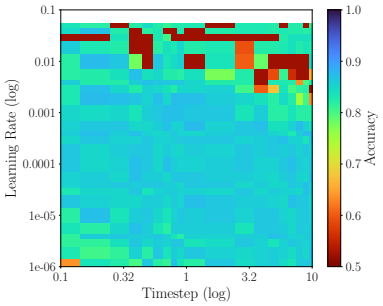
Fig. 5.4 Accuracy variations across all learning rates & timesteps with PES (Median).



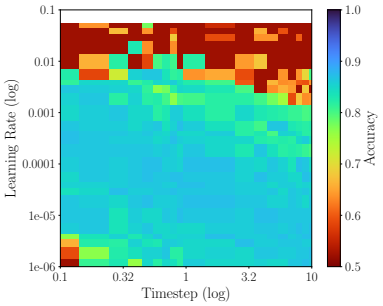
(a) Dataset 1 (benign = 1, attack = 0)



(b) Dataset 2 (benign = 0, attack = 1)

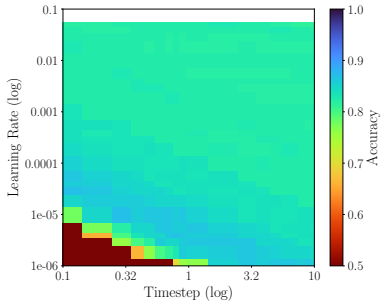


(c) Dataset 3 (benign = 1, attack = -1)

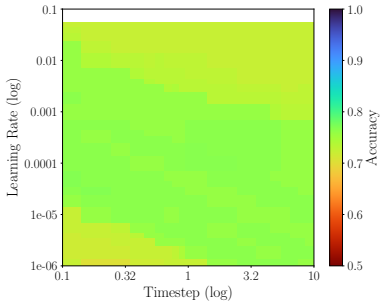


(d) Dataset 4 (benign = -1, attack = 1)

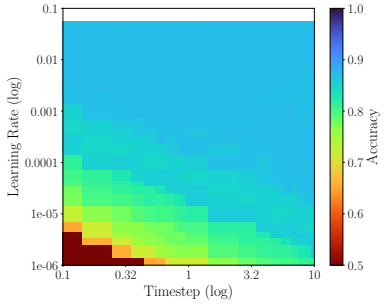
Fig. 5.5 Accuracy variations across all learning rates & timesteps with PES (Best Accuracy).



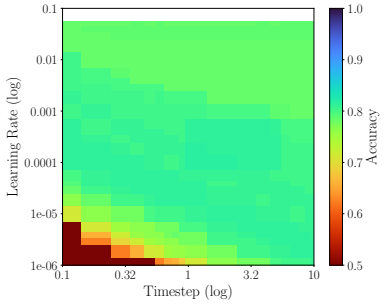
(a) Dataset 1 (benign = 1, attack = 0)



(b) Dataset 2 (benign = 0, attack = 1)

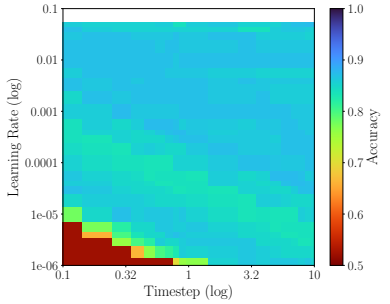


(c) Dataset 3 (benign = 1, attack = -1)

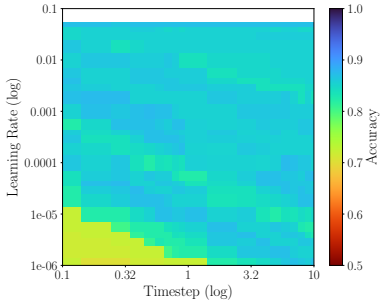


(d) Dataset 4 (benign = -1, attack = 1)

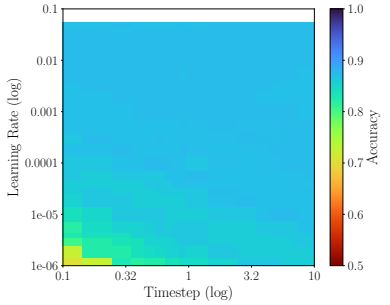
Fig. 5.6 Accuracy variations across all learning rates & timesteps with RLS (Median).



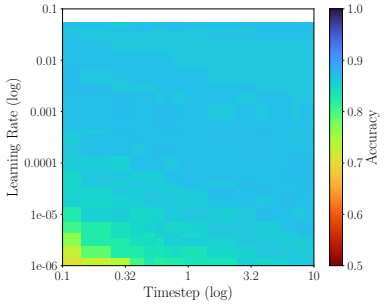
(a) Dataset 1 (benign = 1, attack = 0)



(b) Dataset 2 (benign = 0, attack = 1)

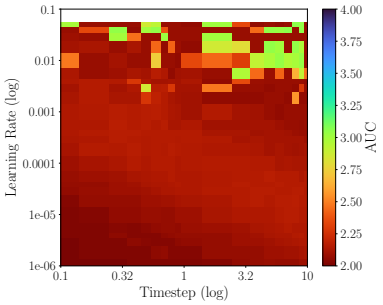


(c) Dataset 3 (benign = 1, attack = -1)

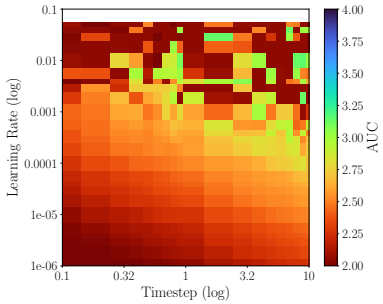


(d) Dataset 4 (benign = -1, attack = 1)

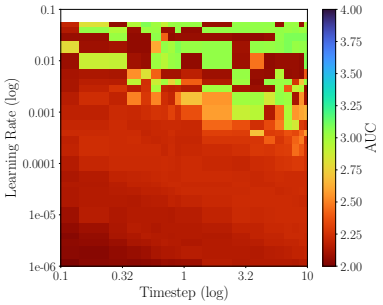
Fig. 5.7 Accuracy variations across all learning rates & timesteps with RLS (Best Accuracy).



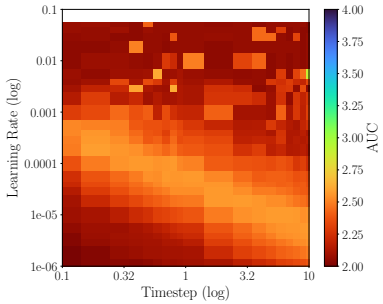
(a) Dataset 1 (benign = 1, attack = 0)



(b) Dataset 2 (benign = 0, attack = 1)



(c) Dataset 3 (benign = 1, attack = -1)



(d) Dataset 4 (benign = -1, attack = 1)

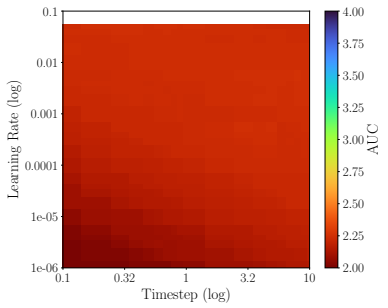
Fig. 5.8 IF variations across all learning rates & timesteps with PES.

Using the median threshold indicates that there is an optimal range for the learning rate and timestep values, and we can see similar banding patterns from the PES figures. For example, with dataset 1 (Figure 5.5a), the highest accuracies are recorded when the learning rates are low ($\leq 10^{-5}$) and the timestep is between 0.5-3 seconds. With datasets 2 (Figure 5.5b) and 4 (Figure 5.5d), the higher accuracies are generally recorded with lower learning rates and longer timesteps, and we can see a decay in the accuracy towards the bottom left and the top right of the heatmaps, similarly to the corresponding results using PES in figures (Figure 5.3b) and (Figure 5.3d). With dataset 3 (Figure 5.5c), there is a distinct linear relationship between learning rate and accuracy until when the learning rate is $< 10^{-3}$ and timestep is 0.1, then we see a small band with a slightly reducer accuracy, then once the learning rate is $> 10^{-3}$ the accuracy plateaus at around 0.85, regardless of the parameters.

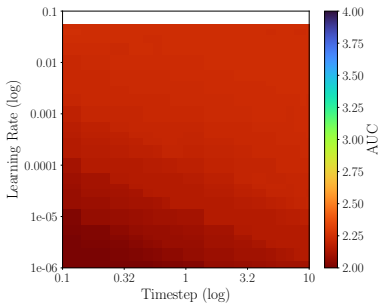
5.3.2 Evaluating the Label Encoding Methods

Figure 5.8 and Figure 5.9 show the IF scores across the different datasets and learning rules.

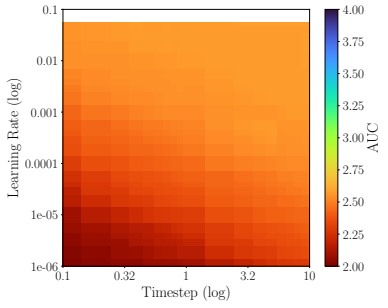
The PES models in Figure 5.8 shows a higher IF score (≈ 3) with larger learning rates and longer timesteps. Datasets 1 and 3 (Figures 5.8a and 5.8c) in particular show the same trend of having the highest IF scores at the top right corner, and gradually decreasing as the



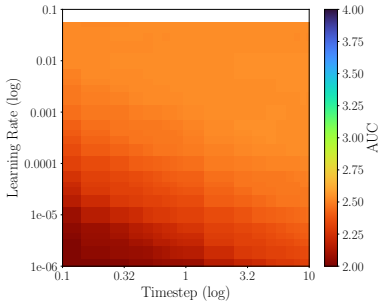
(a) Dataset 1 (benign = 1, attack = 0)



(b) Dataset 2 (benign = 0, attack = 1)



(c) Dataset 3 (benign = 1, attack = -1)



(d) Dataset 4 (benign = -1, attack = 1)

Fig. 5.9 IF variations across all learning rates & timesteps with RLS.

learning rates and timesteps decrease, although the decrease is more gradual with dataset 3. Notably, with datasets 1 and 3, despite the lower half of the heatmaps presenting the highest accuracies using the best threshold (Figures 5.4a and 5.4c), the accuracy with the median threshold (Figures 5.3a and 5.3c) and the IF scores for the same regions are quite low. This disparity indicates that the high accuracies with the best thresholds were acheived only at a single, highly sensitive ideal threshold, and therefore are not likely to be a reliable metric to assess the model. With dataset 2 (Figure 5.8b), the highest IF scores are recorded at a lower learning rate ($0.001 < \eta < 0.001$) and longer timesteps, and the IF scores decreasing in proportion to the learning rate and timestep. We can also see that the IF scores become more sporadic with larger learning rates, alternating between the highest scores ($IF = 3$ at $\eta \approx 0.1$ and $\Delta t = 2$) and lowest scores ($IF = 2$ at $\eta \approx 0.1$ and $\Delta t = 0.5$). Unlike the other datasets, in Figure 5.8d we can see that using dataset 4 consistently has a lower IF score, where the highest score is around 2.5. Interestingly however, the band going through Figure 5.8d aligns with that of Figure 5.3d, which suggests that using the median threshold in makes the model more robust. Similar observations can be made for datasets 1 and 3, where the top half of the heatmaps have the highest accuracies in Figure 5.4 at the same points as most robust points in Figure 5.8.

With RLS (Figure 5.9), the models again show a linear growth in IF in relation to the learning rates and timesteps, until it reaches a certain maximum IF value at around 2.25 for datasets 1 and 2 (Figures 5.9a and 5.9b) and 2.5 for datasets 3 and 4 (Figures 5.9c and 5.9d). As the output values of the RLS models have been observed to follow the target value much closely than PES, the smaller IF scores for datasets 1 and 2 are reasonable. As the range for the outputs trained with datasets 1 and 2 are smaller than the output ranges of datasets 3 and 4, if the same threshold ranges (-2 to 2) are used to measure the IF scores in both scenarios it is inevitable that the former case has a lower score. However, considering that the trend of the IF score aligns with the accuracy using the best threshold and the median threshold, we can assume that the RLS learning rule is relatively robust to threshold changes regardless of the configurations of the learning rates and the timesteps.

5.3.3 Discussion

For models trained with PES, performance was highly sensitive to the choice of learning rate and timestep. As shown in Figure 5.5, models tend to perform better with larger learning rates and timesteps, although this is only in cases where the best threshold is used instead of the median threshold. This result aligns with the theoretical locally greedy nature of the PES learning rule, which updates its weights dynamically to minimise the error between the output and the latest current target value. The learning rate determines the time window of the signal that PES considers when adjusting its weights Banerjee et al. (2024). As a result, the model becomes highly sensitive to sudden or large changes in the output signal. In contrast, models trained with RLS demonstrated significantly more consistent performance across all datasets and parameter configurations. As seen in Figure 5.6, even when using the median threshold, the accuracy remains relatively high and stable, and generally much more aligned with the best results as well. This indicates that RLS is less sensitive than PES to the changes in the learning rate and timestep overall, aside from a few specific instances.

The best configuration of the learning rates and timesteps are quite different depending on the learning rule. For PES, with a smaller learning rate, the model restricts its weight adjustments to a narrow, recent history of the error signal. This makes the model extremely cautious, and requires longer time to converge towards the target, which makes it difficult to understand longer patterns or quickly adapt to large errors. On the other hand, while the model considers a broader context with larger learning rates, it becomes more likely to overcorrect its weights in response to large errors from individual training examples, leading to overfitting if the sample values remain constant for a longer period of time (i.e., larger timesteps), and instability when the samples' values change frequently (i.e., shorter timesteps), as it will be endlessly trying to converge and never understand the full pattern. From Figures 5.5 and 5.7,

we can see that although the PES learning rule does occasionally have higher accuracy levels than with the RLS, they require specific parameter tuning to ensure that the model can align with the final target values. This makes it more disadvantageous for applicability with larger datasets as well as imbalanced datasets – as PES will only ever focus on the most frequent/recent values. RLS, however, remains robust to these shifts, maintaining high accuracy even with shorter timesteps. While a slight improvement is observed when using the best-performing threshold, the overall trend between the median and best thresholds remain the same, suggesting that RLS can perform well even with the default values.

The encoding method significantly influenced the learning dynamics, particularly for PES. From Figures 5.8 and 5.9, we can see that the models trained on Datasets 1 and 3 (encoded 0 to 1) generally underperformed compared to Datasets 2 and 4 (encoded -1 to 1). This confirms that restricting the target signal to 0 to 1 utilises only half of the SNN's representable space ($r = 1$). This limitation hinders the model's ability to generate a balanced error signal, as the network cannot effectively leverage negative decoded outputs to counteract overshooting. The higher performance of the encoding using the full radius range (Datasets 2 and 4) demonstrates that a faithful representation of the SNN's dynamic range is essential for stable weight convergence.

In conclusion, from this section we can confirm that the consistent performance of RLS makes it a more reliable choice for real-world applications, as it provides the highest stability across both median and best-performing thresholds. The optimal configuration with RLS included a learning rate of 0.1 and a timestep of 1.0s, which provided sufficient temporal context for the model to capture attack signatures without inducing instability. Therefore, for the remainder of this experiment, these parameters (RLS, $\eta = 0.1$, $\Delta t = 1.0s$) will be adopted. Additionally, although the bipolar encoding range (-1 to 1) was identified as the optimal configuration, all four dataset mutations will be included in the following experiments. This facilitates a systematic comparison with the original benchmark's 0 to 1 range and allows for a broader assessment of the XAI results across varying label representations.

5.4 Explainability Evaluation For Spiking Neural Network-based Multi-Layer Perceptron

The primary objective of this evaluation is to assess the validity and effectiveness of the proposed XAI approach using a surrogate SNN-MLP model. If the SNN successfully learns the classification task, and the subsequent MLP successfully imitates the actions of the SNN, the resulting SNN-MLP model can be considered a faithful surrogate/proxy for the original

functionality of the SNN. Then if credible model-agnostic XAI methods – such as SHAP – are applied to this functionally accurate SNN-MLP, the resulting feature explanations should align closely with explanatory results derived from conventional, highly-accurate MLP benchmarks, as model-agnostic approaches are expected to yield consistent explanatory results regardless of the underlying model's internal structure.

To test these hypotheses, the benchmark model proposed in Wei et al. (2023) is adopted, and the original paper's XAI benchmarks are compared with the rebuilt Baseline MLP (BMLP) model – trained on the subsampled datasets – and finally the proposed SNN-MLP model. The original model from the previous study consisted of the input layer and 3 hidden layers with sizes (23, 15 and 10) and a sigmoid activation function for binary classification. When tested, it performed with an accuracy of 0.99. Considering that the original study had a much larger dataset than the current experiment, we replicated the original model trained on the same subset of data as the SNN-MLPs. The BMLP acts as the bridge between the original MLP model used in the previous study, and acts as a control group against which the explanatory performance of the SNN-MLP is compared. The rebuilt BMLP was configured with the same architecture and parameters as the original benchmark model, but was trained and tested with the reduced dataset described in Section 5.2 using the 4 different data encoding methods. The performance of the locally built BMLP and the SNN-MLP are shown in Table 5.3. The SNN was trained using RLS, and configured with a learning rate of 0.01 and a timestep of 1.0. The perfect accuracy (1.0) of the BMLP for all datasets

Table 5.3 Performances of the original MLP, BMLP and the SNN-MLP across the 4 datasets.

	Dataset 1	Dataset 2	Dataset 3	Dataset 4
Original MLP from Wei et al. (2023)	0.99			
Baseline MLP	1.00	1.00	1.00	1.00
SNN-MLP	0.82	0.73	0.87	0.78

is expected. Given that the original MLP trained and tested on the full dataset achieved a high accuracy of 0.99, it was highly probable that the same architecture would achieve a near-perfect classification on the smaller, subsampled set.

Although the accuracy of the SNN-MLP is not as high as the baseline, we deemed the accuracies satisfactory to proceed with the explainability testing. The purpose of this evaluation is not focused on optimising maximal classification performance, but rather on validating the faithfulness of the SNN-MLP as a surrogate model and testing whether SHAP can derive consistent explanations from it.

Figure 5.10 show the feature importance order for the rebuilt BMLP (left) and the SNN-MLP (right) with dataset 1, where each of the features' contribution is calculated by taking the absolute mean SHAP value for that feature across all samples. The rest of this section

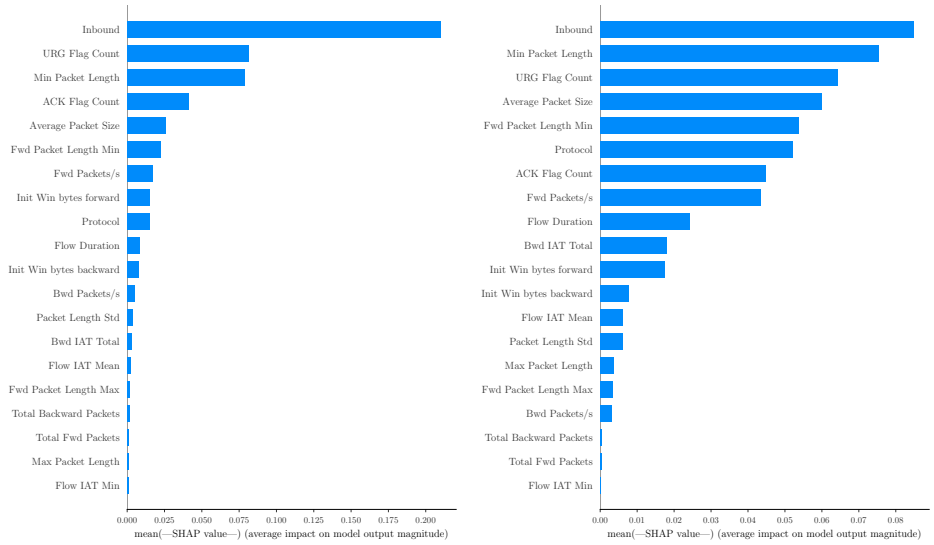
will evaluate the explanations generated for the SNN-MLP models, and discuss the alignment of the SNN-MLP's feature importance results with those of the BMLP.

In figure 2c from Wei et al. (2023), the top 5 contributing features were: *Inbound*, *Min Packet Length*, *Fwd Packet Length Min*, *URG Flag Count* and *Average Packet Size*. With the BMLP (Figure 5.10a), the feature contribution ranking follows a similar order where the top 5 features are: *Inbound*, *URG Flag Count*, *Min Packet Length*, *Ack Flag Count*, and *Average Packet Size*. Although *Fwd Packet Length Min* was the third most important feature in the original results, it moved to number 6 with the BMLP and *ACK Flag Count* instead ranked in the top 5. The general order was also slightly adjusted, however, the most contributing feature *Inbound* remained unchanged. Additionally, in both graphs the *Inbound* can be observed to contribute almost twice as much as its successor. Thus, it can be inferred that the reduced dataset preserved the core structure of the original data. While some variation in feature importance was expected due to the use of different models and training conditions – including dataset size and training state – the overall consistency supports the conclusion that the subsample used for training the SNN remains representative.

In contrast, with the SNN-MLP (Figure 5.10b), the top 5 features are: *Inbound*, *Min Packet Length*, *URG Flag Count*, *Average Packet Size* and *Fwd Packet Length Min*, which align with both the benchmark MLP and the rebuilt MLP. However, the feature contribution values on average much smaller than the benchmark and rebuilt MLPs, as the SHAP value for *Inbound* is only at about 0.08 while with the rebuilt MLP it's almost 0.2. The impact of the most contributing feature is also diminished with the SNN-MLP, where the difference between *Inbound* and *Min Packet length* is only about 15% (0.01) rather than double, as with the MLP. The variance in the feature contribution values and the order is likely due to the lower accuracy of the SNN-MLP, which impacts the subsequent SHAP outcomes. However, the proposed SNN-MLP model still generally follows the same order in the top 5 most important features, which reinforces the viability of using MLP-based explanations to interpret SNN behaviour.

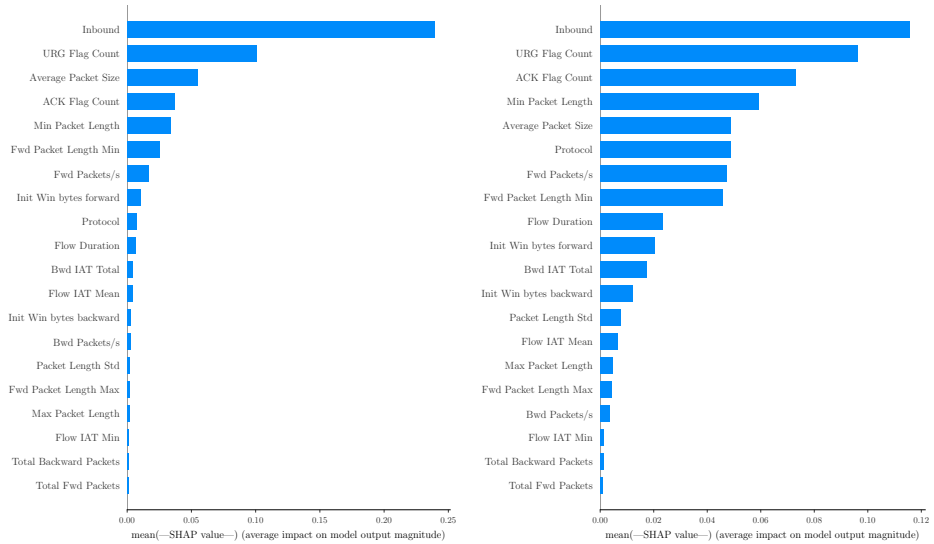
While the original paper only employed the first data encoding method (Dataset 1), our evaluation utilised all four encoding methods to confirm that the SNN-MLP translation process consistently yielded SNN-MLP which were functionally equivalent models across the full range of potential output encodings. Figures 5.11 to 5.13 present the feature contribution scores for the BMLP and the SNN-MLP models for datasets 2, 3 and 4.

We can see the same difference in the feature contribution scores' magnitudes when using dataset 2 – where the *Inbound* feature has twice the more significance with BMLP (Figure 5.11a) than with SNN-MLP (Figure 5.11b). This offset can again be attributed to the reduced accuracy of the SNN-MLP in contrast to the BMLP saturating the contribution



(a) BMLP (b) SNN-MLP

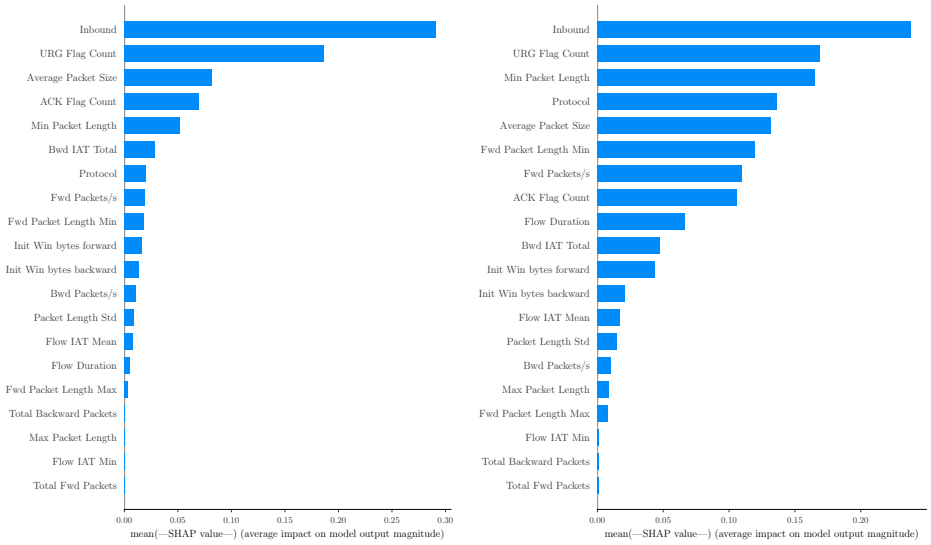
Fig. 5.10 Feature contribution rankings (Dataset 1).



(a) BMLP (b) SNN-MLP

Fig. 5.11 Feature contribution rankings (Dataset 2).

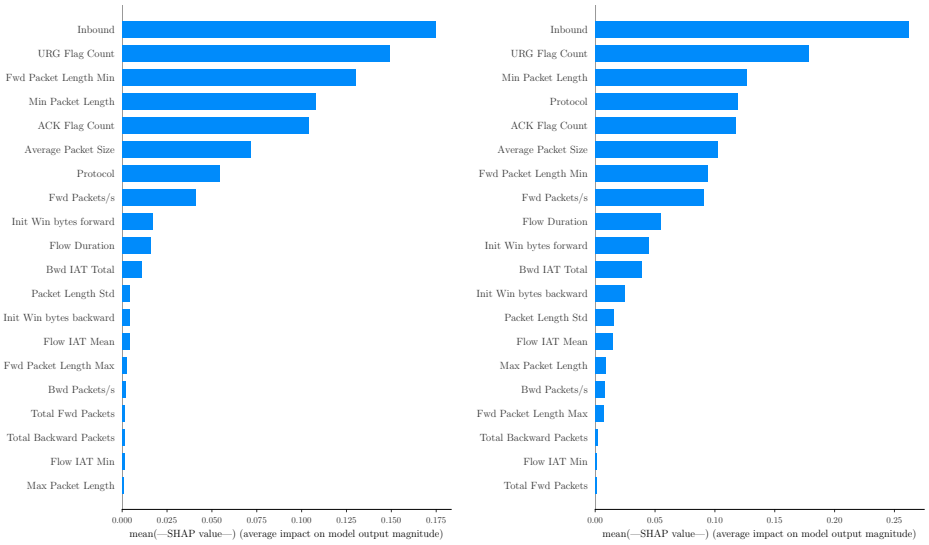
scores. Interestingly, although the SNN-MLP had the worst performance (0.73) with dataset 2 (benign = 0, attack = 1) out of all the data encoding methods, there are more features that share the same ranking between the BMLP and the than its counterpart dataset 1; with 8 features (*Inbound, URG Flag Count, Fwd Packets/s, Bwd IAT Total, Fwd Packet Length Max, Flow IAT Min, Total Backward Packets, Total Fwd Packets*) aligning between the two. The



(a) BMLP (b) SNN-MLP

Fig. 5.12 Feature contribution rankings (Dataset 3).

best performance for the SNN-MLP was achieved with Dataset 3 (0.87), which had the labels encoded as benign = 1, attack = -1. The difference in magnitude of the contribution scores are also a lot more similar. For example, the *Inbound* feature has a score of 0.29 with the BMLP (Figure 5.12a), and 0.24 with the SNN-MLP. However, the feature contribution order also varies the most with this dataset as well. Although the first two features are aligned, the next three features are: *Average Packet Size, ACK Flag Count and Min Packet Length*. In contrast, in Figure 5.12b each of these features are ranked at 5th, 8th, and 3rd, with *Protocol* placing 4th instead. There are also some other features with a large variation in ranking – *Bwd IAT Total* and *Flow Duration* also have notably large differences between the BMLP and the SNN-MLP. The large variance between the features could be attributed to the difference in accuracy, but also due to the different target values being used (i.e., 1 instead of 0). On the other hand, although the performance of the SNN-MLP with dataset 4 was second lowest out of all the data encoding methods, it also had the most features in the same order as the BMLP equivalent, with 10 features in the same order between the two models. Notably, the feature contribution of the BMLP is a lot more proportional, and this time the *Inbound* feature has a smaller contribution score (0.17) than the SNN-MLP (0.25). This again could be explained by the different encoding method used by the SNN, where although the model had learned the classification task accurately, due to the additive calculation method of SHAP – which uses a comparative evaluation of a feature's impact towards the average model prediction value – the magnitude of the feature contribution will be impacted by the label values as well.



(a) BMLP (b) SNN-MLP

Fig. 5.13 Feature contribution rankings (Dataset 4).

The feature contribution order between the original benchmark results and the rebuilt BMLP models demonstrate that the model architecture and training procedure used in the benchmark results are effective even with a reduced dataset. The main reason for variation in the order of feature contribution would likely be due to the fact that the rebuilt BMLP was trained with a smaller dataset sampled randomly from the original dataset, while in the benchmark the full dataset was used. The proposed SNN-MLP model generally follows the same order in the top five most important features, which reinforces the viability of using MLP-based explanations to interpret SNN behaviour. However, notable differences emerge beyond the top five features. In particular, the global SHAP value distribution for the SNN-MLP declines more gradually than in the MLP models, where feature contributions diminish almost exponentially. This difference is likely due in part to the lower accuracy of the SNN-MLP, leading to incorrect answers being incorporated into the global explanations. Additionally, although SHAP doesn't directly consider the activation function when calculating the contribution values, the different non-linear effects applied in each of the models' layers could also influence how the feature attributions propagate through the network. Despite these challenges, as the top contributing features remain relatively consistent across all three models, regardless of the data encoding methods, it supports the hypothesis that weight translation from an SNN to an equivalent MLP, followed by post-hoc explainability techniques, can offer a promising pathway for generating interpretable insights into SNN decision-making.

5.5 Discussion

The main goal of this case study was to demonstrate the effectiveness of the SNN-MLP translation method using single-dimensional SNNs. To achieve this, we first verified that the SNN-MLP model generated with our approach was mimicking the actions of the SNN it was based upon, regardless of the accuracy of the model. In Section 5.3 we saw that the converted SNN-MLP was able mimic the same actions as the SNN and maintained the performance of the trained SNN as well, confirming the validity of our method on single-dimensional SNN translations. The next part of the case study presented our results for experimenting with the SNN parameters(learning rule/learning rate/timestep/data encoding method) during the training phase to enhance the performance of the subsequent SNN-MLP in regards to the accuracy as well as robustness. By doing so, we would be able to reasonably assume that the explanations generated using the SNN would be meaningful, and verifiable through comparisons with other pre-existing results. From the results in Section 5.3, we found that the SNN is sensitive to changes in the learning rate and timestep, especially when using the PES learning rule, and we were able to produce 1-dimensional SNNs which can complete classification tasks well using tabular data. Finally, in Section 5.4 we confirmed that the SNN-MLP was not only correctly classifying the labels, but that it was using the same line of logic as proven methods, which supports our hypothesis of using a surrogate model with SNN characteristics can be used as an explanative method. Thus, with this case study we successfully demonstrated that the SNN-MLP translation method is a reliable approach towards a surrogate method to interpret single-dimensional SNNs, and additionally established a framework for the systematic optimisation of spiking network architectures towards more accurate classifications.

Chapter 6

Results: Multi-Dimensional Spiking Neural Network Classification

Extending the work done from the Distributed Denial-of-Service (DDoS) classification, the second case study focuses on demonstrating the effectiveness of our proposed translation approach with more complex Spiking Neural Network (SNN) model types – i.e., multi-dimensional models. For the results, we apply the proposed framework onto the problem of personalised mental health prediction. The significance of this case study lies in the critical nature of the medical domain, where clinical decisions must be both accurate and justifiable. In mental health monitoring, explainability is critical for clinicians and patients; they must understand the physiological and behavioural drivers behind a model's prediction to build trust and ensure safe intervention (Shah et al., 2021).

To evaluate the effectiveness of the SNN-based MLP (SNN-MLP) framework, we again use a baseline study as our primary benchmark for both predictive performance and interpretability (Chatterjee et al., 2024). In this research, a multimodal dataset involving 14 participants with mild to moderate depression over a one-month period was utilised. The dataset combines active data from longitudinal Ecological Momentary Assessments (EMAs) with passive data from lifestyle wearables and neurocognitive assessments.

The baseline study established a methodology for developing personalised deep learning models using various optimisation schemes. These models were verified using a five-fold cross-validation scheme and achieved error rates as low as 6% for specific individuals. Additionally, several Explainable Artificial Intelligence (XAI) methods – including Shapley Additive ExPlanation (SHAP) – were applied to the models' predictions to identify key patterns impacting the participants' depression levels (e.g., sleep patterns, diet, anxiety and stress levels etc).

By applying our methodology to this case study, we aim to demonstrate that the SNN-MLP can also provide meaningful relationships for highly personalised datasets. If the models are able to successfully capture the patterns of the participants datasets, and the feature explanations align with the baseline study we can verify that our SNN-MLPs identifies the same physiological and behavioural triggers, ensuring that our explanations remain clinically valid. The rest of this chapter is organised as follows: Section 6.1 provides an outline of the dataset used and describes its characteristics. Next, Section 6.2 explains the data preprocessing methods and model configuration parameters, and the performance of the SNN-MLP are presented in Section 6.3.1. Finally, the explanation results are presented in Section 6.4 followed by discussion and reflection of findings in Section 6.5.

6.1 Background

The dataset used in this paper originated from Shah et al. (2021), where data samples were collected from 14 adult human participants (mean age of 21.6 $\pm$ 2.8 years) who were all referred to the study from the University of California San Diego College Mental Health Program[1]. The primary inclusion criterion for the participants was the presence of moderate depression symptoms, which was assessed using the Patient Health Questionnaire (PHQ-9). The PHQ-9 is a diagnostic tool where the total scores are used to categorise the severity of the depression levels (Ford et al., 2020). The scale for the PHQ-9 ranges from 0 to 27, where any score below 5 indicates an absence of a depressive disorder, 5-9 represent minimal levels of depression, scores 0-14 represent moderate depression, 15-19 signify moderately severe depression, and scores of 20 or above indicate severe depression. The participants in this study required to be experiencing moderate levels of depression, and so were included only with PHQ-9 scores greater than 9, and had scores ranging between 10 to 17.

The data was collected through a 30-day period, and were collected using two methods. The first method was through in-person assessments conducted on days 1, 15 and 30 in a lab. Participants completed six cognitive assessment games which assessed their inhibitory control, interference processing, working memory, emotion bias, internal attention and reward processing levels. At this time, the particiants' neurocognitive data – such as the efficiency and consistency across each assesment day – and Electroencephalogram (EEG) data were collected. The second method collected the physiological and lifestyle data of the participants. Physiological data (e.g., heart rate, step count etc.) was collected from the participants using a Samsung Galaxy wristwatch that was worn continuously by the

[1]I gratefully acknowledge Dr. Jyoti Mishra Ramanathan and the authors of Shah et al. (2021) for granting permission to use their dataset for this work.

participants throughout the study excluding the few hours of charging time every 2-3 days. Lifestyle data was collected using a custom app called 'MindLog', through daily EMAs. An EMA is a research methodology that involves the repeated sampling of a subject's current behaviours and states in real-time within their natural environment, thereby minimising retrospective recall bias and increasing ecological validity Shiffman (2009). Participants received notifications to complete the EMA four times a day at 8AM, 12PM, 4PM, and 8PM for 30 days. Each assessment was designed to be completed within two minutes to reduce participant burden and ensure longitudinal compliance. During each EMA session, participants provided several categories of data:

- Mood Ratings: Participants rated their current levels of depression and anxiety using 7-point Likert scales (Robinson, 2014). For depression, the scale ranged from 1 ('Happy') to 7 ('Sad or Depressed'). Similarly, anxiety was rated from 1 ('Relaxed') to 7 ('Anxious').

- Stress Assessment: A 30-second stress assessment was conducted by requiring participants to tap the mobile screen following each complete breath (inhalation and exhalation). This assessment was used to record the mean breathing time as well as the breathing consistency during the assessment.

- Dietary Reporting: Participants logged their consumption of fats, sugars, and caffeine specifically within the preceding four-hour window. This was recorded using a 0-6 item scale, where participants selected the number of each item consumed from a provided list of standard dietary products (e.g., processed snacks or caffeinated beverages).

Finally, all the raw data with different sampling frequencies – seconds to minutes from the wristwatch, hours for the EMA data, days for the EEG data – were aggregated and/or extrapolated using the output variable (depression level) as the baseline.

6.1.1 Evaluation Dataset

Figure 6.1 presents the depression label distribution for each participant. For each participant, the histogram bars show the frequency of self-reported depression levels on the Likert scale (1 = happiest, 7 = most depressed).

Table 6.1 summarises the sample counts and some more details to each participant's depression levels. While the study protocol anticipated 120 samples per participant (four EMAs daily over 30 days), only one participant actually had exactly 120 samples, while most other participants had several samples missing. While this could be attributed to occasionally forgetting to record an entry or technical errors, some of the participants have a much lower

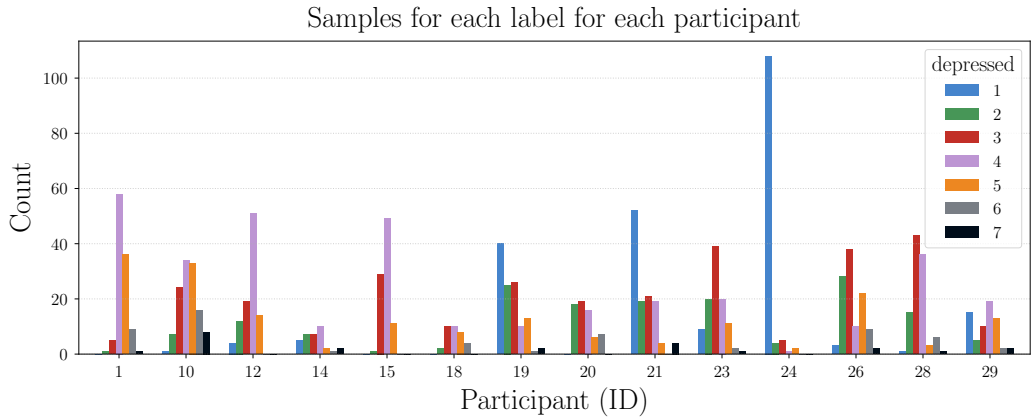


Fig. 6.1 Frequency of each depression level reported by each participant.

sample count, with Participants 20 and 29 containing roughly half of the expected number (66 samples), and Participants 14 and 18 only containing 34 samples. There is also one case where the number of samples exceeds the expected amount (Participant 10), which likely stem from extra assessments (perhaps another day), or accidentally completing entries twice unknowingly. This large variability in sample counts is an important constraint for modelling and is discussed below.

Table 6.1 Summary of depression labels.

Participant	Total Samples	Mean	Median	STD	Participant	Total Samples	Mean	Median	STD
1	110	4.45	4.0	0.79	**20**	66	3.47	3.0	1.28
10	123	4.39	4.0	1.32	**21**	119	2.33	2.0	1.52
12	100	3.59	4.0	1.01	**23**	102	3.14	3.0	1.23
14	34	3.24	3.0	1.60	**24**	120	1.21	1.0	0.71
15	90	3.78	4.0	0.67	**26**	112	3.49	3.0	1.42
18	34	4.06	4.0	1.13	**28**	105	3.45	3.0	1.06
19	117	2.50	2.0	1.49	**29**	66	3.36	4.0	1.65

Notably, from Figure 6.1 we can see that the sample distribution for the output labels are unbalanced. However, this imbalance is expected, as the inclusion criteria targeted individuals experiencing moderate depression symptoms. Therefore, extreme values on the 7-point Likert scale (representing either a total absence or severe levels of depression) are naturally less frequent than the mid-range scores. There are still exceptions – for example, as seen in Figure 6.1, Participants 21 and 24 reported a 1 for the majority of their assessments, suggesting that even within the clinical group, some individuals maintained a high baseline of reported ‘Happy’ states during the 30-day window.

Another likely cause for the inconsistent labels would be due to the native characteristic of EMAs – which are highly subjective assessments. Because these levels are self-reported measures, depending on the particiapnts' perception of the levels it can lead to variability of the results. For example, a score of 4 for Participant 10 (Median = 4) likely represents a different internal state than a 4 for Participant 19 (Median = 2). Given these challenges, in our study we use a personalised modelling approach, where separate models are trained to learn each of the individual's unique characteristics rather than a universal standard.

A more critical factor for model performance is the significant imbalance in the number of data samples, particularly for participants 14 and 18, who provided only a quarter of the expected samples. Despite this scarcity, we omit traditional data-balancing to make sure we maintain the integrity and authentic distribution of these highly personalised datasets. With these constraints in mind, the following section details the specific preprocessing standards and methodologies employed to train our SNNs using these features.

6.2 Experimental Setup

To ensure that we had a valid basis for comparing the SNN-MLP's explanations with the results from the original study, we followed the same approaches for data preprocessing as well as model training as Chatterjee et al.. The following sections detail the implementation of these approaches and outline the specific architectural modifications required to accommodate SNN models. The complete source code and the SNN configurations files in this section are available at https://github.com/janenotjung-hue/xaisnn/tree/depression. This study uses a non-public depression dataset originally published by Shah et al. (2021). Access was granted to the author by the dataset owners solely for use in this thesis. Due to data ownership and permission restrictions, the dataset is not publicly shareable by the author; requests for access should be directed to the original data owners.

6.2.1 Data Preprocessing

Feature Selection The original dataset contained 48 features. First, 3 speed-based features (*cumulative-step-speed, speed, noise*) were removed as they were computed from noisy distance features (Shah et al., 2021), and the *depressed* feature was separated as the target label values. The *datestamp* feature was then used to chronologically order the samples, which left 43 input features in total. Tables 6.2 and 6.3 present a summary of the features used in this case study.

Table 6.2 Summary of EMA-based and lifestyle features.

Category	Feature Name	Description	Window / Frequency
Self-Report (EMA)	Depressed	1-7 rating (Happy vs. Sad or Depressed)	4x daily
	Anxious	1-7 rating (Relaxed vs. Anxious).	4x daily
	Distracted	1-7 rating of self-perceived distraction.	4x daily
	Time of Day	(6:00, 10:00), (10:00, 14:00), (14:00, 18:00), (18:00, 23:59).	At time of rating
Respiratory	MeanBreathingTime	Average breathing duration during active stress test.	30s test (4x daily)
	Consistency	Regularity of breathing patterns during stress test.	30s test (4x daily)
Nutritional	past-day-fats	Total fatty items consumed.	Past 24 hours
	Past-day-sugars	Total sugary items consumed.	Past 24 hours
	Past-day-caffeine	Total cups of caffeine consumed.	Past 24 hours
Physiological	heart rate	Mean heart rate from smartwatch data.	$\pm$ 30 min window
	ppg-std	Standard deviation of heart rate.	$\pm$ 15 min window
Physical Activity	Step Metrics	Cumulative step count, calories, and distance.	Past 12 hours
	Exercise Metrics	Cumulative exercise calories and duration.	Past 24 hours
Sleep	Previous Night Sleep	Total hours of sleep.	Prior night

Table 6.3 Summary of neurocognitive and EEG features.

Category	Task / Domain	Feature Type	Metrics & Region
Inhibitory Control	Go Wait (GW)	Behavioural	Consistency, Performance Efficiency
		Neural Activity	Left DLPFC, dACC
Interference Processing	Middle Fish (MF)	Behavioural	Consistency, Performance Efficiency
		Neural Activity	Left DLPFC, dACC
Working Memory	Lost Star (LS)	Behavioural	Span (1-8), Consistency, Efficiency
		Neural Activity	Left DLPFC, dACC
Emotion Bias	Face Off (FO)	Behavioural	Consistency, Performance Efficiency
		Neural Activity	Left DLPFC, dACC
Internal Attention	Two Tap (TT)	Behavioural	Mean Breathing Time, Consistency
		Neural Activity	Left DLPFC, dACC
Reward Processing	Lucky Door (LD)	Behavioural	Gain/Loss Bias, Rare Gain Preference
		Neural Activity	Left DLPFC, dACC (Gain/Loss Expected Value)

Data Cleaning Next, each participant's datasets were scanned for missing and/or incorrect data which are presented below in Table 6.4. As shown in Table 6.4, 9 out of the 14

Table 6.4 Summary of samples for each participant.

Participant	Total Samples	Missing Values (Out of 43 × Total Samples)	Features with Missing Values
1	110	28	1
10	123	108	27
12	100	114	30
14	34	1	1
15	90	0	0
18	34	0	0
19	117	18	1
20	66	11	1
21	119	0	0
23	102	0	0
24	120	21	1
26	112	9	3
28	105	0	0
29	66	9	1

participants have features with missing values. Additionally, although none of the participants had a sample where all the features were missing, there were some participants who instead had constant values for some features. While this could make sense for categorical or ranking features, it does not for features acquired through the wearables. For example, a participant would be highly unlikely to have the same value for features like *heart-rate* or *cumulative-step-count* for 30 days.

To address the missing and invalid data, three distinct data cleaning strategies – Deletion, Manual Imputation, and Automatic Imputation – were used to create different versions of the dataset. Considering that the models are developed to be highly personalised, the method for handling missing information is likely to have a significant impact on the weighting of the features. For example, deleting entire rows (samples) may remove critical temporal transitions between mood levels, and imputing feature values can potentially add unnecessary noise to the dataset. To examine the impact of the data preprocessing method used, we test all three versions. In doing so, we can identify the preprocessing method that produces the most authentic representation of the participants' features, ensuring that the resulting SNN-MLP's predictions are grounded on high-quality data.

For the first method, Deletion, a simple approach was taken where participants with constant smartwatch feature values were first entirely removed (Participants 10, 15, 18, 21,

23, and 24). Then, any rows containing any missing data points were eliminated, serving as a straightforward baseline for comparison.

The second method, Manual Imputation (Manual), utilised domain knowledge regarding data types (discrete, continuous, or neurocognitive). After removing wearable features with constant or incorrect values, missing discrete data were imputed with the most frequent value. Missing continuous data were addressed using an iterative function that estimated missing values by treating the feature as a function of all other features in a round-robin manner. Neurocognitive feature gaps were filled with zero, assuming this implied a void or incomplete assessment.

The third method, Automatic Imputation (Auto), automated the process of selecting the best imputation technique to preserve the original data distribution. After removing problematic wearable features, seven different data-filling methods were evaluated for each missing feature, which are summarised in Table 6.5. The optimal imputation method was selected using the two-sample Kolmogorov-Smirnov (KS) test (Hodges, 1958), which compares two distributions by finding the maximum difference between the Cumulative Distribution Functions (CDFs) of the two distributions. Thus, our work retained all three

Table 6.5 Summary of all methods used for handling missing data in a feature column. Replicated from Chatterjee et al. (2024)

Method	Description
Mean	Fill missing data with the mean of the available data
Median	Fill missing data with the median of the available data
Forward fill	Fill missing data by continuing the last available data
Backward fill	Fill missing data by continuing the next available data
Linear interpolation	Fill missing data through linear interpolation by using the data points before and after the missing data
Iterative Imputation (van Buuren and Groothuis-Oudshoorn, 2011)	A strategy for imputing missing values by modelling each feature with missing values as a function of other features in a round-robin fashion. This method only uses samples with no missing data as the input (in case multiple features in a data point have missing values)
KNN Imputation (Troyanskaya et al., 2001)	Each sample's missing values are imputed by using the mean value from some nearest neighbours in the training set. Two samples are close if the features that neither are missing are close.

preprocessed datasets for subsequent model training, resulting in a total of 37 (9 (Deletion) + 14 (Manual) + 14 (Automatic)) datasets.

Label Encoding Since the output classes are numeric values ranging from 1 to 7, we encoded the target values into a one-hot encoded vector with 7 elements (i.e., $k = 7$) as discussed in Section 4.1.1. For example, a depression level of 3 is encoded as the vector $[0,0,1,0,0,0,0]$, and a score of 7 is encoded as $[0,0,0,0,0,0,1]$. Unlike the single-dimensional classification instance, the representation of each values (e.g., 0 to 1, -1 to 1) are less important, as the decision of the model can simply be inferred by identifying the index of the vector which has the highest value.

During the initial configuration phase, preliminary tests were conducted to evaluate the impact of the ensemble radius on model performance. Specifically, SNNs configured with a larger radius of 2 were tested to determine if an increased representational space would improve classification accuracy. However, these experiments resulted in significantly degraded performance and higher error rates across the participant datasets. Consequently, the radius was maintained at a value of 1 for all subsequent experiments, as it provided the most stable and accurate basis for the personalised modelling task.

Data Normalisation Data normalisation was performed using the standard normalisation method (Z-score normalisation), which centres the feature distribution around a mean of zero and scales it to achieve a unit standard deviation (Iverson, 2011). For each feature Z in the dataset, the normalisation is computed by subtracting the feature mean (μ) and dividing the result by the feature standard deviation (σ):

$$Z\prime = \frac{Z - \mu}{\sigma} \tag{6.1}$$

In the DDoS task, outliers were indicative of malicious activity and needed to be prominent to trigger an immediate network response, and therefore we used min-max scaling, to put all features in a scale between 0 and 1 proportional to their size. In the context of mental health modelling, however, the dataset contains high-frequency biological and neurocognitive signals where extreme outliers are often the result of sensor noise rather than clinical changes. By scaling data based on the standard deviation, we ensure that all input features contribute equally to the SNN weights regardless of their original units. This is critical for the Leaky Integrate-and-Fire (LIF) neurons within the ensemble, as their membrane potential relies on a consistent range of input currents to reach the firing threshold. If one feature had a significantly larger scale than others, it would dominate the integration process and cause the neuron to fire prematurely, regardless of the informational value of the other features. For example, a heart rate spike from 80 to 200 bpm would saturate the neuron and cause it to fire prematurely, regardless of the informational value of other features. Normalising by standard deviation captures the inherent variance of the data more effectively, providing a stable input

signal that prevents the spiking thresholds from being overwhelmed by noise while still preserving the meaningful temporal transitions in the participant's mood. Therefore, this approach was selected to ensure that all input features contribute equally to the SNN model's weights.

6.2.2 Model Configuration

In Chatterjee et al. (2024), both classification and regression models were explored to predict the depression levels. However, as discussed in Section 4.1.1, without a set index or reliable thresholding rule, it is difficult to map a 1D SNN output back to one of the seven discrete depression levels, to definitively determine the predicted class. Therefore, we only consider classification models in this case study. The original Multi-Layer Perceptron (MLP) utilised a feedforward structure with an input layer, several hidden layers, and a seven-neuron output layer (corresponding to the seven classes), employing a softmax activation for probability normalisation.

Our SNN model was structured as a feedforward chain with two SNN ensembles E_A and E_B. Ensemble E_A contained 300 neurons, and its dimensionality was set to match the feature dimension of the input (ranging from 40 to 43 depending on the preprocessing method). The second ensemble (E_B) was configured with 80 neurons and a dimension of 7, to match the number of output classes like the original MLP. The much larger neuron populations compared to the models in the DDoS study (20 and 5 respectively) is necessary, as an ensemble must possess enough neural resources to accurately represent and transform high-dimensional input vectors. Otherwise, with a smaller population, the ensemble would likely struggle to differentiate between patterns, resulting in a saturated state where neurons fire constantly regardless of the input signal.

The final prediction from the SNN as well as the subsequent SNN-MLP was derived by applying the argmax function to the 7-dimensional decoded output vector, selecting the index (class) corresponding to the highest probability value. Following the results from the previous case study, the SNNs were trained with the Recursive Least Squares (RLS) learning rule, with a learning rate of $\eta = 10^{-5}$ and a timestep (Δt) of 1.0s. Although in the previous study we saw that longer timesteps were generally more likely to result in better performance, as we train these models using cross-fold validation with epoch-based training, longer timesteps may potentially cause the model to overfit. Therefore, the models were configured with a shorter timestep for this experiment.

6.2.3 Evaluation Metrics and Methods

Like the previous case study, we used accuracy as the base measure of performance for the classification tasks. However, in Chatterjee et al. (2024), the authors used the Mean Average Error (MAE) as the unified metric to fairly evaluate performance across both their regression and classification models, and accuracy was omitted during the performance evaluation sections. Thus, to ensure our SNN results were directly comparable to the results from the original paper, we also included MAE as an evaluation metric.

For this experiment, the use of MAE is also beneficial as the target labels are ordinal rather than purely categorical (Lai et al., 2023). In a multi-class classification task involving a 7-point Likert scale, the relationship between classes is sequential. For example, misclassifying a 7 (severely depressed) as a 6 is a significantly smaller error than misclassifying it as a 1 (happy). While traditional accuracy treats both mistakes as equally incorrect, MAE penalises the model based on the magnitude of the error, and provides a more meaningful reflection of the model's predictive reliability.

The MAE is defined as the average of the absolute differences between the predicted values (y_{pred}) and the actual values (y_{actual}) over n data points (Xu et al., 2024):

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_{pred} - y_{actual}| \tag{6.2}$$

6.2.4 Model Training

To ensure a statistically robust and unbiased evaluation, we implemented a Stratified 5-fold Cross-Validation (CV) scheme, following the validation approach used in the original benchmark study. Figure 6.2 outlines the process for CV. This scheme involves dividing

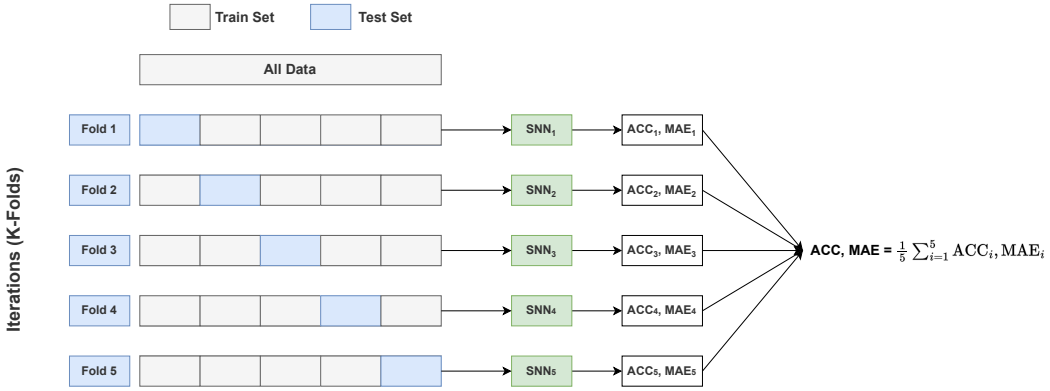


Fig. 6.2 Example of K-fold cross validation using 5 folds (i.e., $k = 5$).

the normalised dataset into five distinct, non-overlapping folds. In this configuration, five

separate models are trained, with each model utilising a unique 80% of the data for training and the remaining 20% for testing. A model is trained on four of these folds (the training dataset) and tested on the remaining single fold (the testing dataset). The division is stratified, meaning each of the five folds is constructed to roughly contain the same proportion of the seven labels (depression levels 1-7). This prevents a scenario where a specific fold might lack examples of a certain mood state, which is a significant risk given the small sample sizes and class imbalances noted in Table 6.1. The final performance metrics are obtained by averaging the results across all five folds. Due to the small number of samples in each of the particpants' datasets, our SNNs for every fold were trained for 100 epochs in Nengo. Since Nengo does not feature the same native early-stopping utility, a custom evaluation method was used instead. As mentioned in Section 4.3, after the training phase of every epoch, an SNN-MLP model is generated using the weights of the SNN at the current timeframe. After the full 100 epochs were completed for all five folds, the resulting SNN-MLP models saved at each epoch were all evaluated on the respective testing data to record the accuracy and the MAE. The epochs that exhibited the minimum MAE across all five folds were selected as the optimal "stopping point". This methodology ensured that our SNN results were validated under an equivalent, statistically robust, and unbiased condition as the original MLP models.

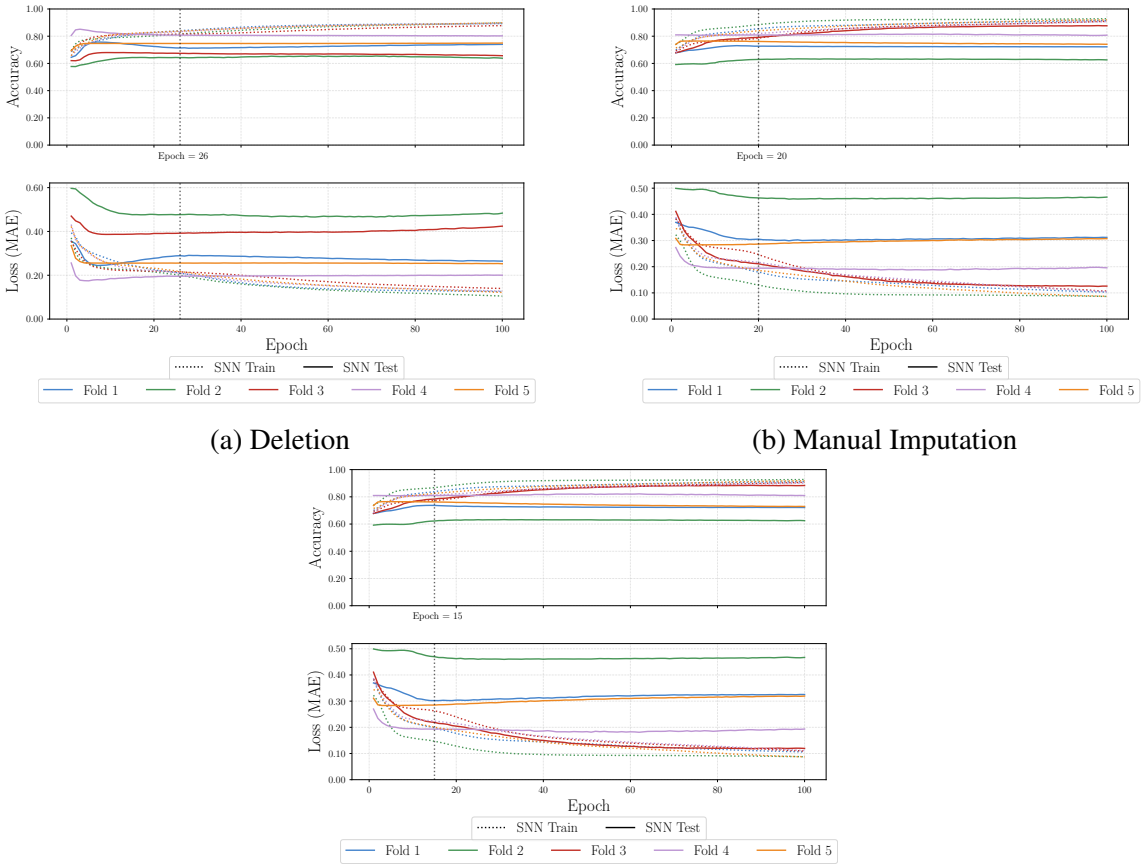
6.3 Evaluating Model Performances Across all Participants

To validate the explanations generated by our models, the models first have to be accurate to some degree, otherwise whatever explanations we generate won't be meaningful. Explainability results are only meaningful if the model has successfully captured patterns within the data; therefore, we evaluate performance based on two criteria. First, we examine the correctness of the SNN by analysing its training and testing trajectories across individual folds. Second, we verify the effectiveness of the translation by ensuring the SNN-MLP surrogate accurately mirrors the behaviour of the original SNN.

6.3.1 Evaluating Impact of Cross-Validation

To understand the variance in model performance, we first examine the performance of the individual training and testing trajectories for each of the five folds. We present the initial results using Participant 1 as they represent a high-quality data baseline for the study. This participant provided nearly all of the 120 expected samples (110 samples) and was evaluated with all preprocessing methods. Additionally, their depression scores demonstrate a reliable distribution, with a median of 4.0 and a standard deviation of 0.79, providing a stable

target for evaluating the model's performance across different folds. Figure 6.3 presents the Accuracy and MAE over 100 epochs for Participant 1 across the the preprocessing methods: Deletion (Figure 6.3a), Manual (Figure 6.3b), and Automatic (Figure 6.3c). In these figures, the dotted lines represent the training performance of the SNN, while the solid lines represent the corresponding testing performance. Each colour corresponds to a specific cross-validation fold.



(a) Deletion

(b) Manual Imputation

(c) Automatic Imputation

Fig. 6.3 Train/test performance for Participant 1 across all folds. The black dotted line represents the epoch which had the best combined test performance (lowest MAE) of the folds.

The SNN training curves (dotted lines) can be seen to stabilise very quickly across all folds and preprocessing methods, generally stabilising within the first 20 to 40 epochs. By epoch 20, the training MAE for most folds drops to near zero (≈ 0.1), indicating that the SNN effectively learned the specific samples present in the training set.

However, a significant variance can be seen in the testing performances (solid lines) across the different folds. For example, in Figure 6.3c, fold 3 (red) achieves a peak SNN test

accuracy of ≈ 0.90. In contrast, fold 2 (green) performs significantly lower, settling at an accuracy of ≈ 0.60. A similar disparity is visible in the MAE values, which fluctuate between ≈ 0.12 (fold 3) and ≈ 0.50 (fold 2). For all five folds, the test MAE settles at slightly values than the training values (between 0.60 and 0.90) regardless of the preprocessing method used.

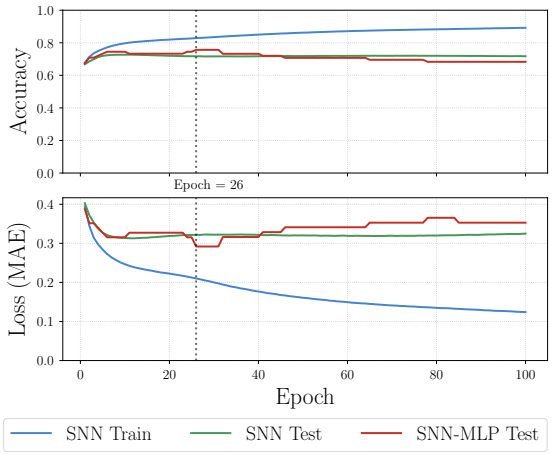
Each of the figures also contains a vertical line, which represents the epoch with the lowest average test MAE across all folds. With the Deletion method, this is epoch 26, for the Manual method this is epoch 20, and for the Automatic method this would be epoch 15. While the best epoch varies between the figures, we can see that generally, after this point, continuing the train the model results in decaying values for the test accuracy and increases in the total loss (even if individual fold's test performances improve) as the training performances continue to improve.

The gap between the training and testing performances indicates that the SNN model suffers from overfitting across most of the data splits; it learns the training examples nearly perfectly, but its ability to generalise to unseen test data is heavily dependent on the specific composition of the individual fold, as shown by the variance between the different folds' test performances. However, by using a 5-fold CV approach, we can provide more reliable performance results that isn't affected by the bias of a single data split. Additionally, we can capture the model state with the highest generalisation by determining the ideal epoch, which is essential to validate the explanations generated by the model as we want to ensure that the feature explanations are produced using learnt underlying patterns, rather than just memorising feature values or capturing noise. Given the constraints of the dataset size, this multi-fold approach provides the most statistically sound alternative for ensuring that the resulting SNN-MLP reflects the true logic of the spiking network.

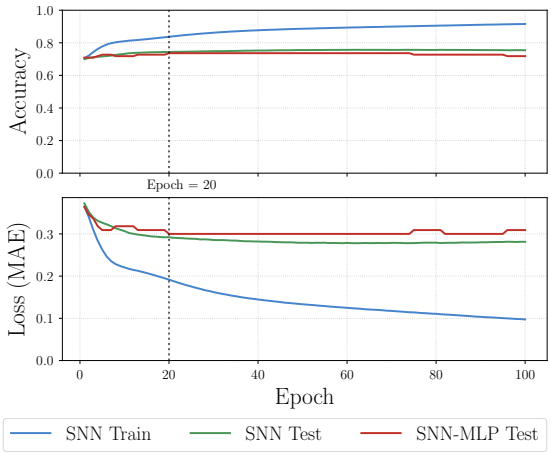
6.3.2 Verifying Initial Translation

The primary objective of this stage is to confirm that the SNN-MLP correctly mimics the actions and generalisation behaviour of the SNN. Figure 6.4 illustrates the average accuracy and MAE at every epoch for Participant 1 across the three preprocessing methods (see: Appendix B for the combined train/test results for all participants). Each of the plots present the train/test performance of the SNN, as well as the SNN-MLP on the same testing data as the SNN at each stage. By plotting the average performance across all five folds, we can evaluate how closely the surrogate model tracks the original spiking model's performance on unseen testing data.

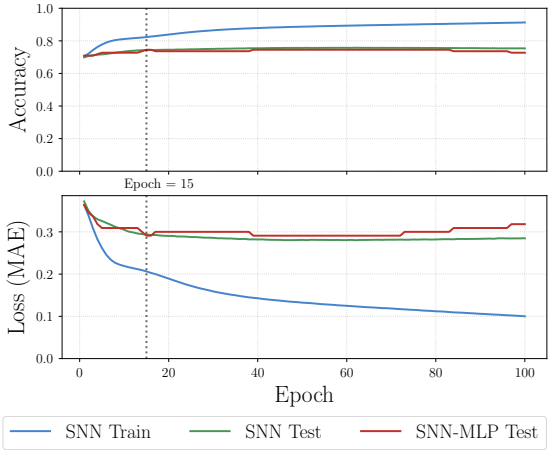
From the figures, we can confirm that the curves for the train and test performances in Figure 6.4 align with the trends for the performances in Figure 6.3.



(a) Deletion



(b) Manual Imputation



(c) Automatic Imputation

Fig. 6.4 Average accuracy & MAE across all folds for Participant 1. The black dotted line represents the epoch which had the best averaged test performance (lowest MAE).

We can also observe that although the testing performance with the SNN-MLPs are lower than that of the training performance, the lines representing the test accuracy of the original SNN-MLP (red line) align closely with the test accuracy of the SNN (green line), and likewise for the loss plots across all three preprocessing methods. As a result, we can observe that the SNN-MLP correctly captures the performance degradation associated with overfitting. For example, in Figure 6.4a, after reaching peak performance at epoch 12, the SNN-MLP accuracy begins to degrade in parallel with the SNN. The alignment between the SNN and the SNN-MLP testing performances thus confirms that the SNN-MLP successfully captured and replicated the internal weights of the SNN, and the resultant behaviour of the trained SNN at each epoch. Therefore, we can consider the surrogate model to be a reliable proxy for the SNN, regardless of the specific epoch.

Table 6.6 Summary of test performances for all participants for SNN and SNN-MLP.

Participant	Model	Deletion		Manual		Automatic		Participant	Model	Deletion		Manual		Automatic	
		AVG	STD	AVG	STD	AVG	STD			AVG	STD	AVG	STD	AVG	STD
1	SNN	0.322	0.011	**0.287**	0.016	0.288	0.015	20	SNN	**0.606**	0.025	0.699	0.012	0.698	0.012
	SNN-MLP	0.337	0.019	0.305	0.010	**0.301**	0.012		SNN-MLP	**0.569**	0.045	0.645	0.035	0.640	0.033
10	SNN			**0.916**	0.017	0.916	0.014	21	SNN			**1.117**	0.016	**1.117**	0.016
	SNN-MLP			1.024	0.072	**0.869**	0.017		SNN-MLP			**1.038**	0.021	**1.038**	0.021
12	SNN	0.721	0.027	0.739	0.029	**0.713**	0.028	23	SNN			**0.943**	0.037	**0.943**	0.037
	SNN-MLP	0.695	0.024	**0.684**	0.020	0.728	0.033		SNN-MLP			**0.940**	0.030	**0.940**	0.030
14	SNN	**1.215**	0.026	1.356	0.049	1.362	0.047	24	SNN			0.213	0.005	**0.210**	0.002
	SNN-MLP	**1.174**	0.031	1.279	0.045	1.284	0.056		SNN-MLP			0.246	0.019	**0.243**	0.020
15	SNN			**0.406**	0.013	**0.406**	0.013	26	SNN	**1.217**	0.018	1.359	0.015	1.367	0.017
	SNN-MLP			**0.441**	0.024	**0.441**	0.024		SNN-MLP	**1.235**	0.043	1.311	0.014	1.316	0.021
18	SNN			**0.958**	0.074	**0.958**	0.074	28	SNN	**0.784**	0.032	**0.784**	0.032	**0.784**	0.032
	SNN-MLP			**0.950**	0.099	**0.950**	0.099		SNN-MLP	**0.786**	0.044	**0.786**	0.044	**0.786**	0.044
19	SNN	**0.769**	0.022	0.913	0.010	0.897	0.012	29	SNN	**1.061**	0.059	1.248	0.054	1.261	0.043
	SNN-MLP	**0.763**	0.022	0.880	0.041	0.856	0.021		SNN-MLP	**1.075**	0.065	1.223	0.089	1.233	0.075

A notable difference between the two model types is the stability of their test curves. The test curves for the SNN are consistently smoother and generally flatter after stabilising. However, the SNN-MLP curves (dashed lines) exhibit more instability, showing sharp, noticeable jumps throughout the epochs. This difference is likely attributable to the underlying mechanisms of the models. MLP models will relies on fixed, deterministic weights to produce the exact same continuous output given the same input. The SNN's final output, however, is derived from an argmax filter applied to decoded spiking activities. Because the SNN output represents an average firing rate over a temporal window, it effectively filters out high-frequency noise, resulting in a smoother mean curve.

Despite these fluctuations, the overall trend alignment confirms that the surrogate remains an accurate representation of the spiking model's learning trajectory. Having established the translation fidelity for an individual participant, the following section examines the results across all participants to identify the optimal epoch (selected based off the minimum test MAE for the SNN-MLP) and the most effective preprocessing method for the final evaluation.

6.3.3 Comparing Across Participants

Table 6.6 presents a summary of the average test performance (MAE) and standard deviation (STD) between the SNN and the SNN-MLP across all three data preprocessing methods, for each of the participants. A key pattern observable from the table is that the SNN-MLP is a highly faithful surrogate. For 11 out of 14 participants, the SNN and SNN-MLP produce the best MAE using the same preprocessing method. Even in the cases where they differed, the absolute difference in the average MAE between the Manual and Automatic methods are negligible (Participant 1: 0.004; Participant 10: 0.047, Participant 12: 0.029). Notably,

in most cases where Deletion was used, it consistently resulted in the lowest average MAE (e.g., Participants 14, 19, 20, 26, 29). For participants where Deletion was not feasible or performed poorly, the Manual method was frequently the best method. Interestingly, in most cases where Manual and Automatic imputation were compared, their performance was nearly identical (e.g., Participants 15, 18, 21, 23), suggesting the models were robust to the specific imputation algorithm used.

Further evaluation is provided in Table 6.7, which presents the smallest test MAE achieved for each participant across the three preprocessing methods, along with the corresponding optimal epoch. Even for models where the optimal epoch differed between the two imputation

Table 6.7 Summary of the epochs with the lowest MAE per participant. For each participant, the lowest MAE value between the preprocessing methods are bolded.

	Deletion		**Manual**		**Automatic**	
Participant	**Epoch**	**MAE**	**Epoch**	**MAE**	**Epoch**	**MAE**
1	26	0.292	20	0.300	**15**	**0.291**
10			**1**	**0.828**	10	0.837
12	**2**	**0.627**	10	0.650	5	0.650
14	**4**	**1.114**	1	1.195	1	1.195
15			**4**	**0.367**	**4**	**0.367**
18			**2**	**0.643**	**2**	**0.643**
19	**72**	**0.727**	10	0.835	6	0.827
20	**12**	**0.473**	27	0.588	26	0.603
21			**90**	**1.005**	**90**	**1.005**
23			**2**	**0.821**	**2**	**0.821**
24			**1**	**0.208**	**1**	**0.208**
26	**24**	**1.163**	8	1.285	11	1.285
28	**5**	**0.667**	**5**	**0.667**	**5**	**0.667**
29	**31**	**0.970**	60	1.114	66	1.145

methods, the difference was minimal; the largest difference recorded was only 9 epochs (Participant 10: Automatic at epoch 10 vs. Manual at epoch 1). The similarities between the two methods suggest that the final quality of the data supplied by both imputation methods leads to highly similar training dynamics – and therefore, the same optimal stopping points. When comparing the model performances across all three methods, the Deletion method leads to the lowest MAE (the best outcome) in the majority of cases (participants 12, 19, 20, 26, and 29). This outcome is notable because the Deletion method removed the highest volume of data, removing all samples and features with missing or invalid data.

In terms of the performance – the highest performance recorded was with Participant 24, where the stabilised test accuracy was $\approx .9$ with all preprocessing methods. This result however is likely due to the extreme class imbalance present in Participant 24's dataset, where the model could achieve high accuracy by primarily predicting the dominant label (depression level 1). Similarly, participants 1, 12 and 15 who also had higher degrees of imbalance across the labels tended to achieve higher accuracies compared to those with more balanced labels (e.g., participants 10, 14, 23).

Figure 6.5 presents the test results for all participants, showing the combined test accuracy and MAE across the 5-fold cross-validation over 100 epochs. In these figures, the blue lines represent the test performance (solid for SNN, dashed for SNN-MLP) using the Deletion dataset, the green lines represent performance using the Manual dataset and the red lines represent performance using the Automatic dataset. For convenience, the best epochs for each preprocessing method (the epochs which produced the lowest testing MAE for the SNN-MLP) presented in Table 6.7 are marked in each of the figures in the corresponding colour as the preprocessing method.

From the figures, we can confirm that for all participants, the SNN-MLP surrogate models have successfully learnt the behaviours of the SNNs, as all the lines for the SNN-MLP follow that of the SNN with minimaldifferences. Because the differences between the two models are minimal across all 100 epochs even with the more erratic trends for the SNN-MLP, it is justifiable to use the MAE of the SNN-MLP as the primary metric for identifying the optimal model state.

The trendlines for all participants confirm that using several epochs rather than a single passthrough leads to better performance, and that fewer epochs generally lead to the best performance. For most of the participants and the preprocessing methods, the test accuracies and MAE stabilise quickly, often within the first 10 to 30 epochs. Continuing training beyond this point typically results in a plateau or slight degradation in test performance, confirming that the models benefit from out early-stopping procedure. Notable exceptions are Participants 21 (with the Manual and Automatic dataset) and 29 (with the Deletion dataset), whose loss curves continue to decrease slightly over a longer epoch range, suggesting that their respective datasets allowed for a slower, more sustained generalisation.

6.3.4 Comparing With Previous Results

In the previous work by Chatterjee et al., the primary focus was on optimising the model performance for each individual participant, resulting in tailored configurations where the best preprocessing method, model type, and hyperparameters were selected specifically for each participant. The current study is designed to replicate the performances for each of

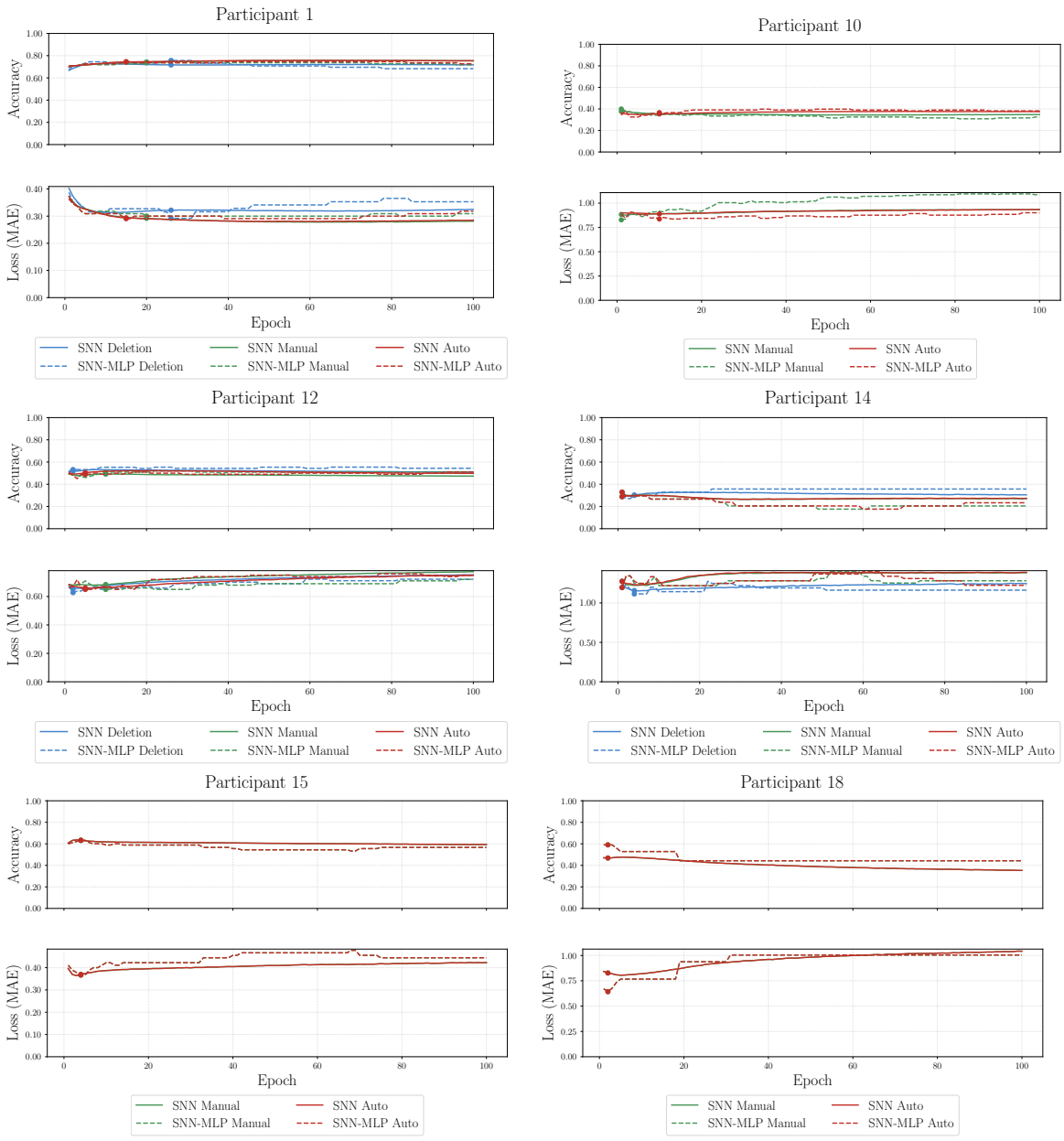


Fig. 6.5 Combined test accuracy & MAE for all participants.

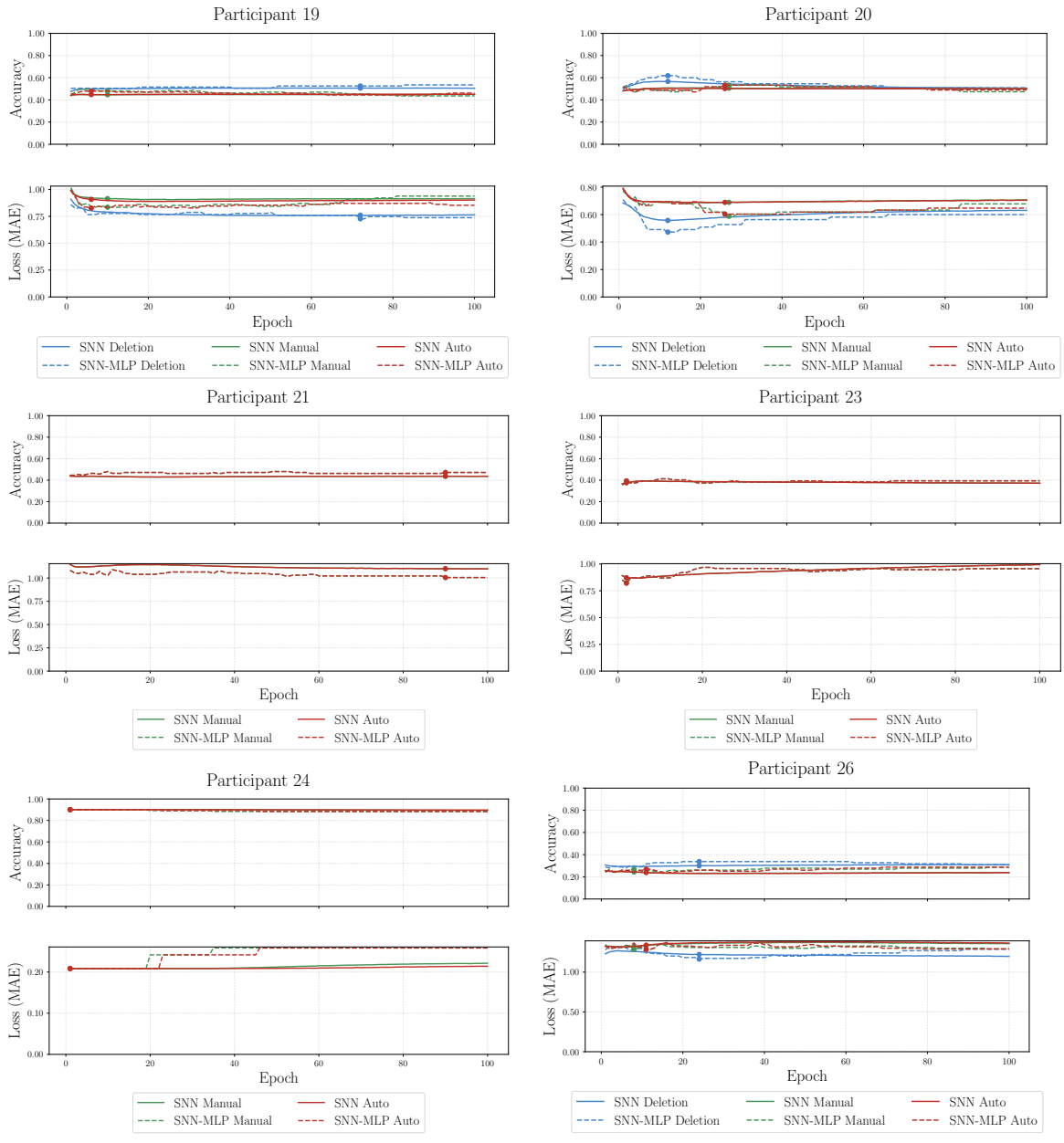

Participant 19
Participant 20
Participant 21
Participant 23
Participant 24
Participant 26
Accuracy
Loss (MAE)
Epoch
SNN Deletion
SNN-MLP Deletion
SNN Manual
SNN-MLP Manual
SNN Auto
SNN-MLP Auto

Fig. 6.5 *Cont.*

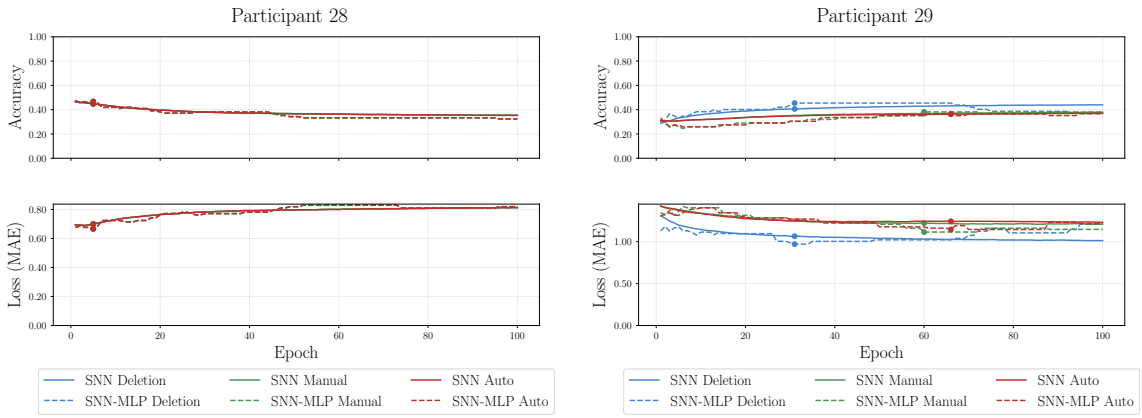


Fig. 6.5 *Cont.*

the optimal MLP from the original study as close as possible to validate the explanations. Since the original study reported the MAE for both regression and classification models, we use this metric as the comparative standard. Table 6.8 presents a direct comparison between the best MAE achieved by the original MLP models (using their fully optimised, best configurations) and the best MAE achieved by our SNN-MLP classification models (using the optimal epoch/preprocessing combination identified in Table 6.7). Overall, the averaged MAE scores for our results (0.63) were slightly lower than the benchmark MLPs (0.71). From the table, we can see that our SNN-MLP models outperform the previous research baseline for six participants (1, 15, 19, 21, 23, and 29). While the baseline MLP models achieve a lower MAE for the remaining participants, the difference is often minimal (e.g., 0.005 for Participant 18). Given that our SNN approach was not focused on an extensive hyperparameter search but rather on creating a fixed architecture for subsequent XAI analysis, we can consider the performance of the SNN to be suitable.

Interestingly, for most participants the smallest MAE was achieved using the same data preprocessing method between our approach and the previous research. The exceptions were Participants 1, 24 and 26 – however, for participants 24 it was more the case that two preprocessing methods (Manual and Automatic) had tied for the best MAE. For Participant 1, the difference between the optimal MAEs (0.291 - achieved using the Automatic method) and the optimal MAE using the same method as the previous work (0.292 - Deletion method) is marginal. However, in Figure 6.5, we can see that the test MAE is consistently lower using the Automatic method (red line) for both the SNN and SNN-MLP than the Deletion method (blue line). Similarly, for Participant 26, the best performance from the original research was achieved using the Manual method. In our experiment, the corresponding MAE was 1.285 (recorded at epoch 8), while the best MAE (1.163) was achieved using the Deletion method (see Table 6.7). Again though, we can see that the test MAE of the Deletion method

Table 6.8 Summary of performances of the proposed SNN model and the benchmark model from Chatterjee et al. (2024).

Participant	Model	Method	MAE	Participant	Model	Method	MAE
1	MLP Classifier	Deletion	0.295	10	MLP Classifier	Manual	**0.715**
	SNN-MLP Classifier	Automatic	**0.291**		SNN-MLP Classifier	Manual	0.828
12	MLP Regressor	Deletion	**0.582**	14	MLP Regressor	Deletion	**0.829**
	SNN-MLP Classifier	Deletion	0.627		SNN-MLP Classifier	Deletion	1.114
15	MLP Classifier	Manual	0.494	18	MLP Classifier	Manual	**0.638**
	SNN-MLP Classifier	Manual	**0.367**		SNN-MLP Classifier	Manual	0.643
19	MLP Regressor	Deletion	0.989	20	MLP Classifier	Deletion	**0.455**
	SNN-MLP Classifier	Deletion	**0.727**		SNN-MLP Classifier	Deletion	0.473
21	MLP Regressor	Manual	1.103	23	MLP Classifier	Manual	0.936
	SNN-MLP Classifier	Manual	**1.001**		SNN-MLP Classifier	Manual	**0.821**
24	MLP Classifier	Automatic	**0.125**	26	MLP Regressor	Manual	**1.013**
	SNN-MLP Classifier	Manual	0.208		SNN-MLP Classifier	Deletion	1.163
28	MLP Classifier	Deletion	**0.562**	29	MLP Classifier	Deletion	1.03
	SNN-MLP Classifier	Deletion	0.667		SNN-MLP Classifier	Deletion	**0.970**

(blue line) SNN is consistently much lower than that of the Manual method (green line). Therefore, we can confirm that the results for these participants are reliable rather than the result of a single outlier even if they do not align with previous work. As the previous work only presented the MAE instead of accuracy, it is difficult to make a concrete judgement of which model was better. However, given that the SNN-MLP achieved similar performance to the fully optimised baseline MLPs in the previous work in terms of MAE, we can consider the SNN-MLP to be a reliable model for the classification task.

6.4 Explainability Evaluation

To demonstrate explainability with the multi-dimensional SNNs we again apply SHAP to our final selected SNN-MLP models to generate feature-contribution explanations. However, a key methodological difference from the previous chapter is that this time, our models were trained and tested using K-fold CV. Therefore, simply using the full sample dataset to generate SHAP values for a single final model would not have represented the decisions made by the different models. To generate explanations compatible with the cross-validation methodology, the SHAP values were computed using the following procedure for each participant: For each of the five K-folds, the SHAP values were calculated using the SNN-MLP surrogate model corresponding to that specific fold. The training data for that fold was used as the SHAP background information, acting as the baseline. The testing data for

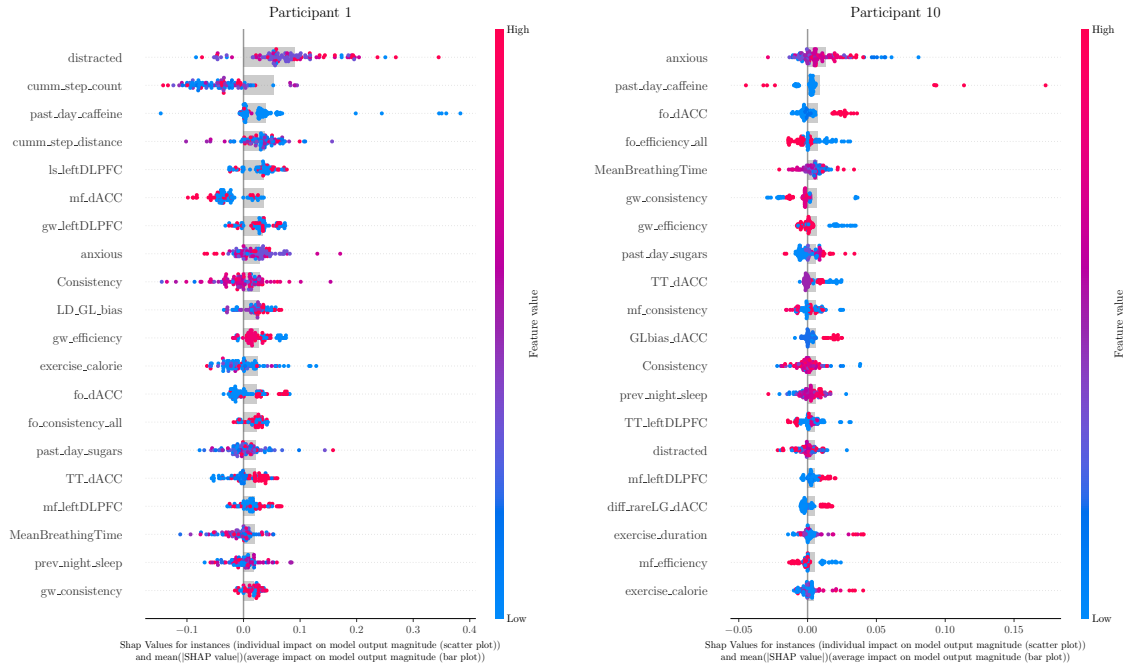
that same fold was used as the set of instances for which the SHAP values were computed, ensuring that the explanations are generated on unseen data samples. After computing the SHAP values for all instances across all five folds, the resulting values were combined. The overall global feature importance for the participant was determined by taking the mean of the absolute SHAP values for each feature across the entire aggregated test set. The following section presents our SHAP feature contribution results using the best model for each of the participants, and we compare to the findings from Chatterjee et al. (2024) in Section 6.4.2 to evaluate the validity of our explanations.

6.4.1 Explanations For Each Participant

Figure 6.6 presents the top contributing SHAP values for each of the participants. Like the previous case study, we use the bar plot which represent the global feature importance values for each feature, calculated by the mean absolute SHAP values across all datapoints in the aggregated test set. We additionally include a scatter plot superimposed on top, which show the distribution of local SHAP values for individual data samples. For every feature, each point represents a single prediction instance, where the colour intensity of the points indicate the magnitude of the feature's original value, and the position on the x-axis indicates the feature's contribution to shifting the model's output away from the base value (average prediction value calculated based on the model's training set). For example, we can see that for Participant 10 (see Figure 6.8a), larger *past-day-sugars* input values generally push the predicted value to a higher level. Participants with bigger datasets will thus have more data points on each of the feature's axes, as there are more predicted instances. For example, there are approximately 100 points in Participant 12's results (Figure 6.6c) than for Participant 14 (Figure 6.6d), who only had 34 samples in their dataset. Although there are a total of 40-43 features used for each of the participants, after the top 15-20 features, the rest of the contributions were minimal. For example, in Figures 6.6j and 6.6l, the last 4 to 5 features can be seen to have almost have zero impact on the model's decision. Therefore, we only present the top 20 features in these graphs.

Generally, the features anxious or distracted are common across most participants as the top contributing features, but then the rest of the rankings vary significantly between each of the participants. This variance however is expected, as it reflects the unique behavioural and physiological triggers of each participant's depressive state.

In some instances, the models captured clear and plausible patterns of feature contribution. For example, with Participant 12 (Figure 6.6c), the SHAP analysis presents a distinct linear trend for the impact of *past-day-caffeine* on the predicted depression level. In this case,

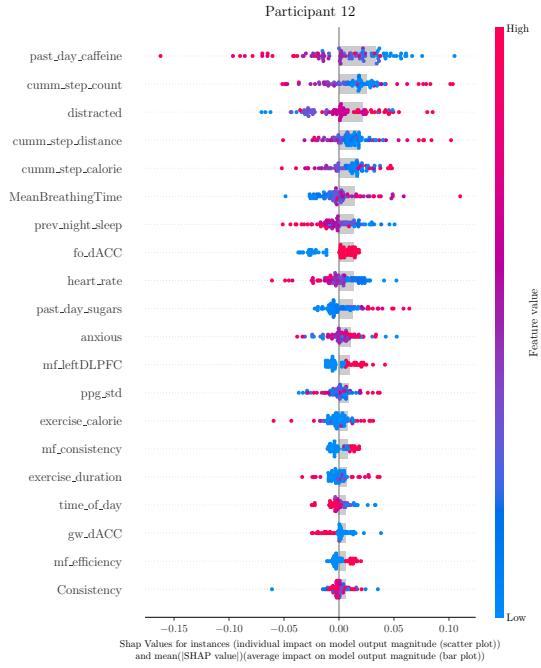


(a) Participant 1 (Automatic) (b) Participant 10 (Manual)

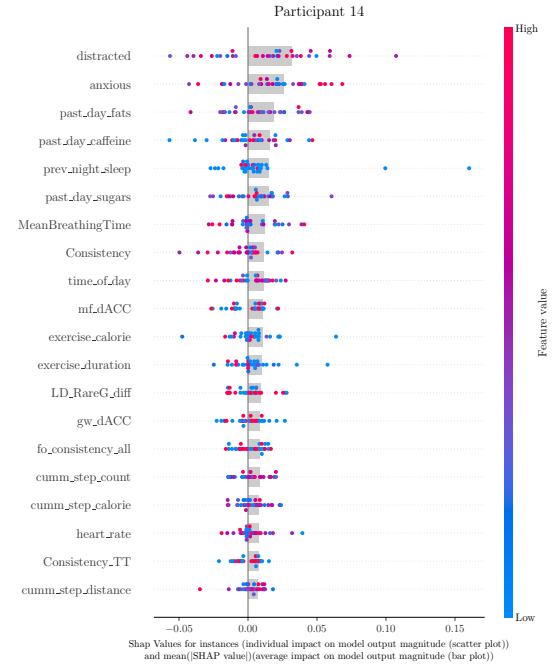
Fig. 6.6 SHAP summary plots for all participants.

higher *past-day-caffeine* values are correlated with a decrease in model prediction, while higher *cumm-step-count* increases the model's prediction.

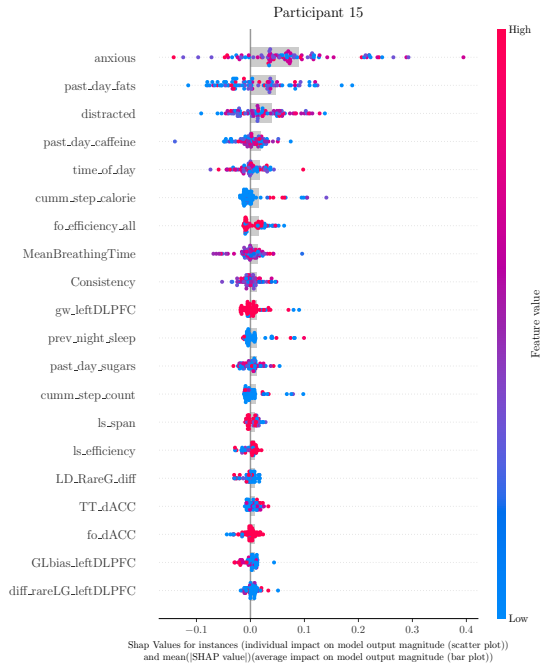
However, the results for Participant 24 (Figure 6.6m) show a much less coherent correlation between the features and the predicted value. Although the SNN-MLP for Participant 24 achieved the lowest MAE of the entire cohort (0.208), its feature rankings were highly irregular. The top contributing features were almost completely collected from a single lab-based neurocognitive assessment, including *MeanBreathingTimeTT mf-consistency, mf-leftDLPFC* and *mf-efficiency*. Given that these features were static metrics recorded during a lab session, they should theoretically lack the temporal variance required to predict mood fluctuations recorded four times daily. The explanation for this discrepancy lies in the underlying label distribution for Participant 24, where the label 1 (the lowest depression level) was recorded over ten times more frequently than all other labels combined (see: Figure 6.1). Therefore, the model likely achieved a low MAE by simply memorising the majority class rather than learning an authentic underlying pattern. This is further evidenced by comparing these results to Participant 10, whose model produced a much higher MAE of 0.828 (Table 6.8) yet yielded more diverse and physiologically relevant feature contributions. From this, we can confirm that a smaller MAE does not necessarily lead to more extensively clear or useful



(c) Participant 12 (Deletion)



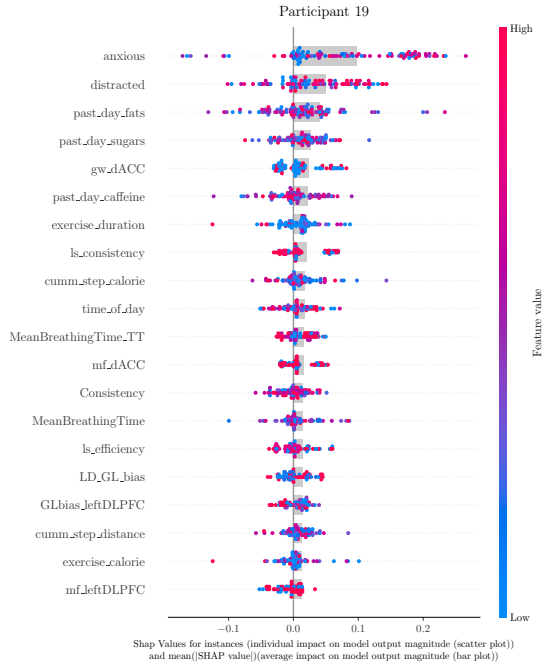
(d) Participant 14 (Deletion)



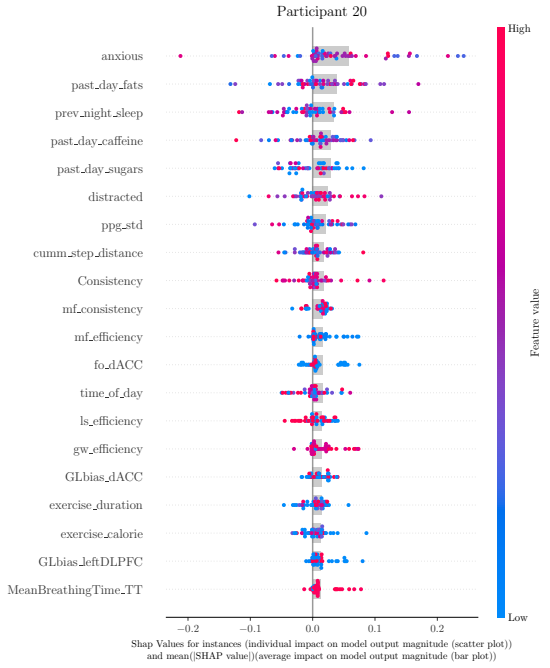
(e) Participant 15 (Manual)

(f) Participant 18 (Manual)

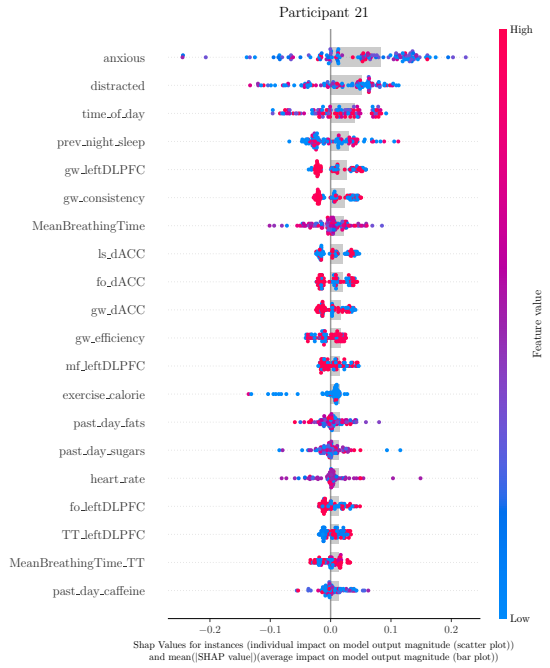
Fig. 6.6 *Cont.*



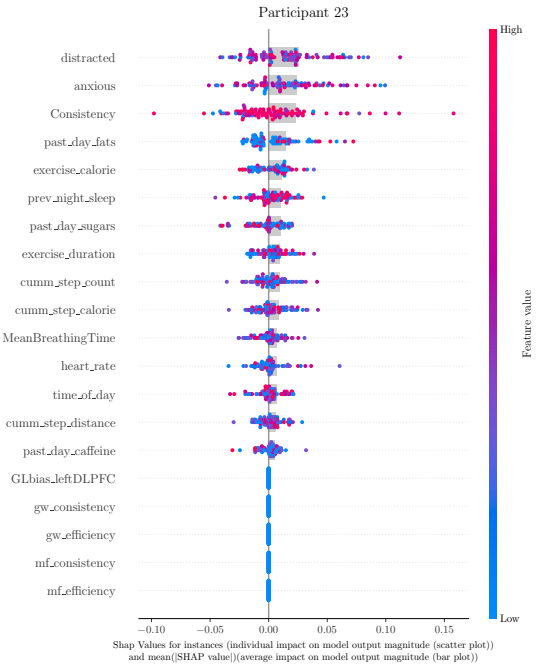
(g) Participant 19 (Deletion)



(h) Participant 20 (Deletion)

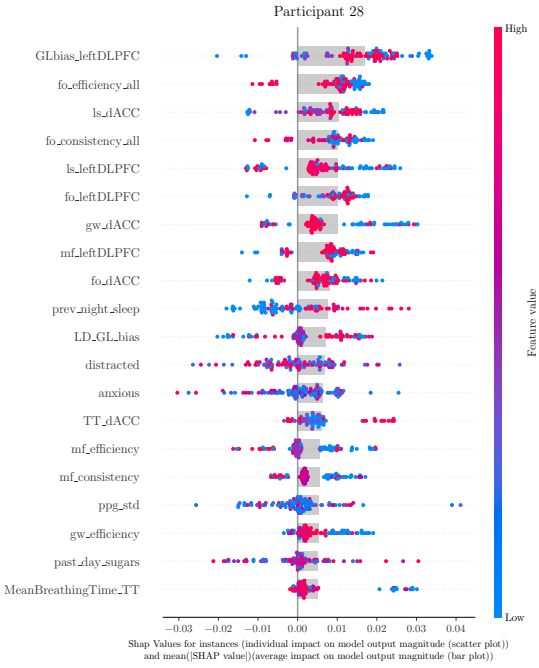


(i) Participant 21 (Manual)

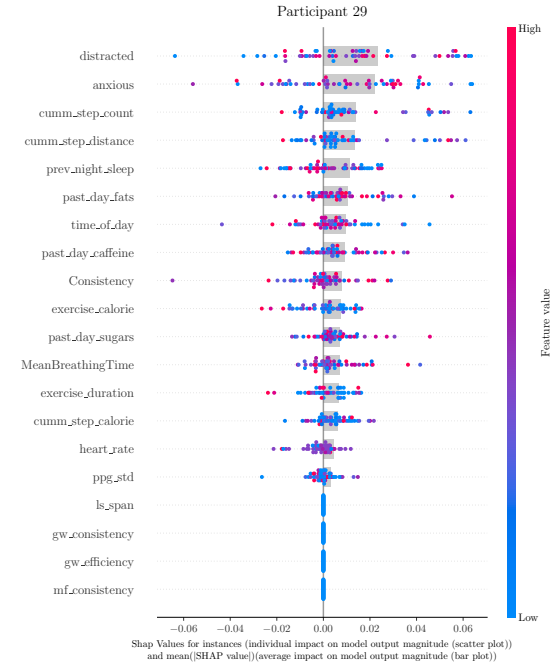


(j) Participant 23 (Manual)

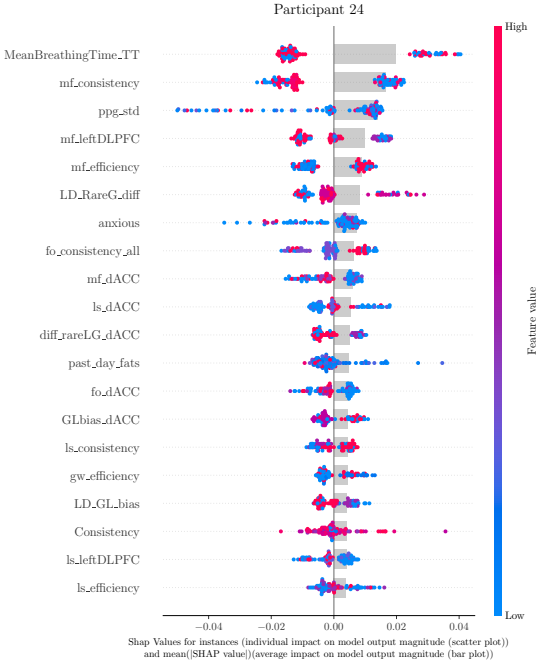
Fig. 6.6 *Cont.*



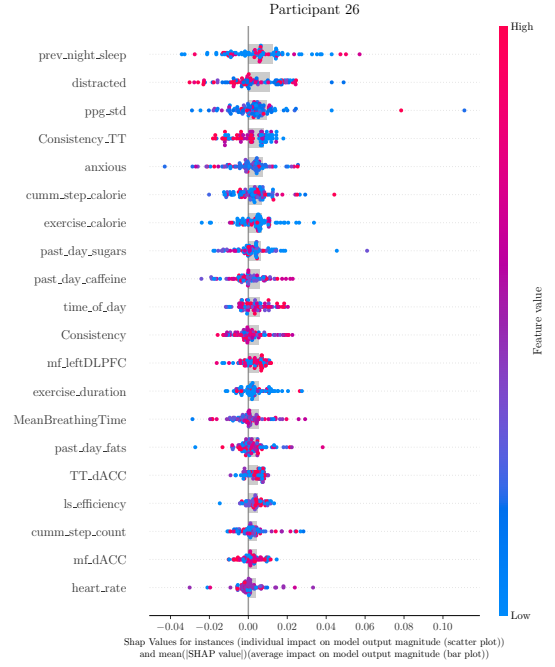
(k) Participant 28 (Deletion)



(l) Participant 29 (Deletion)



(m) Participant 24 (Manual)



(n) Participant 26 (Deletion)

Fig. 6.6 *Cont.*

results given that the dataset has a significant imbalance, resulting in feature explanations that lack practical validity.

Figure 6.7 presents the top 10 features found for all participants, ranked by their frequency of appearance within the top 10 contributing features for each participant. From this, the features collected alongside the *depressed* feature during the EMAs (*anxious, distracted, MeanBreathingTime* and *Consistency*) have the highest impact on the model predictions. Specifically, the mood-related features *distracted* and *anxious* are the most frequently present as top contributors. The high contribution of these features is likely due to their temporal proximity to the target label, capturing the immediate psychological state of the participant. This aligns with the findings from Chatterjee et al. (2024); Shah et al. (2021), which similarly identified co-morbid anxiety and mood ratings as the most powerful predictors in personalised modelling, and therefore validates our approach as an alignment to both results.

Following the mood ratings, the lifestyle of the participants are also shown to have a significant impact on depression levels. The diet of the participants (fats, caffeine, sugar) and sleep duration were identified as secondary drivers, consistently appearing in the top rankings for the majority of the cohort. The *time of day* also was also quite influential for some of the participants as well. Interestingly, the only feature retrieved via the smartwatch to consistently impact the models was *cumm-step-calorie*. Other physiological metrics from the wearable devices showed comparatively minimal influence on the classification outcomes. While neurocognitive features appeared prominently in specific individual models (notably Participant 1), their impact across the broader cohort was minimal, failing to appear in the global top 10 rankings. This reinforces the earlier assumption that the one-time records of data – such as neurocognitive assessments performed in a lab setting – are less likely to contribute meaningfully towards the predictive power than features recorded consistently throughout the month.

With the initial validity of the model's logic established, the next section will focus on directly comparing the results of the current experiment to the benchmark findings by Chatterjee et al..

6.4.2 Comparing With Previous Findings

To evaluate the clinical validity of our findings, we compared our feature rankings with those reported in the benchmark study by Chatterjee et al. (2024). While our initial global analysis included neurocognitive data, the original study focused exclusively on features derived from wearables and the EMAs identify interventions applicable to a participant's immediate personal environment.

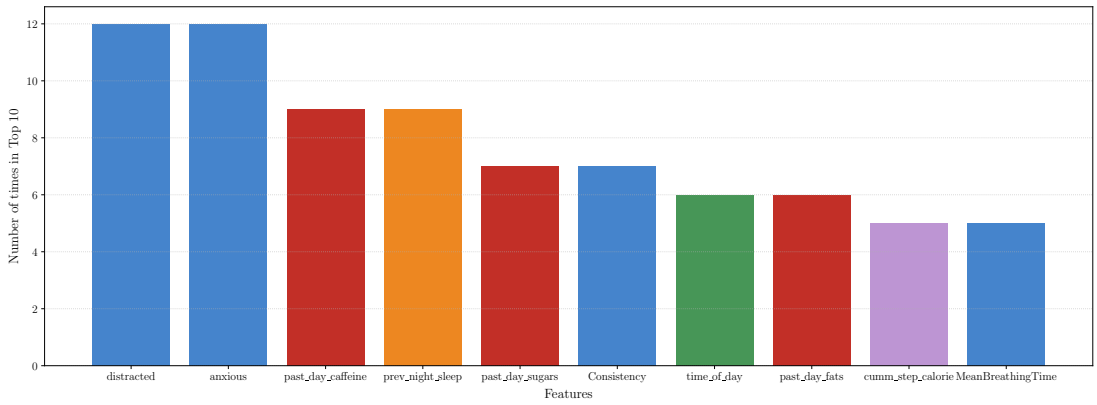
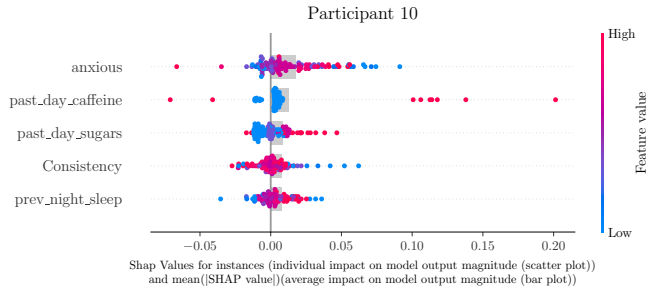


Fig. 6.7 Top 10 features for all participants. The features are arranged based on the frequency of their appearance in the top-10 features. The groups contain the following features. Mood and Stress Assessment (Blue): *anxious, distracted, MeanBreathingTime* and *Consistency*; Diet (Red): *past-day-fats, past-day-sugars* and *past-day-caffeine*; Sleep (Orange): *prev-night-sleep*; Time (Green): *time-of-day*; and Activity (Purple): *cumm-step-calorie*.

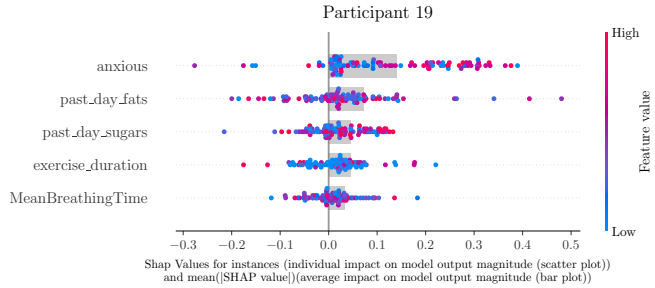
Although limiting the feature space can reduce the representativeness of the explanations for the model's predictions, the findings in Section 6.4.1 showed that neurocognitive data had a minimal impact on global predictions. Therefore, to ensure a fair comparison, a secondary set of explanations was generated using a reduced feature set that mirrors the parameters of the original study (see Table 6.2 for details of the retained features). For consistency with the benchmark's reporting style, we have also restricted this comparison to the top five contributing features.

Figure 6.8 presents these top five features for a selection of participants. We specifically highlight Participants 10, 19, and 24, as these individuals were used as the primary examples in the benchmark findings. The complete set of SHAP plots using the reduced feature set can be found in Appendix C.

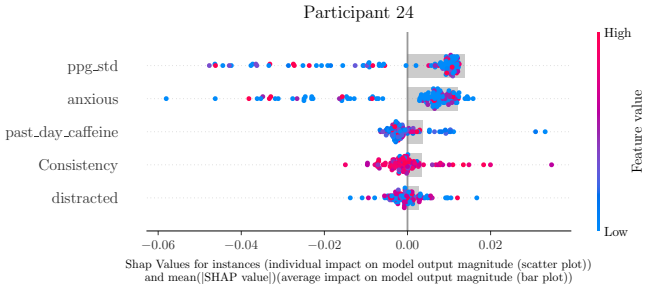
With Participant 10, the top five features from the benchmark were: *anxious, past-day-sugars, MeanBreathingTime, distracted* and *exercise-calorie*. Our results, as shown in Figure 6.8a (using the same subsampled features) partially aligns with this, where two features – *anxious* and *past-day-sugars* are retained as top contributors. The top five features for Participant 19 were *past-day-caffeine, past-day-sugars, ppg-std, anxious* and *distracted* in the benchmark, out of which *anxious* and *past-day-sugars* remained in the top five contributors (see Figure 6.8b). Lastly, for participant 24, the most influential features in the benchmark results were: *past-day-caffeine, past-day-sugars, distracted, cumm-step-calorie* and *past-day-fats*. In Figure 6.8c we can see that some of these features were retained – *past-day-caffeine* was placed third, and *distracted* came fifth. The partial alignment between



(a) Participant 10 (Manual)



(b) Participant 19 (Deletion)



(c) Participant 24 (Automatic)

Fig. 6.8 SHAP summary plots for Participants 10, 19 and 24.

the baseline results and our SNN-MLPs across Participants 10, 19, and 24 suggests that the SNN is effectively learning to recognise the same core features identified by the baseline MLPs.

In terms of the feature group rankings, in the current experiment the most influential factors for the model prediction were from features related to anxiety and stress, followed by the participant's diet, the physical activity, sleep and heart (see Figure 6.9). However, in the previous results the feature groups were in a slightly different order: *Diet, Anxiety and Breathing, Activity, Heart* and *Sleep*.

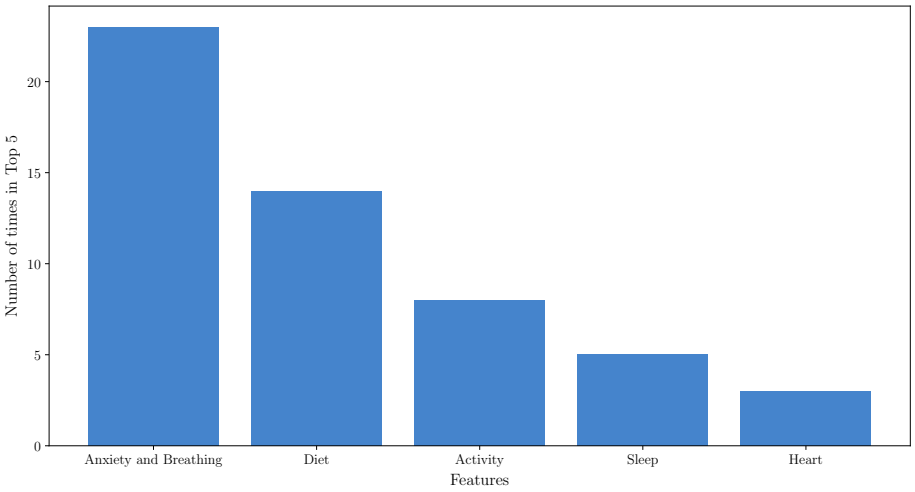


Fig. 6.9 Top 5 feature groups using the reduced feature set.

6.5 Discussion

The primary goal of this study was to demonstrate that the SNN-MLP architecture can function as a reliable surrogate for an SNN completing multi-class classifications for the purpose of explainability. Based on the similar MAE results and the consistent replication of the SNN's test behaviour by the SNN-MLP Section 6.3.1, this objective was successfully achieved. Although the subsequent feature explanations were not as universally stable as those generated in the previous case study, we successfully proved that the SNN-MLP still identified the same globally significant features (anxiety and diet) as the benchmark.

The different order of feature groups from Chatterjee et al. (2024), even when using the smaller subset of features confirm that our models have learnt different feature importance orders from the benchmarked MLP models in the previous work. However, the differences between the results does not necessarily invalidate our findings. As confirmed in Section 6.3.4, our model's MAE was highly comparative to the previous benchmark results, which indicated that the fully optimised models in the benchmark experiments also did not achieve a theoretical 'perfect' prediction.

Therefore, unlike the previous DDoS case study (where the baseline MLP model achieved 100% accuracy and therefore served as the ideal result to aim for), the benchmark feature rankings cannot be considered an absolute standard to align against exactly.

It can be argued that in instances where our model achieves a lower MAE than the benchmark, our generated explanations are more representative of the participant's true state. For example, with Participant 19, our model achieved a lower MAE (0.727) than the model in the previous work (0.989). In this instance, our top five features (presented in

Figure 6.8b) are a more accurate representation of the model's decision factors. Conversely, for Participant 24 the previous work's model had a much better MAE (0.128) than our own (0.209), suggesting their features were derived from a more accurate model. Therefore, considering the neurocognitive features may be redundant, as the features already do not align with the more accurate order presented in Chatterjee et al. (2024) (although, since they exclude the neurocognitive features from their SHAP value calculation this cannot be guaranteed).

Despite these challenges, the framework still successfully identified anxiety and dietary intake as influential features across nearly all participants, aligning with both our initial hypothesis and the expected results from previous psychiatric literature. These results confirm not only the successful SNN-MLP generation with multi-class classification, but also show the potential of the SNN architecture for personalised modelling. With a refined model or larger dataset, it could potentially produce both highly accurate predictions and highly stable, meaningful explanations with the SNN-MLP in the future.

Chapter 7

Conclusions

As Spiking Neural Networks (SNNs) gain traction in industrial and medical sectors, ensuring that their decision-making processes are transparent is essential for human safety. While traditional artificial neural networks have struggled with transparency for decades, the introduction of temporal processing and non-linear dynamics in spiking architectures has made this challenge twofold.

In Chapter 3 we explored the existing methods for Explainable Artificial Intelligence (XAI) with SNNs, and found that in most cases the suggested methods provide only shallow or implicit levels of interpretability. Additionally, current methods are largely restricted to local, single-instance explanations in the image classification domain, where static spatial pixels simplify the problem. Many of these approaches sacrifice generalisability by focusing on narrow tasks or restricting model architectures, while others reduce biological plausibility to achieve model transparency.

By contrast, the methodology proposed in this research maintains the biological essence of the SNN while providing a scalable and generalisable mapping pipeline. This was achieved through the development of a framework—outlined in Chapter 4—that converts generic spiking representations in Nengo into formal models (Neural Interchange Format (NIF)). These models are then used to build an SNN-based MLP (SNN-MLP) surrogate that retains the original characteristics of the spiking system within a differentiable and interpretable structure, compatible with generic XAI methods for Artificial Neural Networks (ANNs).

Chapter 5 presented the results of the first case study for Distributed Denial-of-Service (DDoS) detection, where we successfully demonstrated that the proposed translation approach is effective for single-dimensional SNNs used in binary classification. Additionally, through systematic model optimisation, we identified clear correlations between SNN hyperparameters (e.g., the learning rules, timesteps, data encoding methods) as well as their direct impact on model performance. The optimised spiking models exhibited strong predictive

capabilities, with accuracy levels ranging from 0.73 to 0.87 depending on the specific data encoding method and threshold settings applied. Most importantly, the explanations generated by the SNN-MLP surrogate aligned with both the established benchmarks and our local rebuilt benchmarks using Multi-Layer Perceptrons (MLPs). This alignment provided strong evidence that the surrogate model is an accurate reflection of the spiking network's logic.

However, several limitations were identified during this phase. As the SNN model only used a smaller subsample of the full dataset, the study did not explore the full depth of the DDoS dataset, which may have provided a more robust test of the model's limits. This was due to another limitation of the method – scalability issues – as training with larger samples and datasets significantly increased the simulation duration. This was, however, largely a result of the computational overhead required for temporal simulation rather than a flaw in the translation logic. Additionally, while the accuracy levels were acceptable, they remained slightly below the optimal levels. Finally, the use of a single-dimensional ensemble instead of a 2D argmax approach suggests that different architectural configurations could yield variations in the resulting explanations. Future research could therefore focus on refining the SNN architectures and configurations to achieve higher classification accuracy with reduced computational overhead, while also evaluating whether a two-dimensional output mapping provides more granular or robust explanations.

The second case study is presented in Chapter 6, where we confirmed that the translation method is adaptable to complex, personalised datasets and multi-dimensional SNNs. A significant achievement was that the translation method supported the implementation of epoch-based learning within the Nengo environment, which allowed the SNN-MLP to correctly replicate the behaviour of the SNN at each stage of the learning process. The methodology also proved to be compatible with k-fold cross-validation, demonstrating its reliability for clinical data processing. The limitations of this study were primarily centred on model performance and data complexity. The overall accuracy across the entire participant group was not consistently high, which mirrors the garbage-in, garbage-out challenges found in previous literature regarding sparse or noisy clinical data (C. Weyerer and F. Langer, 2019; Chomicki et al., 2025).

Additionally, the model required a much larger number of neurons (300 to 80) to maintain performance compared to the first case study (20 to 5). Finally, we found that using Mean Average Error (MAE) as the sole metric to decide on the surrogate's validity is insufficient, as a low error rate does not necessarily guarantee that the resulting Shapley Additive ExPlanation (SHAP) distributions will be identical.

However, despite the inherent difficulty of predicting mental health states, the SNN-MLPs still outperformed fully optimised traditional MLPs for several participants, showing the

potential for SNNs to capture nuanced physiological and neurocognitive patterns better than their abstract counterparts. Future work would involve testing the framework against verified multi-class datasets to ensure that multi-dimensional SNN conversions remain accurate when identifying the subtle boundaries between several different categories of data. To improve model performance and reliability, future experiments could also explore the impact of varying ensemble sizes tailored to individual participants. This personalised architectural approach could mitigate the performance issues noted for the participants with a lot of missing samples or severely imbalanced dataset, where a one-size-fits-all model size was occasionally insufficient. Additionally, refining the data preprocessing pipeline by removing highly correlated features (such as the neurocognitive data) or exploring alternative data preprocessing methods could reduce noise and lead to more stable explanations. Furthermore, the inclination distributions between classes for the models' predictions could also be explored, allowing clinicians to see not just the final prediction, but the model's level of uncertainty or bias toward specific diagnostic outcomes.

7.1 Limitations and Future Work

Although our work presents a robust proof of concept for explaining complex SNNs, there are still some limitations with the proposed approach. A primary conceptual limitation lies in the fundamental nature of surrogate modelling. Similar to other methods examined in our literature review, such as those proposed by Kamal et al. (2022) and Sai et al. (2024), we face the question of whether a surrogate explanation truly captures the internal causal dynamics of the original SNN, or merely approximates its input-output behaviour. We mitigated this concern by implementing a custom activation function within the surrogate SNN-MLP that mimics the spiking mechanism, and by capturing the progressive weight updates across training epochs. Furthermore, our results for depression prediction (Chapter 6) demonstrated that despite the challenges of sparse and highly individualised data, our SNN and the surrogate SNN-MLP exhibited consistent behavioural alignment. Even in instances where our models produced incorrect classifications, they reliably made the same errors which provided strong evidence that our SNN-MLP successfully learned the underlying decision-making behaviour of the SNN, rather than defaulting to a high-level input-output mapping. Nevertheless, the explanations are still inherently derived from a surrogate rather than the original network. Future research could explore the development of temporal surrogate models that more closely align with spiking dynamics, or expand upon our current framework to allow for the visualisation of dynamic weight activations within the SNN-MLP to compare against that of the SNN.

Additionally, the exploration into how our methodological choices and hyperparameters directly influence the final explanations is minimal at this stage. For example, in Chapter 5, we optimised hyperparameters such as the learning rate, the simulation timestep, and the learning rule primarily to maximise classification accuracy for DDoS classification. We operated under the assumption that a more accurate SNN would inherently produce more relevant and usable explanations, and thus generated explanations only under the ideal configurations locked for each hyperparameter. Additionally, while we noted that different data encoding strategies produced variations in the resulting explanations, we did not deeply analyse the precise mechanisms through which encoding influences interpretability. However, considering the accuracy patterns observed in the heatmaps and the varying feature explanations, there would undoubtedly be a benefit to examining the impact of these hyperparameters in more detail. Such an investigation would be highly beneficial for verifying the reliability and flexibility of our surrogate explanation framework across diverse operational conditions.

The absence of established domain expertise to validate our generated explanations is yet another limitation. Unlike mainstream datasets, which often have extensive verification baselines established using traditional ANNs, our work relied primarily on comparisons against a single benchmark. Because our model-generated explanations cannot currently be measured against a definitive quantitative metric (aside from a single benchmark), having a human expert verify the results against clinical or technical ground truths would have significantly strengthened our conclusions. For example, in Chapter 5 having expert validation confirming that the *Inbound* feature is truly the most influential factor in identifying a DDoS attack, as well as any reasonings or connections with other features would have increased the depth of the insight for the explanations, and strengthened the validation for our approach. This limitation was even more pronounced in our second case study on depression prediction. Involving a domain expert would have been invaluable for our feature selection and reduction, particularly given the challenges associated with the dataset's structure, which suffered from a significant scarcity of samples per participant and high inconsistency across feature types. For example, this dataset combined continuous aggregated measures, such as cumulative step counts, with singular point-in-time recordings, such as performance efficiency in a neurocognitive assessment. It is highly unlikely that features with such differing temporal scopes possess equivalent interpretative weight. Future work would thus ideally include another experiment with multi-class classification using datasets that provide a larger volume of samples and greater consistency among the recorded features for additional validation. Furthermore, integrating domain experts into the analytical process will be essential to ensure that our identified feature importances align accurately with established physiological and behavioural knowledge.

Regardless, the success of the SNN-MLP surrogate model provides a foundation for several advanced research pathways beyond this work. In the medical field, the ability to interpret model-agnostic SNNs enables the development of sophisticated, privacy-preserving wearables. Because SNNs are highly efficient, they can run locally on low-power embedded devices rather than sending sensitive patient data to the cloud for processing (Hunsberger and Eliasmith, 2016). This framework ensures that these local models are not just efficient but also clinically accountable. It empowers healthcare providers to use AI-driven diagnostics for chronic condition management, such as epilepsy or heart disease monitoring, with the confidence that the model's flags are based on medically relevant features rather than statistical noise.

Another potential direction could be towards using a precise quantification of SHAP feature contribution values to establish a formal metric for the explanation accuracy itself. While current results show qualitative alignment with benchmarks, a mathematical verification of the surrogate's fidelity would provide the rigorous accountability required for clinical deployment. For example, with the second case study, this could be used to examine the individual samples and how correctly each feature is represented, as several participants had close to no observable trends between the original input features and the produced results. Ultimately, developing a quantifiable way to measure the 'correctness' of an explanation could lead to a more unified and standardised definition of XAI as well, and lead to more focused effort towards understanding Artificial Intelligence (AI) models.

Finally, the potential for real-time applications remains a significant frontier for this work. Future research could extend the mapping pipeline to accommodate unsupervised learning architectures such as NeuCube, which model the communicative connectivity between different brain regions. By applying explainability to these highly dynamic systems, it may be possible to map how specific features trigger interactions across the network in real time. Exploring the use of even simpler linear models as surrogates could also be investigated to determine if the complex logic of an SNN can be reduced to a fully transparent set of rules, thereby eliminating the black box problem entirely while retaining the energy-efficient benefits of neuromorphic hardware.

References

Abbott, L. F. and Nelson, S. B. (2000). Synaptic plasticity: Taming the beast. *Nature Neuroscience*, 3(11):1178–1183.

Abeyagunasekera, S. H. P., Perera, Y., Chamara, K., Kaushalya, U., Sumathipala, P., and Senaweera, O. (2022). LISA : Enhance the explainability of medical images unifying current XAI techniques. In *2022 IEEE 7th International Conference for Convergence in Technology (I2CT)*, pages 1–9.

Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A., and Arshad, H. (2018). State-of-the-art in artificial neural network applications: A survey. *Heliyon*, 4(11).

Abouhassan, I., Kasabov, N., Popov, G., and Trifonov, R. (2022). Why Use Evolving Neuro-Fuzzy and Spiking Neural Networks for incremental and explainable learning of time series? A case study on predictive modelling of trade imports and outlier detection. In *2022 IEEE 11th International Conference on Intelligent Systems (IS)*, pages 1–7. ISSN: 2767-9802.

AbouHassan, I., Kasabov, N. K., Jagtap, V., and Kulkarni, P. (2023). Spiking neural networks for predictive and explainable modelling of multimodal streaming data with a case study on financial time series and online news. *Scientific Reports*, 13(1):18367.

Ahmed, I., Jeon, G., and Piccialli, F. (2022). From Artificial Intelligence to Explainable Artificial Intelligence in Industry 4.0: A Survey on What, How, and Where. *IEEE Transactions on Industrial Informatics*, 18(8):5031–5042.

Ahmed, S., Bisht, P., Mula, R., and Dhavala, S. S. (2021). A Deep Learning framework for Interoperable Machine Learning. In *The First International Conference on AI-ML-Systems*, pages 1–7, Bangalore India. ACM.

Alqadhi, S., Mallick, J., Alkahtani, M., Ahmad, I., Alqahtani, D., and Hang, H. T. (2024). Developing a hybrid deep learning model with explainable artificial intelligence (XAI) for enhanced landslide susceptibility modelling and management. *Natural Hazards*, 120(4):3719–3747.

Banerjee, S., Ghosh, S., Banerjee, A., and Mohalik, S. K. (2023). SMT-Based modelling and Verification of Spiking Neural Networks: A Case Study. In *Verification, Model Checking, and Abstract Interpretation*, pages 25–43. Springer Nature Switzerland. ISSN: 1611-3349.

Banerjee, S., Rounak, A., and Pakrashi, V. (2024). Adaptive Linear Control of a Cartpole Using Minimum Spiking Neurons Trained with Prescribed Error Sensitivity. In *2024 21st International Conference on Mechatronics - Mechatronika (ME)*, pages 1–9.

Batchu, R. K. and Seetha, H. (2022). An integrated approach explaining the detection of distributed denial of service attacks. *Computer Networks*, 216:109269.

Bekolay, T., Bergstra, J., Hunsberger, E., DeWolf, T., Stewart, T. C., Rasmussen, D., Choo, X., Voelker, A., and Eliasmith, C. (2014). Nengo: A Python tool for building large-scale functional brain models. *Frontiers in Neuroinformatics*, 7.

Bitar, A., Rosales, R., and Paulitsch, M. (2023). Gradient-based feature-attribution explainability methods for spiking neural networks. *Frontiers in Neuroscience*, 17. Publisher: Frontiers.

Braun, J. (2021). *THEORETICAL NEUROSCIENCE I Lecture 3: Leaky-integrate-and-fire model Cognitive Biology Group Content*. Bernstein Network Computational Neuroscience.

C. Weyerer, J. and F. Langer, P. (2019). Garbage In, Garbage Out: The Vicious Cycle of AI-Based Discrimination in the Public Sector. In *Proceedings of the 20th Annual International Conference on Digital Government Research*, pages 509–511, Dubai United Arab Emirates. ACM.

Cachi, P. G., Soto, S. V., and Cios, K. J. (2023). TM-SNN: Threshold Modulated Spiking Neural Network for Multi-task Learning. In *Advances in Computational Intelligence*, pages 653–663. Springer Nature Switzerland. ISSN: 1611-3349.

Chalkiadakis, G., Elkind, E., and Wooldridge, M. (2011). *Computational Aspects of Cooperative Game Theory*. Morgan & Claypool Publishers.

Chandra, H., Ghosh, A., Singh, R., and Srinivas, M. B. (2024). SNN-LIF Model for Glaucoma Classification. In *2024 IEEE 6th International Conference on AI Circuits and Systems (AICAS)*, pages 114–118.

Chatterjee, S., Mishra, J., Sundram, F., and Roop, P. (2024). Towards Personalised Mood Prediction and Explanation for Depression from Biophysical Data. *Sensors*, 24(1):164.

Chomicki, A., Wójcik, F., and Dudycz, H. (2025). Assessing the impact of dataset quality on the performance of artificial intelligence models in automatic waste classification. *Procedia Computer Science*, 270:1061–1070.

Chu, H., Yan, Y., Gan, L., Jia, H., Qian, L., Huan, Y., Zheng, L., and Zou, Z. (2022). A Neuromorphic Processing System With Spike-Driven SNN Processor for Wearable ECG Classification. *IEEE Transactions on Biomedical Circuits and Systems*, 16(4):511–523.

Dandl, S., Molnar, C., Binder, M., and Bischl, B. (2020). Multi-Objective Counterfactual Explanations. In *Parallel Problem Solving from Nature - PPSN XVI*, pages 448–469. Springer International Publishing. ISSN: 1611-3349.

Das, A. and Rad, P. (2020). Opportunities and Challenges in Explainable Artificial Intelligence (XAI): A Survey. arXiv:2006.11371 [cs].

Devireddy, K. (2025). A Comparative Study of Explainable AI Methods: Model-Agnostic vs. Model-Specific Approaches.

Doborjeh, M., Doborjeh, Z., Kasabov, N., Barati, M., and Wang, G. Y. (2021). Deep Learning of Explainable EEG Patterns as Dynamic Spatiotemporal Clusters and Rules in a Brain-Inspired Spiking Neural Network. *Sensors*, 21(14):4900. Number: 14Publisher: Multidisciplinary Digital Publishing Institute.

Elbrecht, D. and Schuman, C. (2020). Neuroevolution of Spiking Neural Networks Using Compositional Pattern Producing Networks. In *International Conference on Neuromorphic Systems 2020*, ICONS 2020, pages 1–5, New York, NY, USA. Association for Computing Machinery.

Eliasmith, C. and Anderson, C. H. (2003). *Neural Engineering: Computation, Representation, and Dynamics in Neurobiological Systems*. MIT Press.

Ferrand, R., Baronig, M., Limbacher, T., and Legenstein, R. (2023). Context-Dependent Computations in Spiking Neural Networks with Apical Modulation. In *Artificial Neural Networks and Machine Learning - ICANN 2023*, pages 381–392. Springer Nature Switzerland. ISSN: 1611-3349.

Ford, J., Thomas, F., Byng, R., and McCabe, R. (2020). Use of the Patient Health Questionnaire (PHQ-9) in Practice: Interactions between patients and physicians. *Qualitative Health Research*, 30(13):2146–2159.

Frosio, I. and Kautz, J. (2019). Statistical Nearest Neighbors for Image Denoising. *IEEE Transactions on Image Processing*, 28(2):723–738.

Gamoura, S. C. (2023). Explainable AI (XAI) for AI-Acceptability: The Coming Age of Digital Management 5.0. In *2023 IEEE International Conference on Networking, Sensing and Control (ICNSC)*, volume 1, pages 1–6. ISSN: 2766-8665.

Gerum, R. C. and Schilling, A. (2020). Spiking machine intelligence: What we can learn from biology and how spiking neural networks can help to improve machine learning. *CoRR*, abs/2004.13532.

Gerum, R. C. and Schilling, A. (2021). Integration of Leaky-Integrate-and-Fire Neurons in Standard Machine Learning Architectures to Generate Hybrid Networks: A Surrogate Gradient Approach. *Neural Computation*, 33(10):2827–2852.

Ghosh, A., Nowotny, T., and Knight, J. C. (2023). Insect-inspired Spatio-temporal Downsampling of Event-based Input. In *Proceedings of the 2023 International Conference on Neuromorphic Systems*, ICONS '23, pages 1–5, New York, NY, USA. Association for Computing Machinery.

Hamad, R. A., Kimura, M., Yang, L., Woo, W. L., and Wei, B. (2021). Dilated causal convolution with multi-head self attention for sensor human activity recognition. *Neural Computing and Applications*, 33(20):13705–13722.

Harbour, D. A., Cohen, K., Harbour, S. D., Ratliff, B., Henderson, A., Pennel, H., Schlager, S., Taha, T. M., Yakopcic, C., Asari, V. K., Boril, J., Sultana, A., Abouzahra, S. N., Bulter, A., and Prikkel, C. (2024). Martian Flight: Enabling Motion Estimation of NASA's Next-Generation Mars Flying Drone by Implementing a Neuromorphic Event-Camera and Explainable Fuzzy Spiking Neural Network Model. In *2024 AIAA DATC/IEEE 43rd Digital Avionics Systems Conference (DASC)*, pages 1–10. ISSN: 2155-7209.

Hassan, I. Y. A. and Kasabov, N. K. (2025). NeuDen: a framework for the integration of neuromorphic evolving spiking neural networks with dynamic evolving neuro-fuzzy systems for predictive and explainable modelling of streaming data. *Evolving Systems*, 16(1):3.

Hodges, J. L. (1958). The significance probability of the smirnov two-sample test. *Arkiv för Matematik*, 3(5):469–486.

Hodgkin, A. L. and Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, 117(4):500–544.

Huang, T., Ou, X., Yang, H., Hu, S., Geng, J., Hu, J., and Xu, Z. (2024). Remembering is Not Applying: Interpretable Knowledge Tracing for Problem-solving Processes. In *Proceedings of the 32nd ACM International Conference on Multimedia*, MM '24, pages 3151–3159, New York, NY, USA. Association for Computing Machinery.

Hunsberger, E. and Eliasmith, C. (2016). Training Spiking Deep Networks for Neuromorphic Hardware.

Hussain, I. and Thounaojam, D. M. (2024). An Extensive Review of the Supervised Learning Algorithms for Spiking Neural Networks. In *Big Data, Machine Learning, and Applications*, pages 63–80. Springer, Singapore.

Iverson, G. L. (2011). Z Scores. In *Encyclopedia of Clinical Neuropsychology*, pages 2739–2740. Springer, New York, NY.

Izhikevich, E. (2003). Simple model of spiking neurons. *IEEE Transactions on Neural Networks*, 14(6):1569–1572.

Javeed, D., Gao, T., Kumar, P., and Jolfaei, A. (2024). An Explainable and Resilient Intrusion Detection System for Industry 5.0. *IEEE Transactions on Consumer Electronics*, 70(1):1342–1350.

Jeong, Y. H., Hwang, S., and Chae, D.-K. (2024). HiLite: Hierarchical Level-implemented Architecture Attaining Part-Whole Interpretability. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, CIKM '24, pages 983–993, New York, NY, USA. Association for Computing Machinery.

Jeyasothy, A., Sundaram, S., Ramasamy, S., and Sundararajan, N. (2019a). A novel method for extracting interpretable knowledge from a spiking neural classifier with time-varying synaptic weights. arXiv:1904.11367 [cs].

Jeyasothy, A., Sundaram, S., and Sundararajan, N. (2019b). 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):1231–1240.

Jeyasothy, A., Suresh, S., Ramasamy, S., and Sundararajan, N. (2024). Development of a Novel Transformation of Spiking Neural Classifier to an Interpretable Classifier. *IEEE Transactions on Cybernetics*, 54(1):3–12. Conference Name: IEEE Transactions on Cybernetics.

Jia, S., Zuo, R., Zhang, T., Liu, H., and Xu, B. (2022). Motif-Topology and Reward-Learning Improved Spiking Neural Network for Efficient Multi-Sensory Integration. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8917–8921. ISSN: 2379-190X.

Jiang, X., Zhang, Q., Sun, J., Cao, J., Ma, J., and Xu, R. (2024). Fully Spiking Neural Network for Legged Robots.

Kamal, M. S., Chowdhury, L., Dey, N., Fong, S. J., and Santosh, K. (2021). Explainable AI to Analyze Outcomes of Spike Neural Network in Covid-19 Chest X-rays. In *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3408–3415. ISSN: 2577-1655.

Kamal, M. S., Chowdhury, L., Nimmy, S. F., Hasan Rafi, T. H., and Chae, D.-K. (2023). An Interpretable Framework for Identifying Cerebral Microbleeds and Alzheimer's Disease Severity using Multimodal Data. In *2023 45th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pages 1–4. ISSN: 2694-0604.

Kamal, M. S., Dey, N., Chowdhury, L., Hasan, S. I., and Santosh, K. (2022). Explainable AI for Glaucoma Prediction Analysis to Understand Risk Factors in Treatment Planning. *IEEE Transactions on Instrumentation and Measurement*, 71:1–9. Conference Name: IEEE Transactions on Instrumentation and Measurement.

Kariri, E., Louati, H., Louati, A., and Masmoudi, F. (2023). Exploring the Advancements and Future Research Directions of Artificial Neural Networks: A Text Mining Approach. *Applied Sciences*, 13:3186.

Kasabov, N. K. (2014). Neucube: A spiking neural network architecture for mapping, learning and understanding of spatio-temporal brain data. *Neural Networks*, 52:62–76.

Kasabov, N. K. (2019). Methods of spiking neural networks. In *Time-Space, Spiking Neural Networks and Brain-Inspired Artificial Intelligence*, pages 127–167. Springer Berlin Heidelberg, Berlin, Heidelberg.

Kasabov, N. K., Tan, Y., Doborjeh, M., Tu, E., Yang, J., Goh, W., and Lee, J. (2023). Transfer Learning of Fuzzy Spatio-Temporal Rules in a Brain-Inspired Spiking Neural Network Architecture: A Case Study on Spatio-Temporal Brain Data. *IEEE Transactions on Fuzzy Systems*, 31(12):4542–4552. Conference Name: IEEE Transactions on Fuzzy Systems.

Kaur, S. and Kaur, B. (2012). Neural networks and their applications. *INTERNATIONAL JOURNAL OF ELECTRONICS AND COMMUNICATION TECHNOLOGY (IJECT)*, 3(4).

Kayan, H., Heartfield, R., Rana, O., Burnap, P., and Perera, C. (2024). CASPER: Context-Aware IoT Anomaly Detection System for Industrial Robotic Arms. *ACM Trans. Internet Things*, 5(3):18:1–18:36.

Khan, N., Nauman, M., Almadhor, A. S., Akhtar, N., Alghuried, A., and Alhudhaif, A. (2024). Guaranteeing Correctness in Black-Box Machine Learning: A Fusion of Explainable AI and Formal Methods for Healthcare Decision-Making. *IEEE Access*, 12:90299–90316. Conference Name: IEEE Access.

Kharal, A. (2020). Explainable Artificial Intelligence Based Fault Diagnosis and Insight Harvesting for Steel Plates Manufacturing.

Khoei, T. T. and Kaabouch, N. (2024). A safe and reliable bayesian spiking neural network in industrial control systems. In *2024 IEEE 3rd International Conference on Computing and Machine Intelligence (ICMI)*, pages 1–7.

Kim, Y. and Panda, P. (2021). Visual explanations from spiking neural networks using inter-spike intervals. *Scientific Reports*, 11(1):19037.

Krichmar, J. L., Ketz, N. A., Pilly, P. K., and Soltoggio, A. (2022). Flexible Path Planning in a Spiking Model of Replay and Vicarious Trial and Error. In *From Animals to Animats 16*, pages 177–189. Springer International Publishing. ISSN: 1611-3349.

Kulkarni, S., Simon, S. P., and Sundareswaran, K. (2013a). A spiking neural network (SNN) forecast engine for short-term electrical load forecasting. *Applied Soft Computing*, 13(8):3628–3635.

Kulkarni, S., Simon, S. P., and Sundareswaran, K. (2013b). A spiking neural network (SNN) forecast engine for short-term electrical load forecasting. *Applied Soft Computing*, 13(8):3628–3635.

Kulkarni, S. R., Tabassum, A., Lim, S.-H., Schuman, C. D., Theilman, B. H., Rothganger, F., Wang, F., and Aimone, J. B. (2024). Explaining Neural Spike Activity for Simulated Bio-plausible Network through Deep Sequence Learning. In *2024 Neuro Inspired Computational Elements Conference (NICE)*, pages 1–7.

Kumar, A., Braud, T., Tarkoma, S., and Hui, P. (2020). Trustworthy AI in the Age of Pervasive Computing and Big Data. In *2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 1–6, Austin, TX, USA. IEEE.

Kumari, M., Chaudhary, A., and Narayan, Y. (2023). Explainable AI (XAI): A Survey of Current and Future Opportunities. In Hassanien, A. E., Gupta, D., Singh, A. K., and Garg, A., editors, *Explainable Edge AI: A Futuristic Computing Perspective*, pages 53–71. Springer International Publishing, Cham.

Lai, S., Hu, X., Xu, H., Ren, Z., and Liu, Z. (2023). Multimodal sentiment analysis: A survey. *Displays*, 80:102563.

Lin, W., Yi, H., and Li, X. (2023). Image Reconstruction and Recognition of Optical Flow Based on Local Feature Extraction Mechanism of Visual Cortex. In *International Conference on Neural Computing for Advanced Applications*, pages 18–32. Springer Nature Singapore. ISSN: 1865-0937.

Linardatos, P., Papastefanopoulos, V., and Kotsiantis, S. (2021a). Explainable AI: A Review of Machine Learning Interpretability Methods. *Entropy*, 23(1):18.

Linardatos, P., Papastefanopoulos, V., and Kotsiantis, S. (2021b). Explainable AI: A Review of Machine Learning Interpretability Methods. *Entropy*, 23(1):18. Number: 1 Publisher: Multidisciplinary Digital Publishing Institute.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Machlev, R., Perl, M., Belikov, J., Levy, K. Y., and Levron, Y. (2022). Measuring Explainability and Trustworthiness of Power Quality Disturbances Classifiers Using XAI—Explainable Artificial Intelligence. *IEEE Transactions on Industrial Informatics*, 18(8):5127–5137. Conference Name: IEEE Transactions on Industrial Informatics.

Maglogiannis, I. G., Firm, P., and Al, E. (2007). *Emerging artificial intelligence applications in computer engineering : real word AI systems with applications in eHealth, HCI, information retrieval and pervasive technologies*, volume 160. Ios Press, Amsterdam ; Washington, Dc.

Markov, K. (2023). Multilayer perceptron with backpropagation, hdl coder, and fpga technology: An integrated approach for efficient neural network implementation. *Problems of Engineering Cybernetics and Robotics*, 80.

Mikulasch, F. A., Rudelt, L., and Priesemann, V. (2023). Visuomotor Mismatch Responses as a Hallmark of Explaining Away in Causal Inference. *Neural Computation*, 35(1):27–37. Conference Name: Neural Computation.

Mishra, S., Sundaram, S., and Thousif, P. Md. (2025). A neuro-inspired approach for Continual Multi-Label Learning with evolving spiking networks. *Engineering Applications of Artificial Intelligence*, 160:111889.

Nasteski, V. (2017). An overview of the supervised machine learning methods. *HORIZONS.B*, 4:51–62.

Nathan Allen (2025). Neural-interchange-format/nif.py at main · nallen01/neural-interchange-format. https://github.com/nallen01/neural-interchange-format/blob/main/nif.py.

Nguyen, E., Nauta, M., Englebienne, G., and Seifert, C. (2023). Feature Attribution Explanations for Spiking Neural Networks. In *2023 IEEE 5th International Conference on Cognitive Machine Intelligence (CogMI)*, pages 59–68.

Ni, J., Chen, Y., Chen, Y., Zhu, J., Ali, D., Cao, W., Ni, J., Chen, Y., Chen, Y., Zhu, J., Ali, D., and Cao, W. (2020). A Survey on Theories and Applications for Self-Driving Cars Based on Deep Learning Methods. *Applied Sciences*, 10(8).

Ordóñez, F. J., Toledo, P. D., Sanchis, A., Ordóñez, F. J., Toledo, P. D., and Sanchis, A. (2013). Activity Recognition Using Hybrid Generative/Discriminative Models on Home Environments Using Binary Sensors. *Sensors*, 13(5):5460–5477.

Pargent, F., Schoedel, R., and Stachl, C. (2023). Best Practices in Supervised Machine Learning: A Tutorial for Psychologists. *Advances in Methods and Practices in Psychological Science*, 6(3):9.

Perdigão, D. (2024). Bayesian causal inference in deep spiking neural networks.

Pfeiffer, M. and Pfeil, T. (2018). Deep Learning With Spiking Neurons: Opportunities and Challenges. *Frontiers in Neuroscience*, 12.

Raghupathi, M. and Teja, N. S. P. (2024). An Explainable Ensemble Learning Model for Detection of Low-rate and High-rate DDoS Network Traffic.

Ribeiro, M. T., Singh, S., and 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*, pages 1135–1144, San Francisco California USA. ACM.

Robinson, J. (2014). Likert Scale. In *Encyclopedia of Quality of Life and Well-Being Research*, pages 3620–3621. Springer, Dordrecht.

Runyu, L., Xiaoling, L., and Jun, W. (2022). An Intelligent And Transparent Inference: Spiking Neural Network For Causal Reasoning. In *2022 19th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, pages 1–5. ISSN: 2576-8964.

Sai, S., Sharma, S., and Chamola, V. (2024). Explainable AI-empowered Neuromorphic Computing Framework for Consumer Healthcare. *IEEE Transactions on Consumer Electronics*, pages 1–1. Conference Name: IEEE Transactions on Consumer Electronics.

Schuler, N., Hoffmann, M., Beise, H.-P., and Bergmann, R. (2023). Semi-supervised Similarity Learning in Process-Oriented Case-Based Reasoning. In *Artificial Intelligence XL*, pages 159–173. Springer Nature Switzerland.

Seras, A. M., Del Ser, J., Lobo, J. L., Garcia-Bringas, P., and Kasabov, N. (2022). A Novel Explainable Out-of-Distribution Detection Approach for Spiking Neural Networks. arXiv:2210.00894 [cs].

Servanshi, P., Bindra, S. K., Gera, M., and Kaushal, R. (2021). Covid-19 Detection from CT-scan Images: Empirical Evaluation and Explainability. In *2021 Sixth International Conference on Image Information Processing (ICIIP)*, volume 6, pages 395–400. ISSN: 2640-074X.

Shah, R. V., Grennan, G., Zafar-Khan, M., Alim, F., Dey, S., Ramanathan, D., and Mishra, J. (2021). Personalized machine learning of depressed mood using wearables. *Translational Psychiatry*, 11(1):338.

Sharafaldin, I., Lashkari, A. H., Hakak, S., and Ghorbani, A. A. (2019). Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy. In *2019 International Carnahan Conference on Security Technology (ICCST)*, pages 1–8.

Shiffman, S. (2009). Ecological Momentary Assessment (EMA) in Studies of Substance Use. *Psychological assessment*, 21(4):486–497.

Simonyan, K., Vedaldi, A., and Zisserman, A. (2014). Deep inside convolutional networks: Visualising image classification models and saliency maps. (arXiv:1312.6034). arXiv:1312.6034 [cs].

Smith, C. U., Llado, C. M., Puigjaner, R., and Williams, L. G. (2007). Interchange Formats for Performance Models: Experimentation and Ouput. In *Fourth International Conference on the Quantitative Evaluation of Systems (QEST 2007)*, pages 91–100.

Steven Walczak, Narciso Cerpa, and Robert A. Meyers (2003). Artificial Neural Networks. In *Encyclopedia of Physical Science and Technology (Third Edition)*, pages 631–645. Academic Press, New York, third edition edition.

Sun, T., Yin, B., and Bohté, S. (2023). Efficient Uncertainty Estimation in Spiking Neural Networks via MC-dropout. In Iliadis, L., Papaleonidas, A., Angelov, P., and Jayne, C., editors, *Artificial Neural Networks and Machine Learning - ICANN 2023*, pages 393–406, Cham. Springer Nature Switzerland.

Sundararajan, M., Taly, A., and Yan, Q. (2017). Axiomatic attribution for deep networks. (arXiv:1703.01365). arXiv:1703.01365 [cs].

Szczęsny, S., Huderek, D., and Przyborowski, L. (2023). Explainable spiking neural network for real time feature classification. *Journal of Experimental & Theoretical Artificial Intelligence*, 35(1):77–92. Publisher: Taylor & Francis_eprint: https://doi.org/10.1080/0952813X.2021.1957024.

Talairach, J. and Szikla, G. (1980). Application of Stereotactic Concepts to the Surgery of Epilepsy. In Gillingham, F. J., Gybels, J., Hitchcock, E., Rossi, G. F., and Szikla, G., editors, *Advances in Stereotactic and Functional Neurosurgery 4*, pages 35–54, Vienna. Springer.

Tocchetti, A., Corti, L., Balayn, A., Yurrita, M., Lippmann, P., Brambilla, M., and Yang, J. (2024). A.I. Robustness: a Human-Centered Perspective on Technological Challenges and Opportunities. *ACM Comput. Surv.* Just Accepted.

Troyanskaya, O., Cantor, M., Sherlock, G., Brown, P., Hastie, T., Tibshirani, R., Botstein, D., and Altman, R. B. (2001). Missing value estimation methods for DNA microarrays. *Bioinformatics*, 17(6):520–525.

van Buuren, S. and Groothuis-Oudshoorn, K. (2011). Mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, 45(3):1–67.

Wei, Y., Jang-Jaccard, J., Sabrina, F., Singh, A., Xu, W., and Camtepe, S. (2021). AE-MLP: A Hybrid Deep Learning Approach for DDoS Detection and Classification. *IEEE Access*, 9:146810–146821.

Wei, Y., Jang-Jaccard, J., Singh, A., Sabrina, F., and Camtepe, S. (2023). Classification and Explanation of Distributed Denial-of-Service (DDoS) Attack Detection using Machine Learning and Shapley Additive Explanation (SHAP) Methods.

Wu, K., Yao, M., Chou, Y., Qiu, X., Yang, R., Xu, B., and Li, G. (2024). RSC-SNN: Exploring the Trade-off Between Adversarial Robustness and Accuracy in Spiking Neural Networks via Randomized Smoothing Coding. In *Proceedings of the 32nd ACM International Conference on Multimedia*, MM '24, pages 2748–2756, New York, NY, USA. Association for Computing Machinery.

Xie, K., Zhang, Z., Li, B., Kang, J., Niyato, D., Xie, S., and Wu, Y. (2022). Efficient Federated Learning With Spike Neural Networks for Traffic Sign Recognition. *IEEE Transactions on Vehicular Technology*, 71(9):9980–9992. Conference Name: IEEE Transactions on Vehicular Technology.

Xu, Z., Lv, Z., Chu, B., Sheng, Z., and Li, J. (2024). Progress and prospects of future urban health status prediction. *Engineering Applications of Artificial Intelligence*, 129:107573.

Yamazaki, K., Vo-Ho, V.-K., Bulsara, D., and Le, N. (2022). Spiking Neural Networks and Their Applications: A Review. *Brain Sciences*, 12(7):863.

Yue, B., Wang, K., Zhu, H., Yuan, X., and Yang, C. (2024). Spiking autoencoder for nonlinear industrial process fault detection. *Information Sciences*, 665:120389.

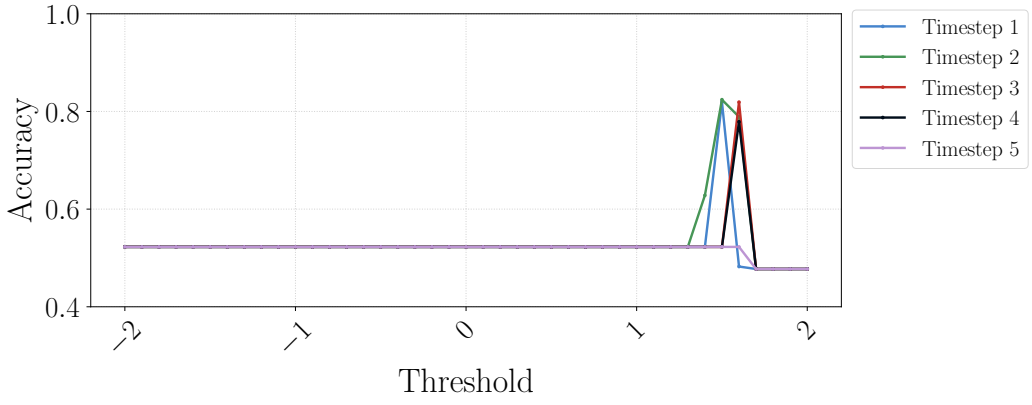
Zhang, J., Wu, C., Lu, K., and Gao, R. (2024). DCA: An Interpretable Deep Learning Model for Cancer Classification and New Knowledge Discovery Using Attention Mechanism with Discriminate Feature Constraint. In *Proceedings of the 2024 3rd International Symposium on Intelligent Unmanned Systems and Artificial Intelligence*, SIUSAI '24, pages 243–249, New York, NY, USA. Association for Computing Machinery.

Zhou, D., Chen, W., Chen, K., and Mi, B. (2023). Fast and Accurate SNN Model Strengthening for Industrial Applications. *Electronics*, 12(18):3845. Number: 18Publisher: Multidisciplinary Digital Publishing Institute.

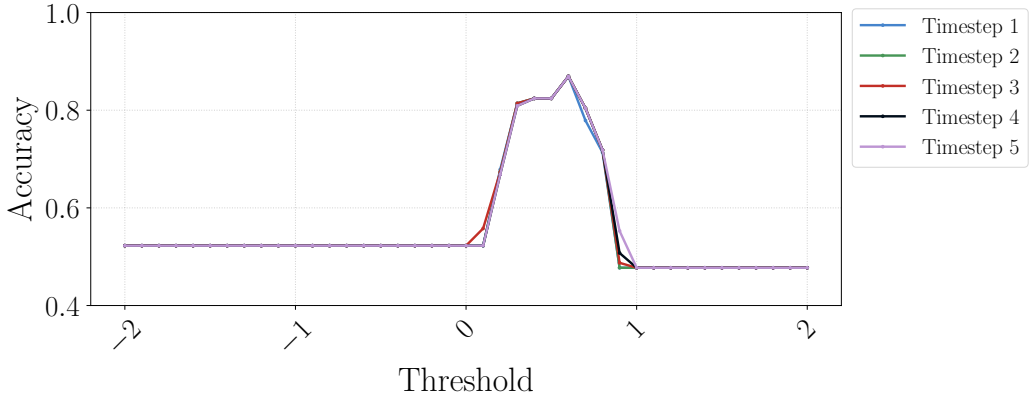
Appendix A

Accuracy Variations between Learning Rules: All Datasets

The following figures extend from Figure 5.3, and presents the variations of accuracy using Datasets 2, 3 and 4 across the threshold range $-2.0 - 2.0$ for the Spiking Neural Network (SNN) model trained using the Prescribed Error-Sensitivity (PES) and Recursive Least Squares (RLS) learning rules. For all datasets, the learning rate remained constant at $\eta = 0.001$, and $\Delta t = [1, 2, \ldots, 5]$ with both learning rules.

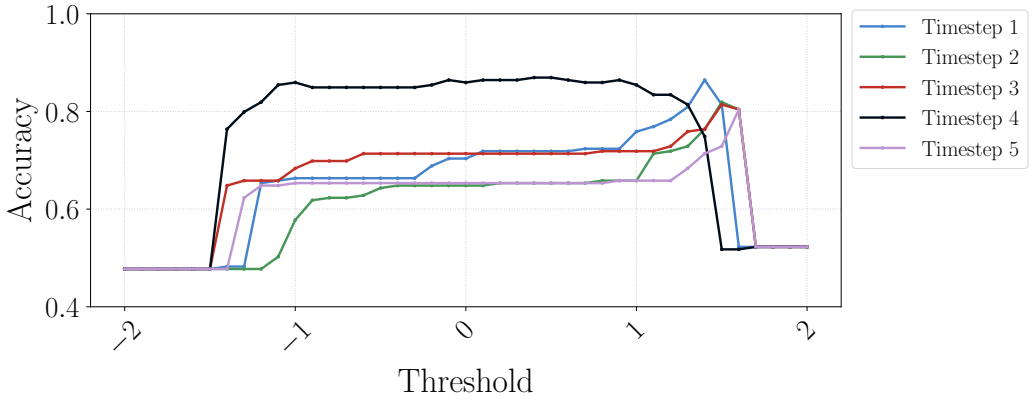


(a) PES

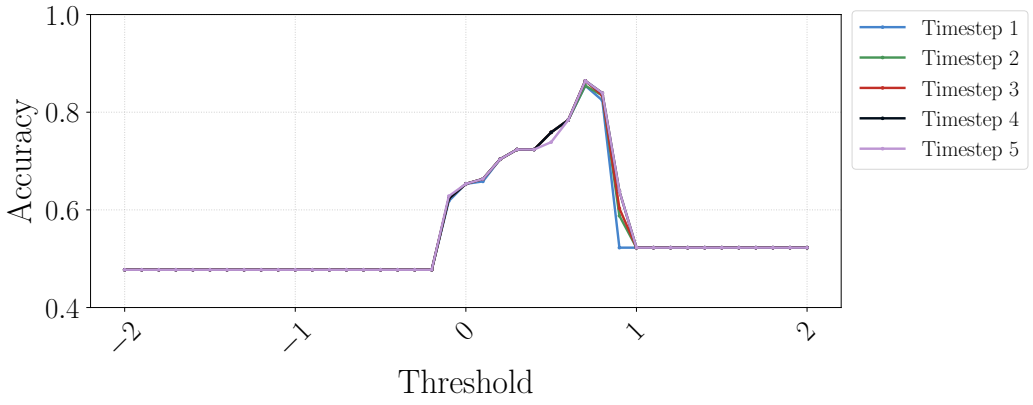


(b) RLS

Fig. A.1 Accuracy variations between PES and RLS (Dataset 2: benign = 0, attack = 1)

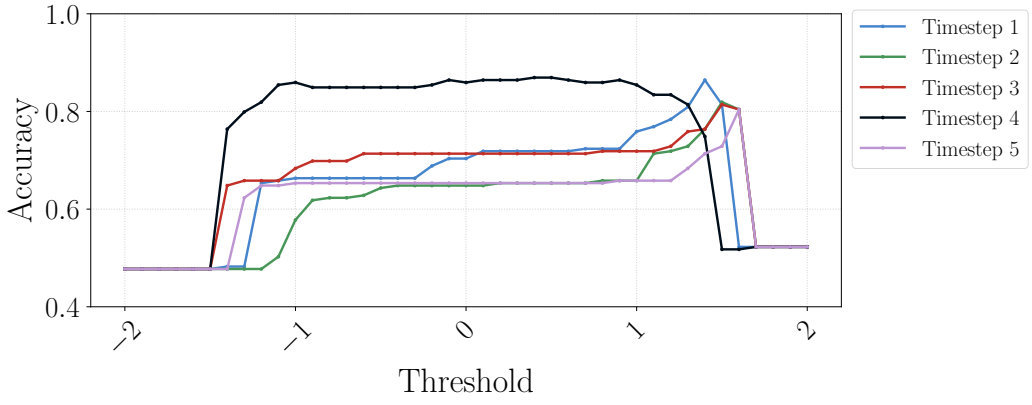


(a) PES

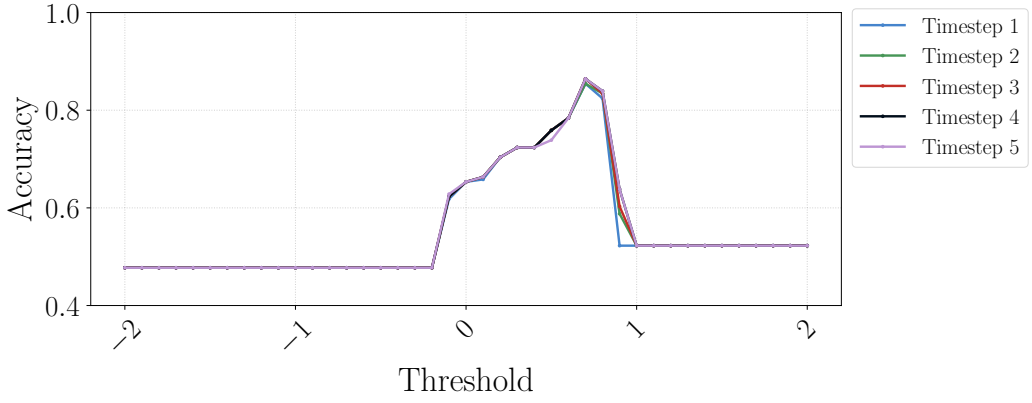


(b) RLS

Fig. A.2 Accuracy variations between PES and RLS (Dataset 3: benign = 1, attack = -1)



(a) PES



(b) RLS

Fig. A.3 Accuracy variations between PES and RLS (Dataset 4: benign = -1, attack = 1)

Appendix B

Train/Test Performances Across All Preprocessing Methods: All Participants

Expanding upon Section 6.3.2, this section presents the performance of both the Spiking Neural Networks (SNNs) and their corresponding SNN-based MLPs (SNN-MLPs) across different preprocessing methods. The figures illustrate the SNN models' training and testing performances over consecutive epochs, alongside the testing performance of the SNN-MLPs generated immediately after each training phase. Note that all plotted values represent averages obtained through 5-fold cross-validation.

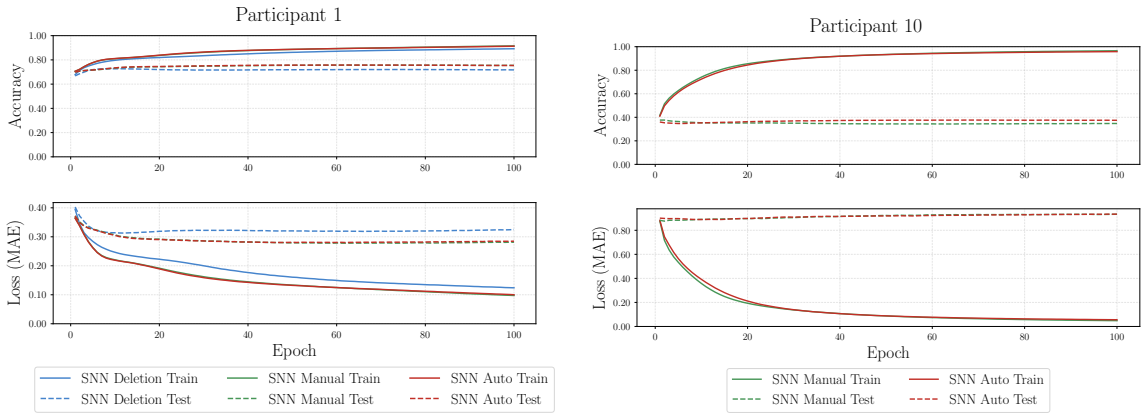


Fig. B.1 Accuracy and Mean Average Error (MAE) for SNNs and SNN-MLPs during training and testing at every epoch across all participants and preprocessing methods.

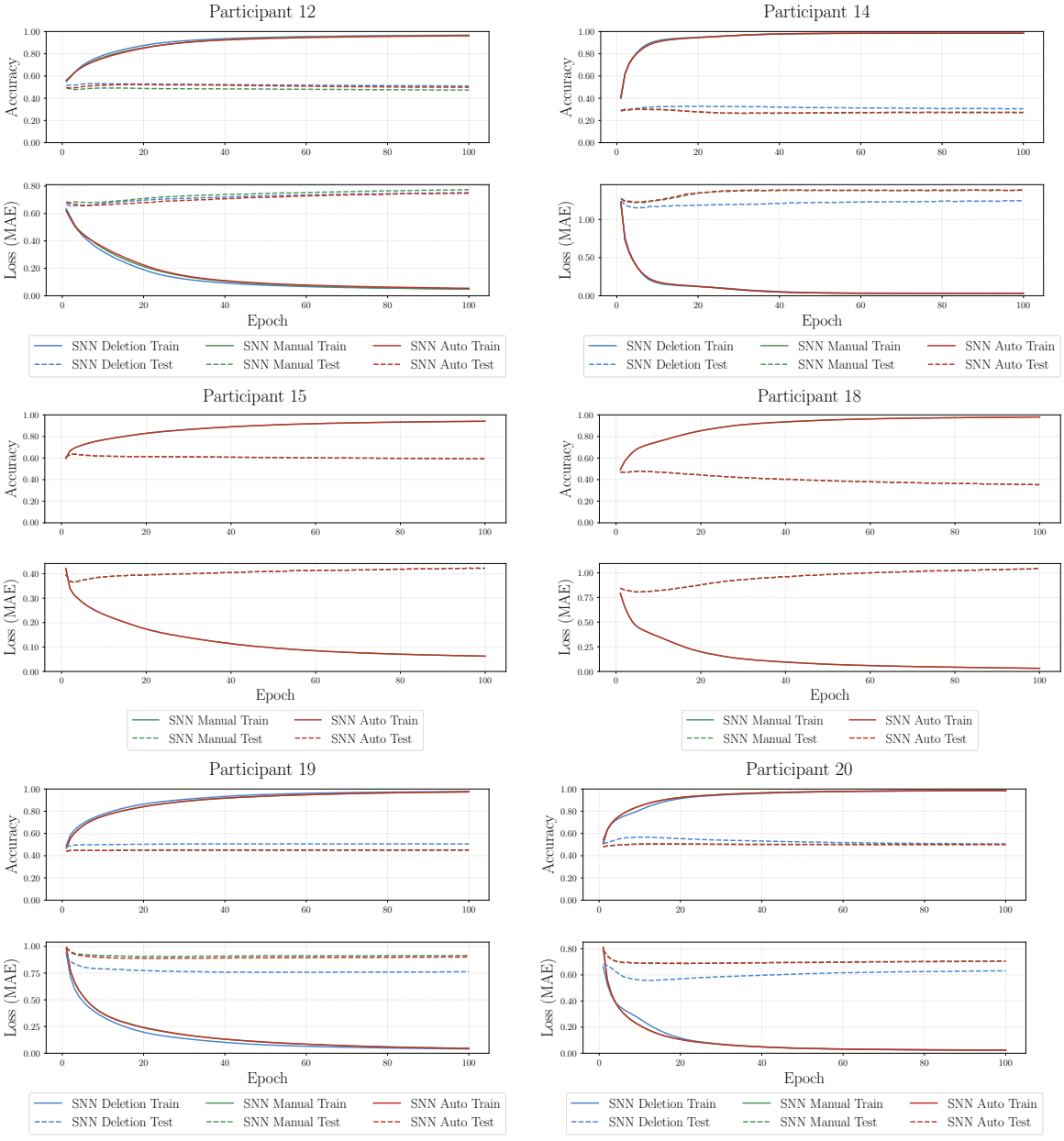

Participant 12
Participant 14
Participant 15
Participant 18
Participant 19
Participant 20
Accuracy
Loss (MAE)
Epoch
SNN Deletion Train
SNN Deletion Test
SNN Manual Train
SNN Manual Test
SNN Auto Train
SNN Auto Test

Fig. B.1 *Cont.*

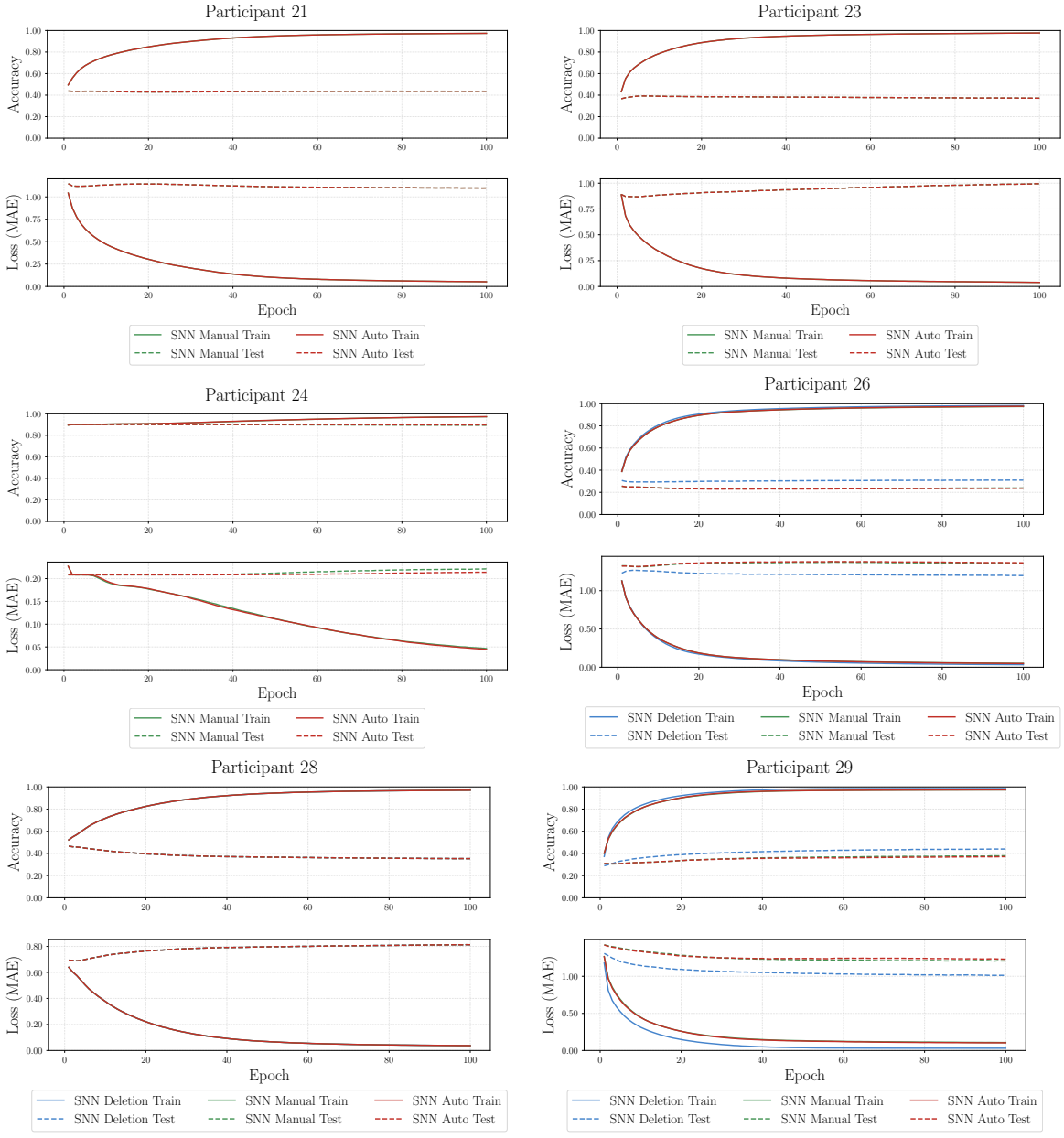
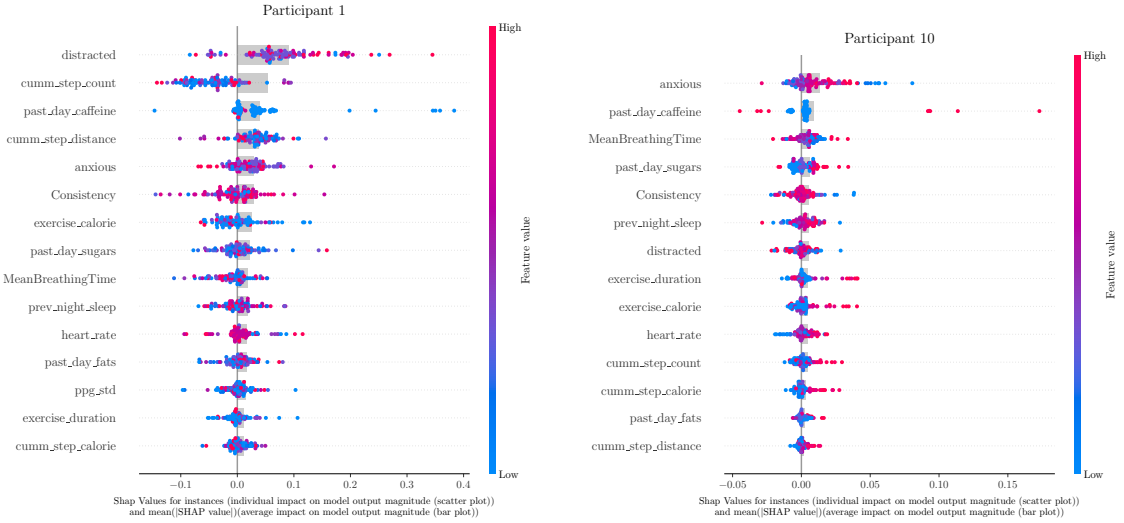

Participant 21
Participant 23
Participant 24
Participant 26
Participant 28
Participant 29
Accuracy
Loss (MAE)
Epoch
SNN Deletion Train
SNN Deletion Test
SNN Manual Train
SNN Manual Test
SNN Auto Train
SNN Auto Test

Fig. B.1 *Cont.*

Appendix C

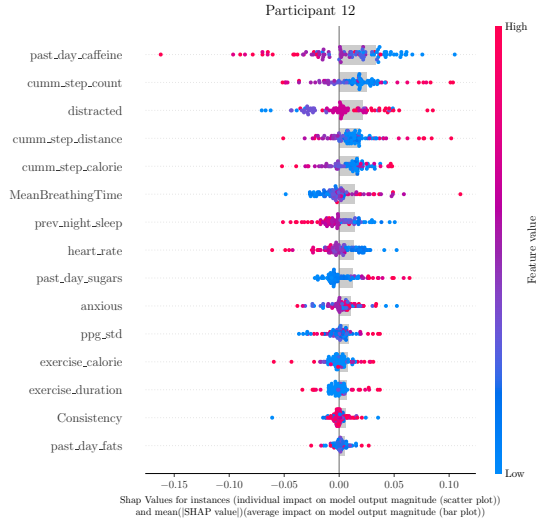
Shapley Additive ExPlanation Feature Contribution Plots Using Reduced Feature Set: All Participants



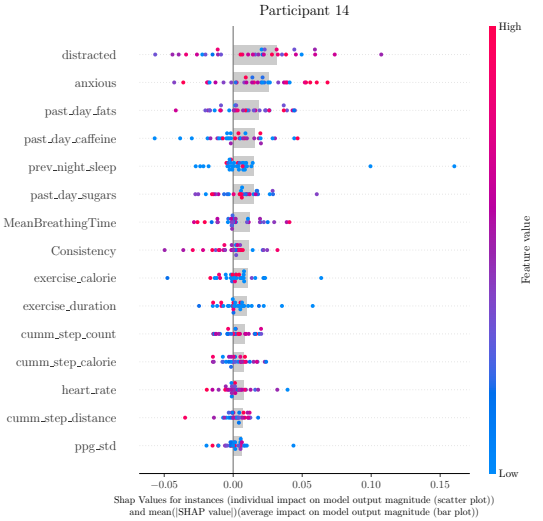
(a) Participant 1 (Automatic)

(b) Participant 10 (Manual)

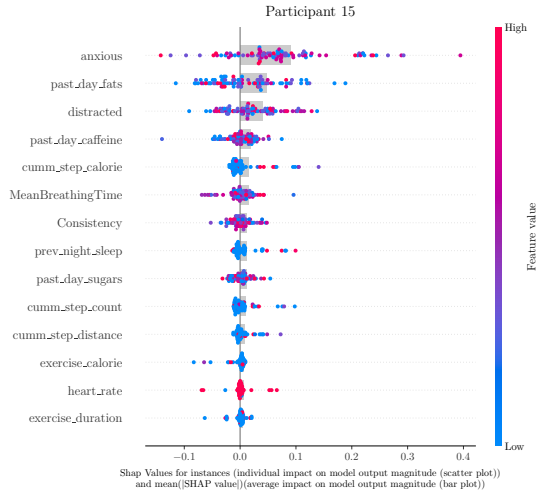
Fig. C.1 Shapley Additive ExPlanation (SHAP) feature contribution plots using the reduced feature groups. Note that depending on the number of features reduced for constant values, the number of features in the Shapley Additive ExPlanation (SHAP) plots may differ as well.



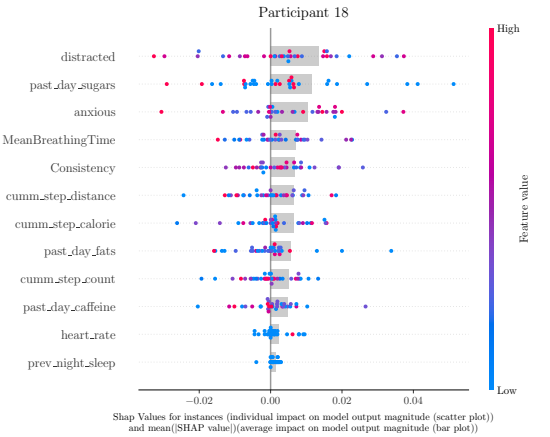
(c) Participant 12 (Deletion)



(d) Participant 14 (Deletion)

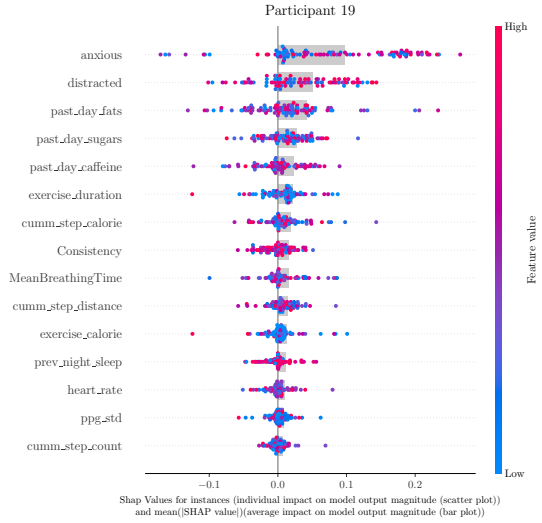


(e) Participant 15 (Manual)

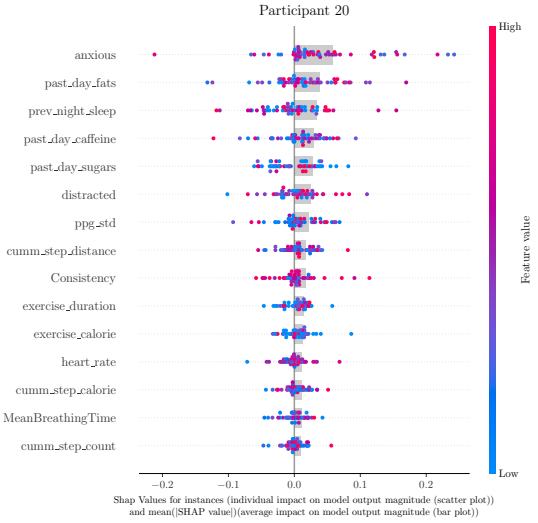


(f) Participant 18 (Manual)

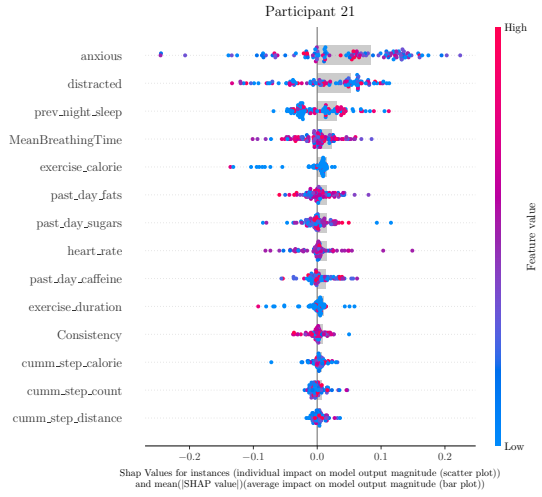
Fig. C.1 *Cont.*



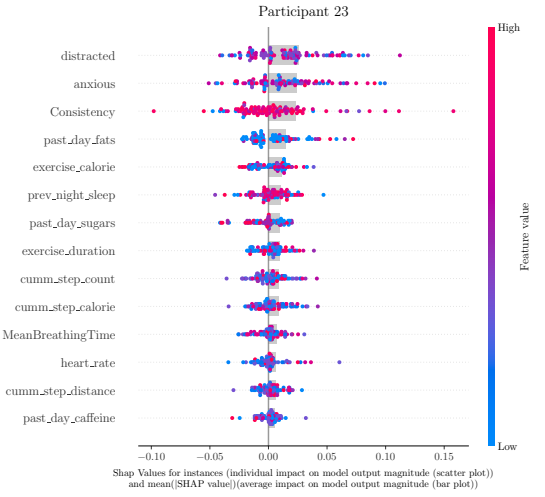
(g) Participant 19 (Deletion)



(h) Participant 20 (Deletion)

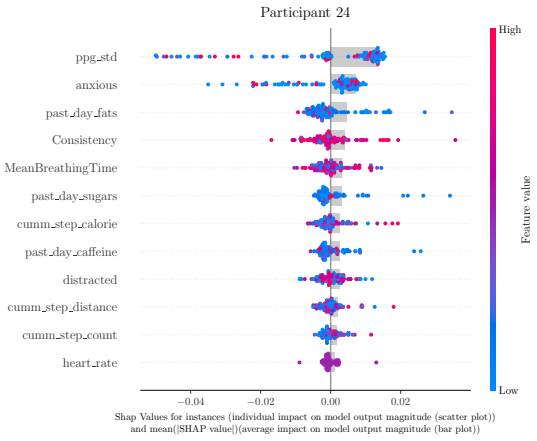


(i) Participant 21 (Manual)

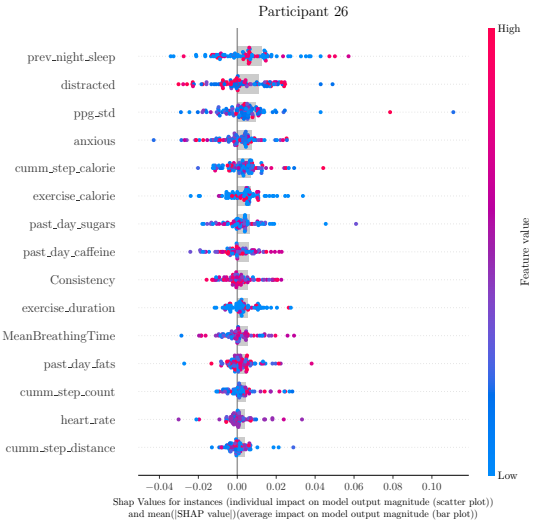


(j) Participant 23 (Manual)

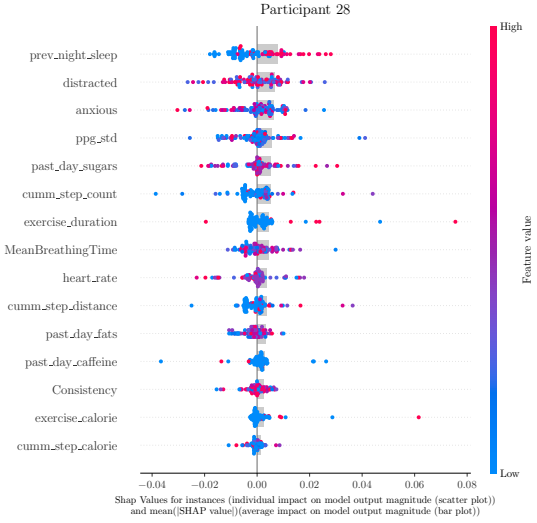
Fig. C.1 *Cont.*



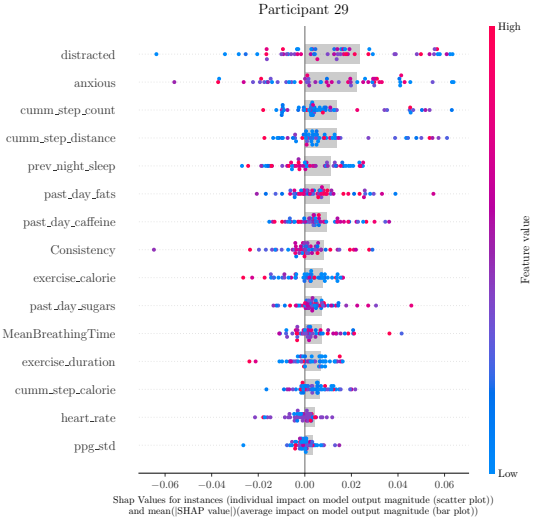
(k) Participant 24 (Manual)



(l) Participant 26 (Deletion)



(m) Participant 28 (Deletion)



(n) Participant 29 (Deletion)

Fig. C.1 *Cont.*