IMPERIAL

IMPERIAL COLLEGE LONDON

DEPARTMENT OF BIOENGINEERING

Final report for MSc Human and Biological Robotics Project

A Characterisation of Convolutional Spiking Autoencoders

Author:
Luke Alderson

Supervisor(s):
Prof. Claudia Clopath
Albert Albesa González

Submitted in partial fulfillment of the requirements for the award of MSc in Human and Biological Robotics from Imperial College London

March 2024

Acknowledgments

I'd like to take the opportunity to thank several people for the material, intellectual, and emotional support provided to me as I worked on my thesis, as well as for the entire past year of study.

Firstly, I am extremely grateful to Professor Claudia Clopath for supervising this project. I would also like to offer additional thanks for delivering related modules in such an engaging way. Her approach is the reason I chose this project, and it didn't disappoint. I would further like to extend a sincere thank you to Albert Albesa González for the quality and quantity of advice and discussion around my topic. Coming from a background in electronic engineering, our near-weekly meetings were invaluable to my development in a new area. I wouldn't have made anywhere near as much progress without his patient approach to teaching new concepts.

I'd like to also send my love and my thanks to my partner, Megan ("Mog"), for enabling me to embark on this course in the first place and for being so supportive throughout its duration. It's a cliché, but I truly wouldn't have been able to do it without her (or our cat, Loaf – thanks for keeping my desk warm!). To my Dad and Stepmum, Nik, thank you so much for the constant words of encouragement, your interest, and for keeping me sane and happy with regular bonfire nights, takeaways, and pub trips. Thanks also to my Mum and Stepdad, David, for your words of support throughout this last year.

Finally, I'd like to mention my Granny, Catherine, who ignited my interest in science and engineering from an early age and continued fanning the flames throughout my life. I owe so much to her for her unwavering support and the boundless energy with which she offered it. I miss you, Gran.

Abstract

Spiking neural networks (SNNs) propagate information using discrete spikes over time, emulating action potentials in the brain. This sparsity enhances memory and energy efficiency compared to traditional artificial neural networks (ANNs). This study characterizes the behaviour of a spiking autoencoder (SAE) applied to the Modified National Institute of Standards and Technology (MNIST) and Dynamic Vision Sensor (DVS) Gesture datasets. SAEs, as self-supervised networks, encode input data into a lower-dimensional latent space and then reconstruct it at the output. We first established a stable baseline model through a grid search of hyperparameters. Next, we evaluated the impact of rate coding on test loss, followed by an assessment of implicit and explicit recurrent connections and their influence on performance. Additionally, we examined the effect of varying embedding sizes. The ratio of maximum to minimum firing frequency in rate-coded spike trains, acting as contrast control, was inversely related to reconstruction loss. However, recurrent connections offered minimal improvement in this SAE implementation, contrary to other literature. Although increasing embedding size generally reduced test loss, certain values led to significant performance drops. This exploratory study's findings align with prior research on kernel size dependency and suggest that future work should explore parameter initialization techniques and skip connections to mitigate unstable embeddings and enhance network learning dynamics.

Contents

List of Figures

List of Tables

1 Introduction

Humanity has long looked to nature for direction when developing technology across a range of domains. The design of artificial neural networks (ANNs) was inspired by synaptic connections in the mammalian brain, while convolutional neural networks (CNNs) mimic the receptive fields of human retinal ganglion cells. These networks are widely used to address classification (Specht, 1991; Acharya et al., 2017) and regression (Dreiseitl and Ohno-Machado, 2002) problems. Spiking Neural Networks (SNNs) continue this biomimetic approach, representing a third generation of neural network models, following perceptrons and multi-layer networks with activation functions (Maass, 1997). This work aims to characterise the behaviour of convolutional, spiking autoencoder (SAE) models when trained on visual datasets. It is hoped that this analysis could provide deeper insights into the way the brain processes information in low-level visual cortical regions.

Autoencoders (Figure 1) are self-supervised neural networks designed to decompose an input signal into a lower-dimensional latent space (encoding) and then reconstruct the original data from this representation (decoding). This is achieved by minimising a reconstruction error, which quantifies the difference between the input and it's approximation at the decoder output, and optimising the network parameters through backpropagation. Autoencoders have a number of applications including data compression (Cheng et al., 2018; Liu et al., 2023), missing value imputation (Beaulieu-Jones and Moore, 2017), denoising (Chiang et al., 2019; Ryu et al., 2020), and feature extraction (Toğaçar et al., 2021; Eom et al., 2021). Mathematically, the encoder is a parameterised vector function that maps inputs onto a lower dimensional latent space. The decoder then attempts to reconstruct the input data from the latent space using learned parameters. The latent space must be smaller than the input space to avoid the model learning a trivial identity function of the input.

Typically, autoencoder parameters are learned by backpropagating the loss through the network using gradient descent. A common loss function is the L2 norm of the reconstruction error, which measures the Euclidean distance between the input data and the reconstruction. The vanilla autoencoder has been extended in various ways. For example, Rifai et al. (2011) propose a contractive autoencoder that includes a regularisation term to enhance robustness to noise, while variational autoencoders (Kingma and Welling, 2019) introduce a generative component by parameterising a probability distribution in the latent space and sampling from it to generate new data.

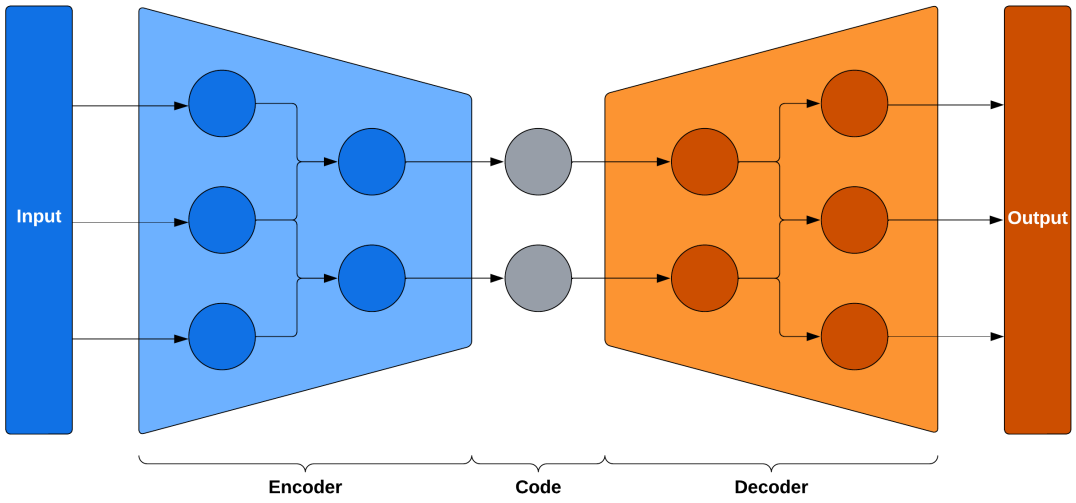


Figure 1: Autoencoders consist of three sections. The input is fed into the encoder, which passes and compresses the data to a lower dimensional space, called the code. The output of the code layer is passed to the decoder section, which learns to reconstruct the original data from the low dimensional representation at the output, by minimising a reconstruction error.

This work presents an analysis of SAE; a subset of SNNs. SNNs aim to replicate information propagation between neurons in the brain through action potentials, or 'spikes'. Unlike ANNs that

use floating-point values, SNNs represent data as time-series binary spikes. These artificial spikes approximate biological data transmission with varying fidelity, depending on the neuronal model and learning rules. The use of binary spiking representations offers energy-efficient computation similar to that in the brain, especially when implemented on neuromorphic hardware. Additionally, spikes enhance sparsity, significantly reducing memory complexity (Eshraghian et al., 2023).

The brain demonstrates remarkable computational capability while operating on a power budget of 12 - 20 W. In contrast, modern machine learning models require substantial energy for both training and operation (Brown et al., 2020; Dhar, 2020). For example, OpenAI's GPT-3 required approximately 190 MWh to train (Anthony et al., 2020), a figure far exceeding the brain's energy usage over several human lifetimes. Although SNNs have not yet matched the performance of traditional ANNs, they have proven effective in tasks such as temporal signal classification (Yan et al., 2021; Wu et al., 2018), image classification (Iakymchuk et al., 2015), regression (Dreiseitl and Ohno-Machado, 2002; Specht, 1991; Gehrig et al., 2020), dimensionality reduction (Chaturvedi and Khurshid, 2011), and statistical filtering (Juárez-Lora et al., 2022; Bobrowski et al., 2009). It is theorized that advancements in SNNs could eventually surpass the computational capabilities of ANNs, while maintaining high energy efficiency and low memory complexity (Maass, 1997).

Aligning machine learning algorithms with neural structures and mechanisms could lead to more efficient implementations of deep learning on neuromorphic hardware and provide insights into how the brain processes information. The foundational unit of an ANN is the neuron, while the foundational unit of an SNN is the spiking neuron, which more closely mimics the dynamics of biological neurons. Spiking neurons aim to replicate the dynamics of a neuron's membrane potential in response to an input current, incorporating spiking and resetting mechanics. Membrane potential increases when fed with an input current, approximating excitatory and inhibitory post-synaptic potentials. When the membrane potential exceeds a threshold through summation of these graded potentials, a spike is triggered and propagated to neurons in the next layer. This threshold can be constant or a function of activity in more complex architectures (Wu et al., 2023; Ding et al., 2022). Typically, the spike is modelled as an instantaneous impulse, similar to the Dirac function, as the timing of an action potential is more important for information transmission than its precise shape. After firing, the membrane potential is either reduced by the threshold (soft reset) or reset to zero (hard reset).

A variety of spiking neuron models exist, each with different levels of biological plausibility and computational complexity. Eshraghian et al. (2023) provides a review of some of the most common models. Hodgkin and Huxley (1952) presented the first detailed model of neuron dynamics with extensive biological detail, building upon Lapicque's earlier work (Brunel and van Rossum, 2007). Represented by a four-dimensional system of first-order ordinary differential equations (ODEs) with conductance terms for specific ion channels, this model effectively captures complex neuronal behaviour but is computationally intensive for large-scale applications. The Izhikevich model (Izhikevich, 2003) offers a high-fidelity replication of biological processes with less complexity than the Hodgkin-Huxley model, using two first-order ODEs. It has been used in SNNs deployed on FPGAs (Rice et al., 2009; Alkabaa et al., 2022) for classification tasks. The Leaky Integrate and Fire (LIF) model is widely used due to its simple yet effective representation of membrane potential dynamics. It employs a first-order linear ODE for membrane potential, with additional conditions for spiking and resetting (Stein, 1967; Tal and Schwartz, 1997; Wang et al., 2014; Brunel and van Rossum, 2007). The current-based leaky-integrate-and-fire (CUBA-LIF) model is a variant where the leaky integrator dynamics are applied to the current before affecting the membrane potential, effectively acting as a second-order filter of the membrane potential (Eshraghian et al., 2023; Lava, n.d.). Dampfhoffer et al. (2022) demonstrated that the CUBA-LIF model can outperform the LIF model in processing temporal sequences, such as audio datasets.

Learning rules are a critical component to biological and machine learning. In ANNs, the use of backpropagation to adjust network parameters based on the minimisation of a cost function is near ubiquitous. Following a forward pass through the network, the loss is calculated, representing the difference between the network's output and the desired output. In the case of supervised neural networks, this loss is often the mean squared error (MSE) or cross-entropy (CE) between the output of the network and the supplied target variables. In self-supervised networks like autoencoders, the loss is formulated as a reconstruction error, using a metric to measure the difference between the input and its latent space reconstruction. Backpropagation then updates network parameters to minimize

this loss, with gradient descent and the chain rule guiding the parameter adjustments towards a local minimum across multiple training iterations.

Backpropagation faces challenges when applied directly to SNNs because spikes are modelled as instances of the Dirac-delta function, making them non-differentiable. The general form of the chain rule for a SNN is given by:

$$\frac{\partial L}{\partial W_{ij}} = \frac{\partial L}{\partial S_{out}} \frac{\partial S_{out}}{\partial U} \frac{\partial U}{\partial W_{ij}} \tag{1}$$

where L is the loss function, such as the Euclidean norm, W_{ij} represents the weight between the ith neuron of a given layer and the jth neuron in thee subsequent layer, and S_{out} is the spike released from the neuron as a function of the membrane potential, U.

The weight update is then computed as:

$$W_{ij}^{t+1} = W_{ij}^{t} - \alpha \frac{\partial L}{\partial W_{ij}^{t}} \tag{2}$$

where W_{ij}^{t+1} is the updated weight, W_{ij}^{t} is the current weight, and α is the learning rate.

Direct application of the chain rule in spiking neuron layers is problematic because it ultimately prevents useful weight updates as described below:

$$\frac{\partial S_{out}}{\partial U} \in \{0, \infty\} \implies \frac{\partial L}{\partial W_{ij}^{t}} \in \{0, \infty\} \implies W_{ij}^{t+1} \in \{W_{ij,-\infty}^{t}\} \tag{3}$$

Another issue unique to SNN is the ”dead neuron” problem, not to be confused for the vanishing gradient problems of recurrent neural networks. A dead neuron in an SNN is one that fails to fire in response to stimuli because its membrane potential never reaches the spike threshold. In this state, the neuron does not contribute to the network loss and therefore its weights are never updated during backpropagation, effectively disabling the neuron. Fortunately, various techniques have been proposed to address this issue. These techniques, like neuron models, vary in their biological plausibility and computational effectiveness. The simplest approach is 'shadow learning', in which an ANN is trained and then converted to a SNN for testing and deployment. This conversion involves translating the activation function of each non-spiking neuron into either a rate or latency code (Eshraghian et al., 2023; Han and Roy, 2020). While this method allows SNNs to leverage advancements in ANN techniques, it has the drawback that the shadow-trained SNN remains an approximation of the ANN and is unlikely to outperform the original network.

A popular method found in the literature is the use of surrogate gradients, enabling standard backpropagation methods to be applied despite the spiking mechanism. The forward pass through the network is conducted using the ordinary spiking neuron, which given the thresholding is analogous to the non-differentiable Heaviside activation function. The backward pass is then conducted using a differentiable surrogate gradient, most commonly a sigmoid or arctan (Neftci et al., 2019; Malcolm and Casco-Rodriguez, 2023) activation function, which approximates a smoothed version of the Heaviside function (Figure 2).

The aforementioned methods have demonstrated promising results from a computational perspective, but lack grounding in observed biological processes. Biologically plausible local learning rules, such as spike-timing-dependant-plasticity (STDP), have been described in several studies (Hao et al., 2020; Iakymchuk et al., 2015; Lee et al., 2018; Kheradpisheh et al., 2018; Gerstner et al., 1996), offering closer alignment with neural mechanisms while aiming to preserve performance. STDP is a learning rule whereby the weighting of synapses is determined by the firing order of pre- and post-synaptic action potentials within a specific time window (Song et al., 2000; Bi and Poo, 1998; Gerstner et al., 1996) (Figure 3). This phenomenon has been observed across a range of species, including humans, rats and insects (Bi and Poo, 1998; Caporale and Dan, 2008). STDP has been incorporated into various SNN models to either supplement or replace traditional backpropagation methods inherited from classical machine learning.

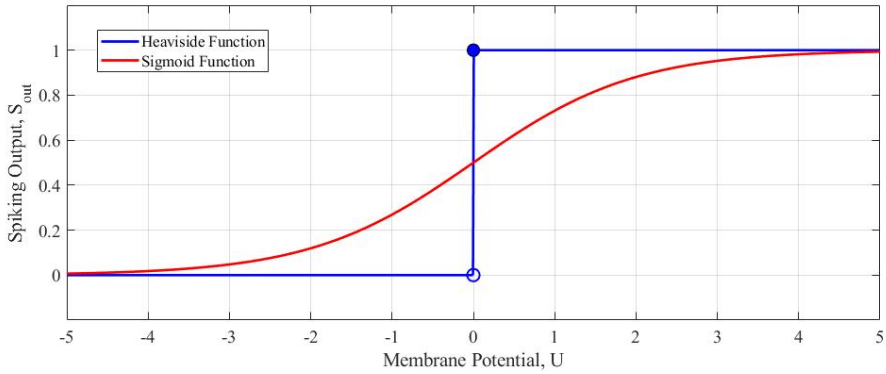


Figure 2: When using surrogate gradients, the discontinuous Heaviside function is used on the network forward pass, but replaced with a differentiable function, such as a sigmoid, on the backward pass during parameter updates.

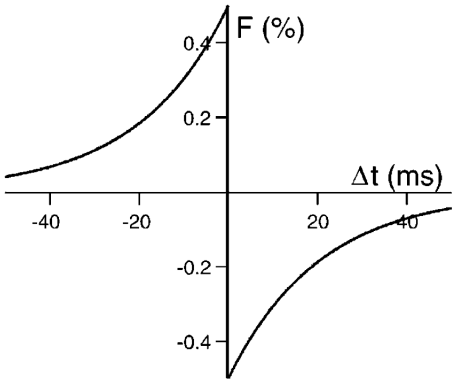


Figure 3: The STDP modification function shows how the peak conductance at a synapse changes depending upon the time difference between pre- and post-synaptic spikes (Song et al., 2000)

Hao et al. (2020) utilized local STDP learning rules and dynamic thresholding within a LIF SNN, trained on MNIST and Fashion-MNIST (F-MNIST) datasets. Their model achieved accuracies of 97% and 85%, respectively. The MNIST performance is comparable to modern ANNs (Byerly et al., 2020; Ravi, 2017), while the F-MNIST results are only slightly behind those of contemporary ANNs (Sinha et al., 2019; Greeshma and Sreekumar, 2019; Nocentini et al., 2022). In a separate study, Iakymchuk et al. (2015) introduced a simplified spike response model neuron with STDP learning rules, demonstrating rapid classification on embedded platforms. Their model classified numerals within the SEMEION dataset in 5-6% of the time required by traditional network architectures, underscoring the advantages of SNNs in optimizing time and memory complexity.

Developing effective SNN architectures requires careful consideration of how data is converted into spiking form, reflecting neural coding mechanisms in the brain. Neural coding encompasses various strategies by which the brain encodes and decodes information as spikes; a topic of considerable debate (Gerstner et al., 1997; Prescott and Sejnowski, 2008; Iakymchuk et al., 2015; Johnson et al., 2016; Comşa et al., 2021). Traditionally, rate coding and latency coding have been regarded as the two main methods. Rate coding maps data to a spike frequency, which can be taken across time, trials, or neuron populations. For instance, higher intensity data may produce a higher frequency response, while lower intensity data results in a lower frequency. Latency coding, on the other hand, maps data to the delay between the stimulus and the first neuron in a population to fire. In this scheme, data with higher values or intensities corresponds to quicker firing times. It is challenging to reconcile either method exclusively, leading recent research to explore new or combined encoding mechanisms. It is proposed that the brain likely uses a combination of these methods, either antagonistically or synergistically, depending on the context (Prescott and Sejnowski, 2008; Ainsworth

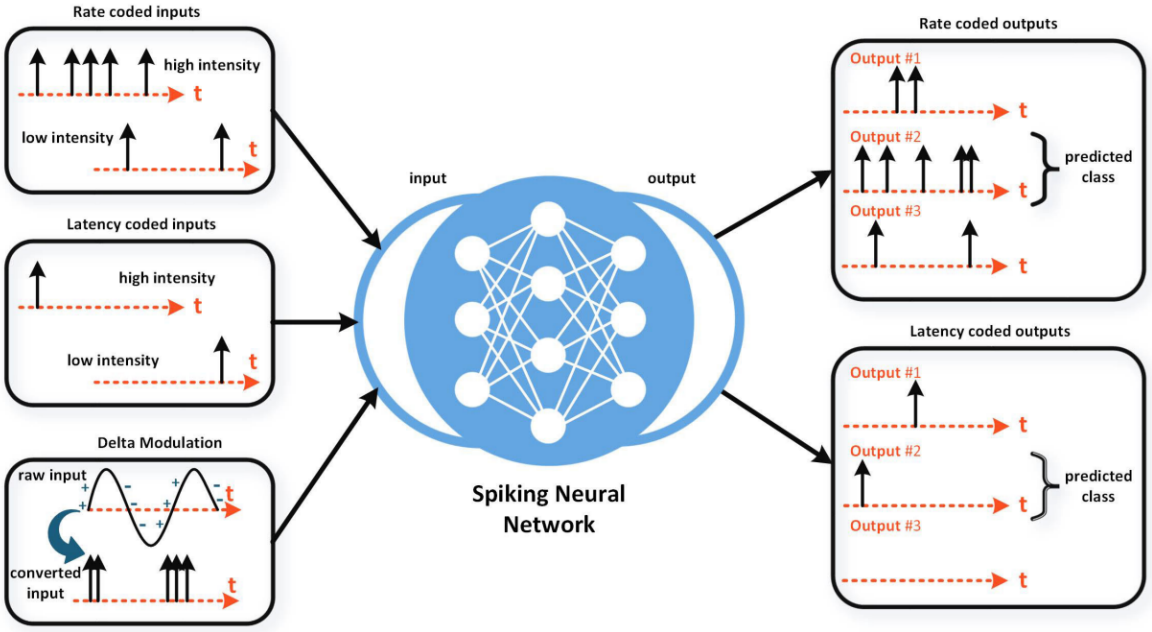


Figure 4: Eshraghian et al. (2023) present a summary of neural encoding and coding schemes, considering population coding as a form of rate coding.

et al., 2012; Wang et al., 2008).

Eshraghian et al. (2023) provide an overview of coding and decoding mechanisms (Figure 4) and evaluate their efficacy in terms of speed, efficiency and representation ability from an engineering perspective. Gerstner et al. (1996) modelled the auditory system of a barn owl and demonstrated that temporal coding, coupled with an unsupervised Hebbian learning rule, could process temporal information with greater resolution than expected based on membrane time constants. Iakymchuk et al. (2015) developed a neuronal spike response model (SRM) with STDP learning rules for embedded applications, generating effective representations using rate decoding. Han and Roy (2020) introduced a temporal coding strategy whereby each image pixel is encoded as a pair of spikes, where the inter-spike interval represents pixel intensity. This method increased sparsity and improved energy efficiency in a shadow-trained SNN, achieving accuracies of 93.63%, 70.97% and 73.46% on CIFAR-10, CIFAR-100, and ImageNet datasets, respectively. Schultz et al. (1997) identified dopaminergic neuron spikes representing error rates between expected and actual rewards, suggesting rate coding linked to error signals. They modelled this behaviour using temporal difference (TD) methods from reinforcement learning, and propose that spike rates are consistent with scalar prediction error signals. Marblestone et al. (2016) similarly suggest that the brain uses mechanisms analogous to ANN cost functions, locally optimised for specific brain regions and unfolding through time during development.

Spiking neurons have been integrated into popular neural network architectures, including convolutional (Iakymchuk et al., 2015; Kheradpisheh et al., 2018; Cao et al., 2015; Lee et al., 2018; Tavanaei and Maida, 2017), and recurrent networks (Yin et al., 2021; Zhang and Li, 2019; Shen et al., 2021). Integrating spiking neurons into autoencoder architectures to create SAEs is an expanding area of research, though it remains relatively underrepresented. There is increasing interest in investigating the representations that SAEs develop compared to non-spiking autoencoders, particularly in image generation and reconstruction. Kamata et al. (2022) used a spiking variational autoencoder for image generation, achieving quality equal to or greater than ANNs, with latent spike trains modelled as a Bernoulli process. Burbank (2015) implemented a SAE using a modified STDP learning rule. They were able to learn distributed representations of MNIST data analogous to those learned in parts of mammalian visual cortical regions. Comşa et al. (2021) built a SAE with a single hidden layer to analyse compact MNIST representations using latency coding, exploring the effect of spike timing on the latent structures. They reconstructed MNIST digits with a fidelity on par with

conventional autoencoders.

The development and analysis of SAEs is a young but expanding field of research. The representations formed in the code layer of autoencoders could be considered analogous to how information is stored and processed in lower-level cortical regions. Further research could provide insights into neural coding mechanisms and learning rules in these areas. It remains unclear how latent representations of input data are encoded within the latent space of SAE, or whether these networks favour rate, latency, or more complex coding schemes. This study proposes an analysis of the latent representation found during the training of a SAE. This work presents a comparison between the latent representations and reconstructions learned from both spiking and vanilla autoencoders. Evaluations of latent representations using non-spiking neurons in autoencoder architectures has been performed in depth (Kingma and Welling, 2019; Beaulieu-Jones and Moore, 2017; Liu et al., 2023; Cheng et al., 2018). However, to the best of the author's knowledge, a detailed analysis has not yet been conducted in the context of a spiking network. This work aims to elucidate the representations that emerge during training of a convolutional SAE of LIF neurons on a selection of visual data, using surrogate gradients.

1.1 Aims

The aims of this work are to characterise the behaviour of SAE under various encoding, recurrence, and embedding configurations.

1.2 Objectives

The objectives of this study are as follows;

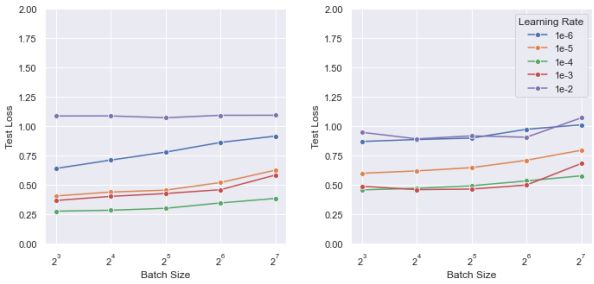
1. Identify and access datasets for training and testing the autoencoders.
2. Develop the foundations of non-spiking and spiking networks, with similar high level architecture.
3. Conduct a hyperparameter search on both models to identify baseline performance metrics.
4. Evaluate different rate coding schemes on the reconstruction loss of the SAE.
5. Evaluate the impact of recurrence in the latent layer when training on temporally stationary (MNIST) and non-stationary data (DVS).
6. Evaluate the impact of different embedding sizes on autoencoder performance.

2 Results

2.1 Spiking and Non-Spiking Networks of Equivalent Architecture Share Optimal Hyperparameters

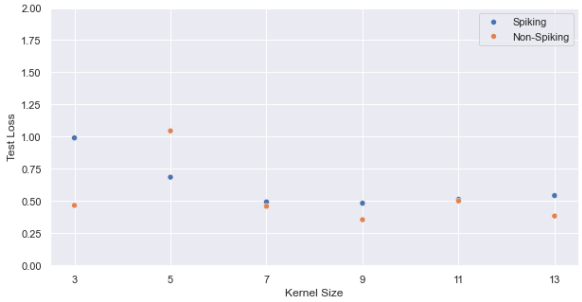
To identify a model with optimal baseline performance for future experiments, a grid search was conducted over learning rate, batch size, and kernel size hyperparameters (Section 4.6.1). Analysis revealed that a learning rate of $1\mathrm{e}{-4}$ achieved the best performance across a wide range of batch sizes for both models (Figure 5a and 5b). The batch size had a relatively minor impact on the loss at the selected learning rates, therefore a batch size of 64 (2^6) was selected as a practical balance between performance, training duration, and memory usage.

A sweep over kernel sizes for a fixed learning rate and batch size (Figure 5c) was performed. A kernel size of 7×7 was selected as the baseline for both models, as it achieved one of the lowest loss values without excessive memory usage. The spiking and non-spiking networks were able to provide good fidelity reconstructions with these parameters (Figure 5d).



(a) AE

(b) SAE



(c) Kernel size sweep ($lr = 1\mathrm{e}{-4}$, $bs = 64$).

(d) i) Original MNIST, ii) Reconstructed original MNIST, iii) Rate encoded MNIST, iv) Reconstructed rate encoded MNIST

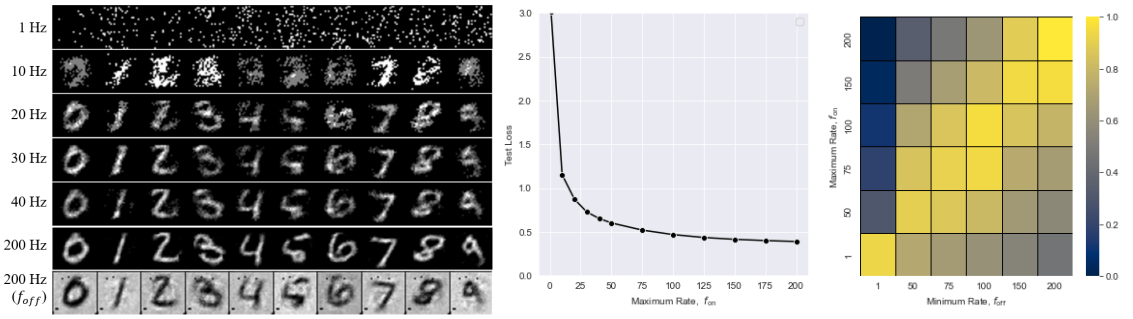
Figure 5: The test accuracy of each model was evaluated through a hyperparameter grid search. (a, b) The optimal learning rate was determined to be $lr = 1\mathrm{e}{-4}$ for most batch sizes in both models. A batch size of $bs = 64$ was chosen for its balance between performance, training speed, and memory usage. (c) The minimum test loss for the spiking network was achieved for a kernel size of 7. For the non-spiking network, the minimum test loss was recorded as 9; however, 7 was selected as the baseline since the difference in loss was negligible, and it allowed for consistent baseline parameters across both networks. All kernels used are square. (d) Inputs and reconstructions of the spiking and non-spiking autoencoders using the selected optimal values.

2.2 Increasing the Ratio of Maximum and Minimum Encoding Frequency Reduces Autoencoder Reconstruction Loss and Induces a Semantically-Biased Clustering in the Latent Space

To evaluate the effect of varying encoding frequency on model performance, the model was trained on MNIST data subject to various rate code frequencies (Section 4.6.2). The minimum firing rate (f_{off}) was fixed, while the maximum firing rate (f_{on}) was varied. The reconstruction fidelity was observed to improve as the difference between the two parameters increased (Figure 6a). This corresponds with an asymptotically decreasing test loss as f_{on} increased (Figure 6b).

When f_{off} was allowed to vary, the lowest test losses occurred for higher $f_{on} : f_{off}$ ratios (Figure 6c). When $f_{on} \approx f_{off}$, the test loss was maximized due to the lack of differentiation between pixel activations. Reconstruction was also possible with a reversed ratio, resulting in negative reconstructions (Figure 6a), though the test loss was generally higher compared to positive ratios.

Uniform Manifold Approximation and Projection (UMAP) (McInnes et al., 2018) was used to project the output of the code layer onto 2-dimensions for display (Figure 7). The colour of each data point corresponds to an arbitrary label from the MNIST dataset; note how the clusters correspond to the semantic labels of the input data even though they were not explicitly provided. The degree of clustering formed in the latent space appears to be closely linked to the level of rate coding applied to the input. Homogenous structures emerge where $f_{on} \approx f_{off}$, suggesting the network struggles to learn effective embeddings of the input. Greater disparity between encoding thresholds causes distinct clusters to emerge in the latent space, forming the basis for effective decoding. There also appears to be a correlation between the degree of clustering and the recorded test loss (Figure 6c), with better test losses stemming from embeddings with greater separation. The exact degree of causality between test loss and embedding separation is unclear, as further tuning of the decoder parameters or structure may enable the reconstruction of even poor quality embeddings.



(a) Reconstructions for various f_{on} with $f_{off} = 1$ Hz, and a reconstruction with $f_{off} = 200$ Hz and $f_{on} = 1$ Hz

(b) Loss against maximum rate

(c) Loss heatmap

Figure 6: Model efficacy was evaluated under a variety of rate coding schemes. (a) Reconstructions are shown for various input rate codes. (b) Holding the minimum firing rate constant while increasing the maximum firing rate decreases the test loss of the reconstruction. (c) The heatmap shows that the model achieves the lowest loss when there is a high differentiation between minimum and maximum firing rates. Loss values comprising the heatmap were range normalised between 0 and 1, hence slight variation in loss along the diagonal. This does not reflect a significant difference in loss between these diagonal elements, as the unadjusted nRMSE was above 1 for all, confirming no learning occurred at these values.

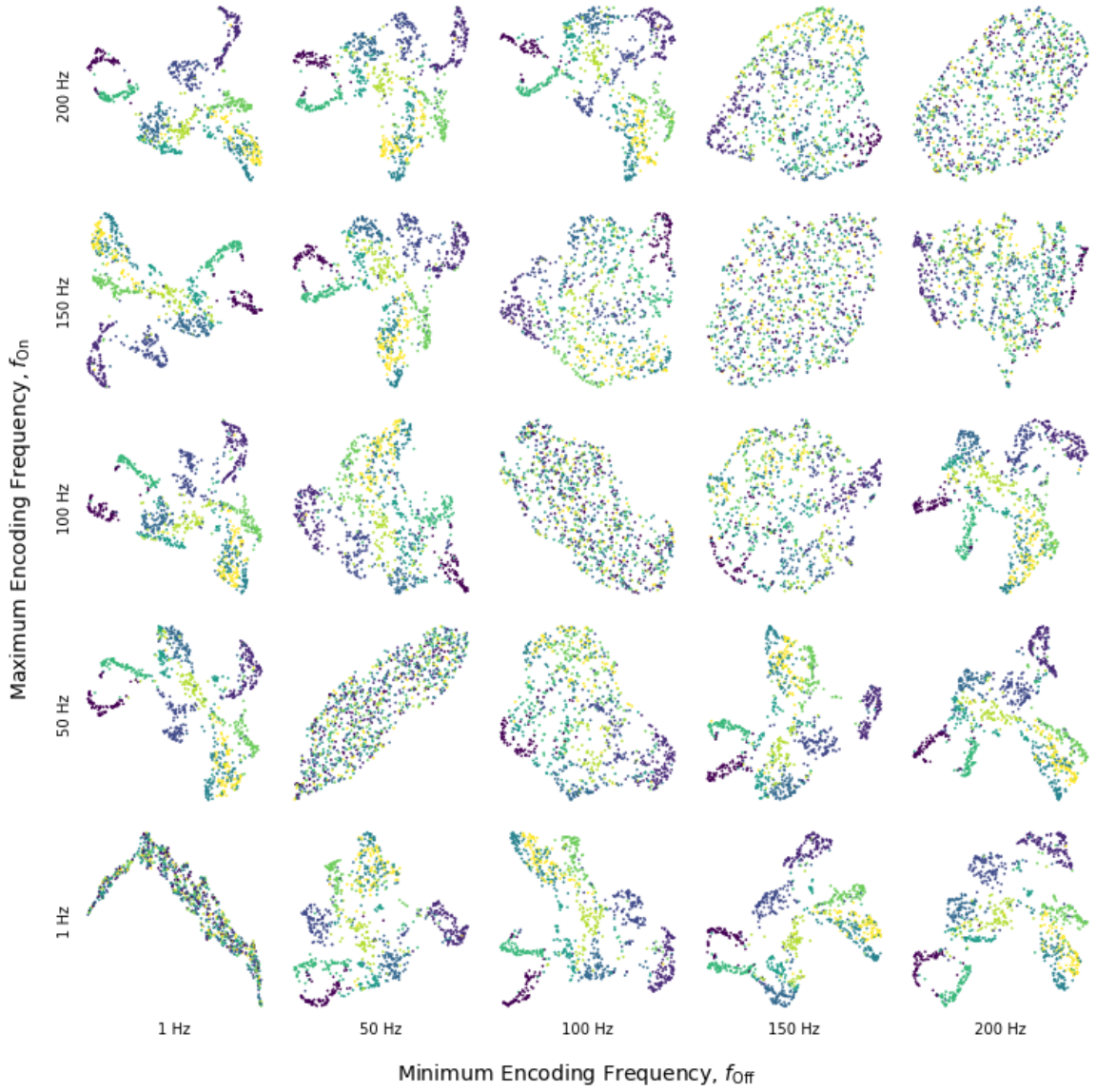


Figure 7: The autoencoder's reconstruction ability is influenced by the quality of the embeddings learnt by the encoder. The 25 subplots display 2D UMAP projections of the code layer of the autoencoder for various encoding frequency thresholds. The colour coding of the data points corresponds to different MNIST digits. The most effective representations are observed where there is a significant disparity between f_{on} and f_{off}, as shown in the upper left and lower right quadrants. When f_{on} and f_{off} are similar, the autoencoder struggles to learn distinct patterns, resulting in a homogeneous latent space.

2.3 Explicit Recurrence in Hidden Layer has No Significant Effect on Performance

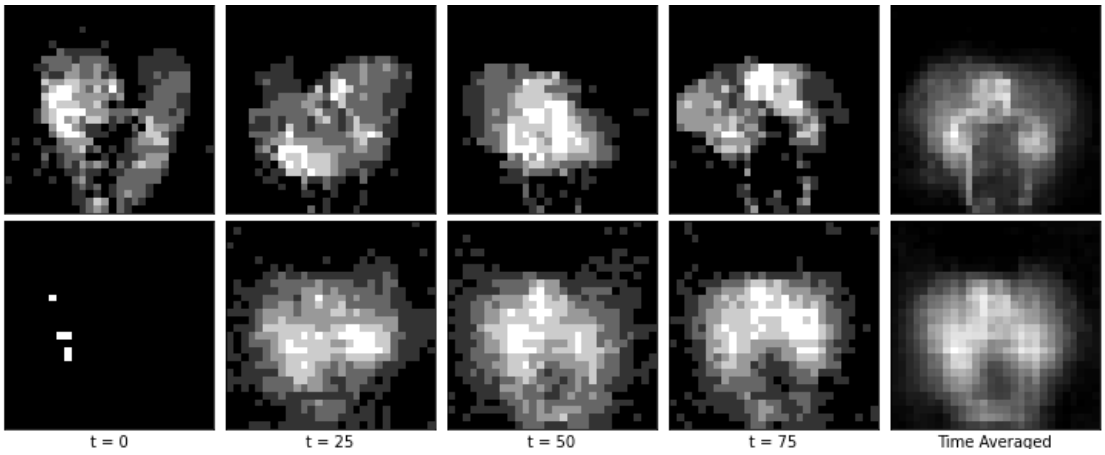
Spiking networks can exhibit implicit and explicit recurrence (Neftci et al., 2019; Bouanane et al., 2023). Implicit recurrence results from the dynamics of the LIF neuron, since the membrane potential at the present time-step, $U[t]$, depends on its value at the previous time-step, $U[t-1]$, scaled by the decay rate constant β as shown in (10). Explicit recurrence can be introduced by feeding the output spikes of a LIF neuron or LIF layer directly back to their input. The effect of incorporating different recurrent dynamics into the spiking network is analysed in this section for a variety of training data and network architectures.

Three different recurrence modes were applied to different configurations of spiking network

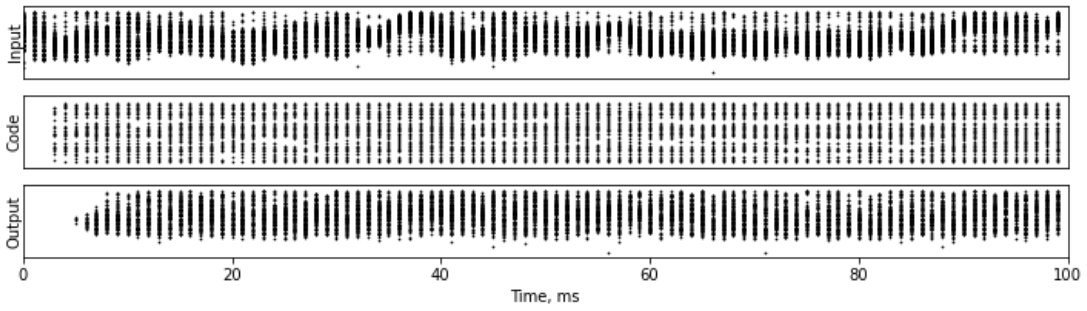
(Section 4.6.3); explicit recurrence (RLeaky), and implicit recurrence using one (1-layer) and two layers (2-layer) in the latent space. The impact of recurrent connections was evaluated for; a spiking autoencoder trained on MNIST with varying kernel size, a spiking autoencoder trained on DVS with varying kernel size (Figure 8), a spiking autoencoder trained on DVS with varying training targets, and a spiking classifier trained on DVS with varying kernel size.

The data was observed to be independent and non-normal. Consequently, the Kruskal-Wallis Test was selected to evaluate groupwise significance between the three recurrence treatments within groups (Table A1). Dunn's test was then applied post-hoc to assess pairwise significance of recurrence treatments (Table A2), using the Holm p-value adjustment (Holm, 1979).

No statistical significance between any of the treatments was recorded during testing on the MNIST dataset (Figure 9a), suggesting that recurrence had no effect on the models' ability to reconstruct static spike trains. However, 2-layer recurrence consistently yielded a significantly higher test loss than either of the other two treatments in several experiments conducted on DVS data (Figure 9b, 9c, and 9d). Additionally, no significant difference in test loss was observed between recurrence and 1-layer recurrence across any of the datasets, indicating that these two treatments performed similarly. Given that 2-layer recurrence exhibited a worse test loss than 1-layer recurrence, it is plausible that the observed differences in test performance may arise from factors other than recurrence, potentially due to the increased network depth.

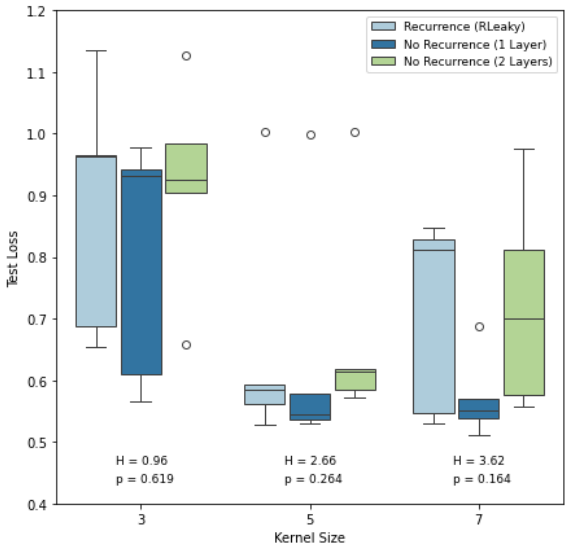


(a) Image slices

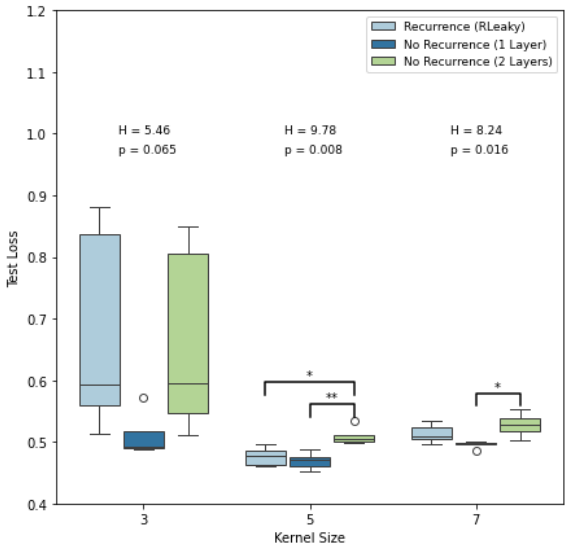


(b) Image raster

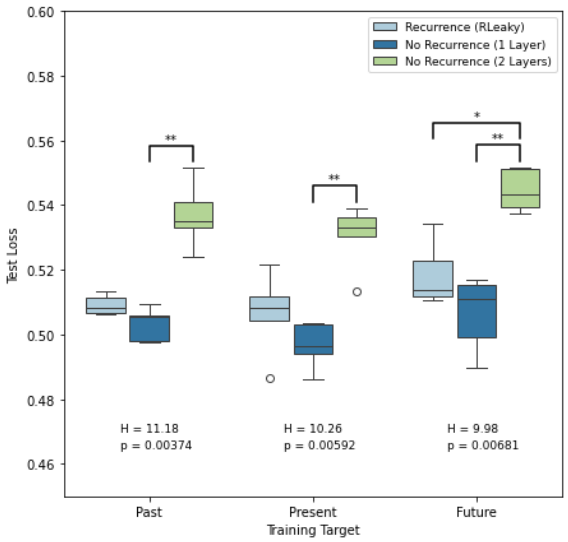
Figure 8: DVS gesture input and reconstruction with implicit recurrence (1-layer). (a) The top row shows the input spiking data at each time-step, averaged over the subsequent 10 time-steps to produce a smoothed trajectory. This input data represents an arm-waving motion performed by a subject with both arms. The bottom row depicts the corresponding reconstruction, similarly averaged over 10 time-steps. (b) The temporal dynamics of the original DVS data are illustrated by the oscillatory envelope of the input spike train. The code spike train represents neurons firing in the code layer, while the reconstructed output spike train shows the ability of the autoencoder to capture temporal patterns, as evidenced by the characteristic oscillatory envelope in the reconstructed data.



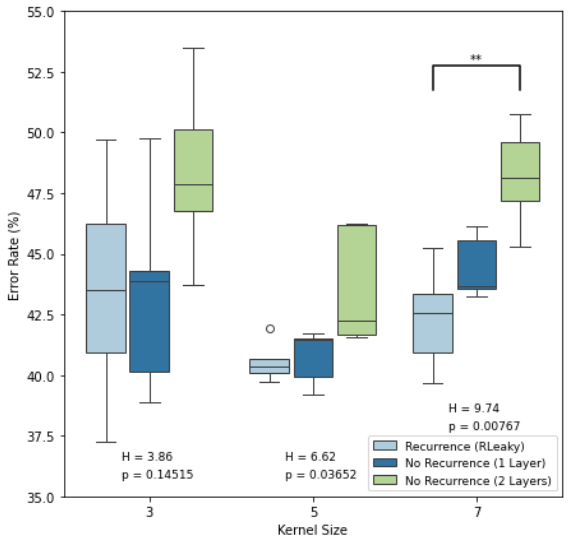
(a) SAE trained on MNIST data with varying kernel size



(b) SAE trained on DVS data with varying kernel size



(c) SAE trained on DVS data with varying training targets



(d) Classifier trained on DVS data with varying kernel size

Figure 9: The effect of introducing recurrence into the latent layer is shown for various trials (a - d). Asterisks *, **, and *** denote significance levels of $p \leq \{0.05, 0.01, 0.001\}$. No asterisk corresponds to $p > 0.05$. No statistical significance was recorded between any instances of RLeaky and 1-Layer. In many cases, the test loss of the 2-Layer treatment was significantly greater than the other two treatments.

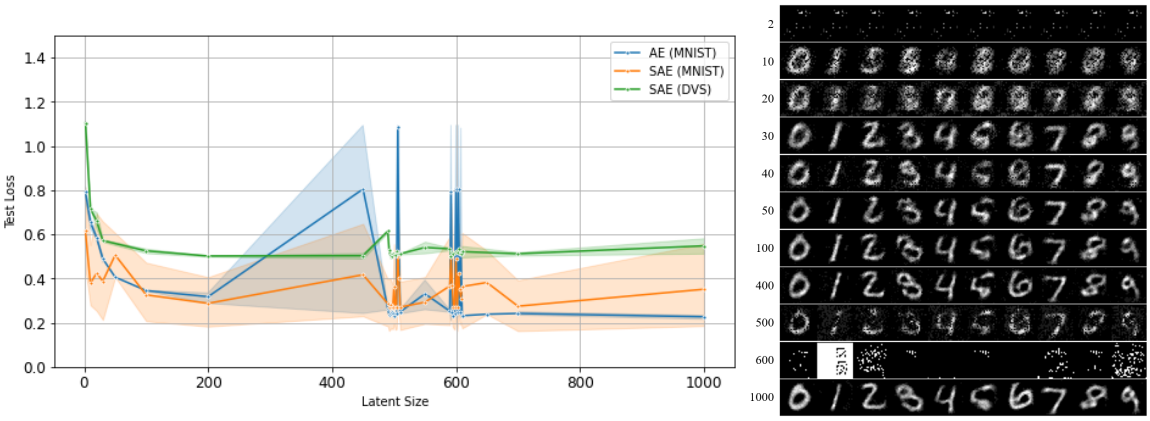
2.4 Training Instability in Spiking and Non-Spiking Autoencoders Occurs at Similar Embedding Sizes

In order to test how the embedding size of the code layer affects the quality of the autoencoder reconstruction, an analysis of the test loss over a wide range of embedding sizes was performed (Section 4.6.4). The number of neurons in the code layer of the autoencoder was varied from 2 to 1000, with the upper limit constrained by hardware capabilities. The code layer was varied while training three networks; a non-spiking autoencoder trained on MNIST, and a spiking autoencoder trained on MNIST and DVS data. Each test condition was conducted over three replicates.

It was observed for all test conditions that the test loss of the autoencoder generally decreased asymptotically as the number of neurons in the code layer was increased (Figure 10a). The benefits of increasing the embedding size were limited beyond approximately 200 neurons. Evaluating this relationship at various kernel sizes indicated that kernel size had limited impact on the unstable conditions, since instability was observed across the full spectrum of tested values (Figure 11).

It was also observed that both spiking and non-spiking models experienced higher reconstruction loss at certain embedding sizes, particularly at around 500 and 600 neurons (Figure 10a). These sizes resulted in poor reconstructions compared to neighbouring values (Figure 10b). Variation in this range was consistent across the three replicates. Deeper analysis of the latent spike trains for unstable embedding sizes consistently revealed bursting activity in neurons within the latent layer (Figure 12).

These results indicate that while increasing the size of the code layer generally improves autoencoder performance for both network types, there are also specific values that prevent learning altogether.



(a) Test Loss against latent space size

(b) Spiking reconstructions

Figure 10: (a) Test loss as a function of embedding size. The test loss is shown for non-spiking MNIST, spiking MNIST, and DVS Gesture datasets over 3 replicates. Variability in performance is observed across different latent sizes, with a notable decline in quality around 500 and 600 neurons. The training curves for spiking and non-spiking autoencoders exhibit similar shapes. (b) MNIST reconstructions are shown for embedding sizes ranging from 2 to 1000, noting that even at the upper limit, there is significant compression compared to the output of the previous layer.

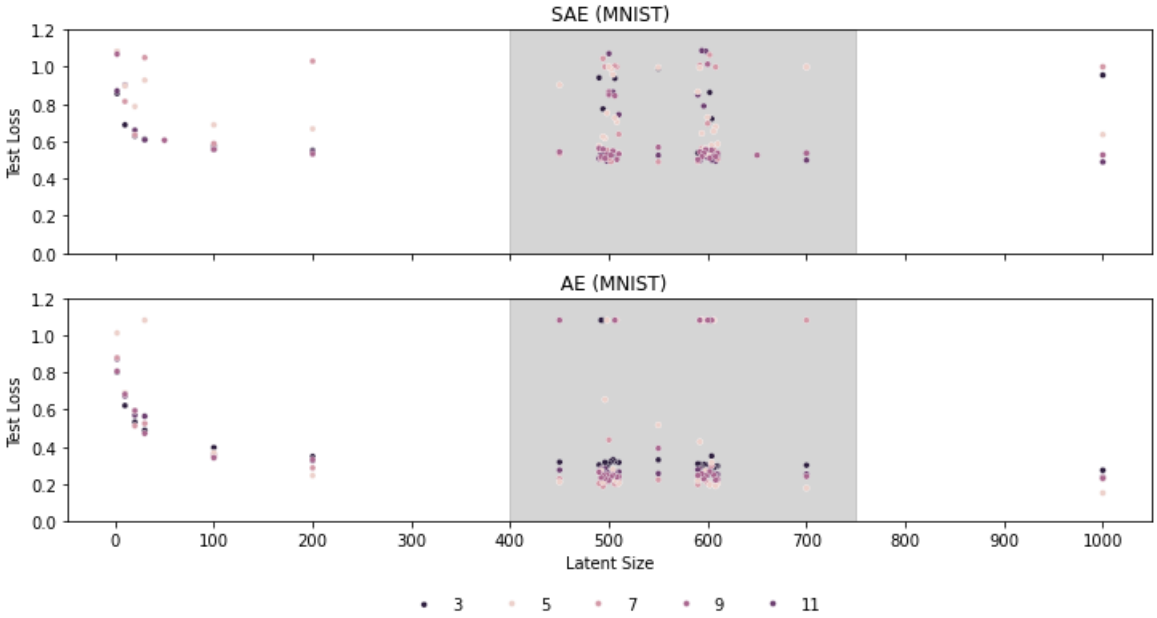
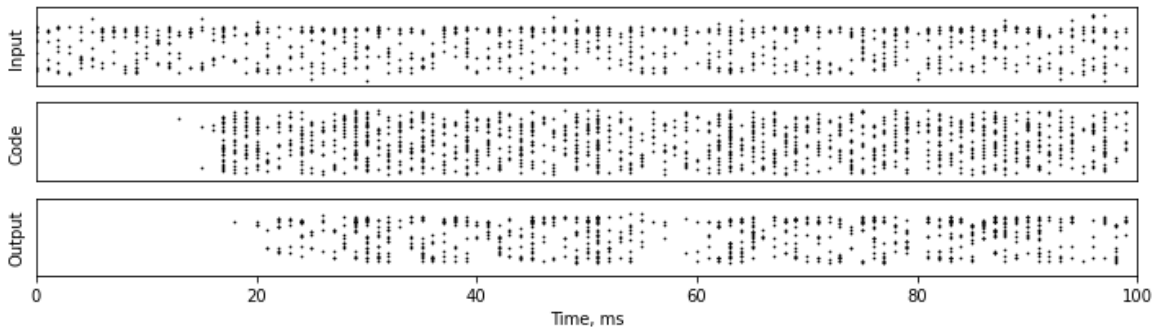
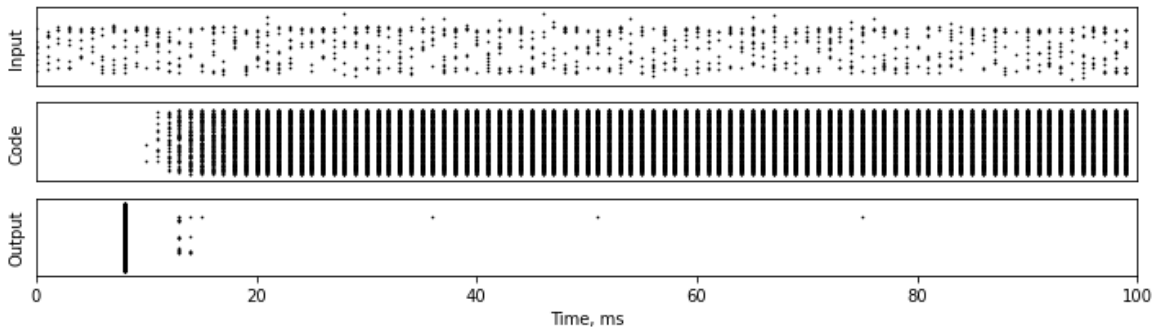


Figure 11: Test loss as a function of embedding size for (a) spiking and (b) non-spiking autoencoders, assessed at different kernel sizes. The main region of instability is indicated by the shaded region. Unstable training was observed across numerous kernel sizes, indicating it is not a significant contributor to stability in this context.



(a) Embedding size of 100



(b) Embedding size of 600

Figure 12: Raster plots are shown for the input, code, and output layers of the spiking autoencoder trained on MNIST with embedding sizes of (a) 100 and (b) 600. Bursting neuron activity in the code layer is evident in (b), which is known to impede learning.

3 Discussion

This work aimed to characterize the behaviour of a SAE trained on both rate-encoded (MNIST) and natively neuromorphic (DVS) data. Three core analyses were conducted to evaluate autoencoder performance. First, the impact of varying the rate-coding paradigm on input data was assessed, revealing a strong dependence on kernel size and frequency coding for high-fidelity reconstructions. Second, different recurrence topologies were analysed to determine the effect of recurrent connections on data reconstruction. It was found that RLeaky and 1-layer recurrence had minimal impact on performance in both supervised and self-supervised networks, while 2-layer recurrence consistently increased reconstruction loss. Finally, an exploration of various embedding sizes was conducted to examine the influence of data compression on reconstruction loss. Increasing the embedding size generally reduced test loss, though several unstable sizes were identified, leading to training failure.

The performance of machine learning models depends heavily on hyperparameter selection. A hyperparameter sweep revealed a significant dependency on kernel size for autoencoder reconstruction loss, with larger kernels reducing test loss in spiking networks (Figure 5c). Similarly, Putra and Shafique (2024) found that larger kernels improved predictive capability in SNNs for classification, hypothesizing that larger kernels extract more information from the input space, enhancing feature generation. Given that the SAE encoder acts as a feature extractor, it is plausible that larger kernels lead to more effective embeddings and therefore more accurate reconstructions.

The rate-coding paradigm significantly influenced the quality of the learned embeddings, with greater disparities between maximum and minimum thresholds promoting more distinct clusters. These clusters correspond to the semantic labels of the MNIST data, which are never provided explicitly, reproducing the results obtained in Ben-Shaul et al. (2023), who conducted an empirical assessment of self-supervised learning representations for different regularisation terms. Models with well-defined latent representations typically exhibited lower reconstruction test loss, suggesting that frequency acts as a contrast control mechanism when representing spike trains in lower-dimensional space. However, higher encoding rates, while improving fidelity, also increase energy requirements (Eshraghian et al., 2023; Prescott and Sejnowski, 2008).

This work also investigated the effect of recurrent connections in the SAE's code layer across multiple training modes and datasets. Recurrence had no impact on performance when applied to MNIST data, consistent with Bouanane et al. (2023), who observed that explicit recurrences in hidden layers had no effect when applied to static datasets. Since recurrence is inherently temporal, they suggested that its benefits for purely spatial data are limited. However, we also found no significant difference between RLeaky and 1-layer recurrence configurations when applied to the DVS dataset, which has a strong temporal component. This contradicts Bouanane et al. (2023), who reported that recurrence improved accuracy of a spiking classifier for the temporally-rich Spiking Heidelberg Digits (SHD) dataset. The discrepancy may be attributed to their use of a cross-entropy max-over-time loss function (Cramer et al., 2022), in contrast to our approach of summing the cross-entropy loss of membrane potential over all time-steps. Additionally, 2-layer implicit recurrence increased test loss across all configurations, possibly due to exacerbated vanishing gradient problems in deeper SNNs (Zheng et al., 2021; Ledinauskas et al., 2020; Guo and Chen, 2024).

Autoencoders create lower dimensional embeddings of their inputs, with the embedding size generally treated as a hyperparameter (Chiang et al., 2019; Ryu et al., 2020). The SAE generally benefitted from larger embedding sizes; however, certain sizes led to increased test loss in both spiking and non-spiking networks. The complex internal learning mechanisms of neural networks often obscure the specific cause of such issues (Dobson, 2023). One possible explanation is that gradient descent struggles to navigate the loss surface of the network at these specific embedding sizes, failing to escape suboptimal local minima. Li et al. (2017) suggests that parameter initialization and skip connections can help regulate the loss surface and stabilizing learning. These techniques be applied in future work, particularly since both have been shown to stabilise learning in SNNs (Benmeziane et al., 2023; Lee et al., 2016).

This study focused on applying SAEs to encoded and natively neuromorphic datasets, with controlled frequency thresholds during encoding. However, neuron firing rates in hidden layers were

unconstrained. Future work could address this by introducing regularization to the loss function to limit firing rates within biological ranges. Hübotter et al. (2021) propose several biologically-inspired regularisation terms, including penalties for high synaptic weights, large deviations of membrane potential from the resting state, and excessive spiking activity. They demonstrated effective MNIST reconstruction while avoiding dead or bursting neurons, both of which were observed in this study (Figure 12).

In this work, we discussed how SNNs are analogous to recurrent neural networks with their implicit recurrent dynamics. This makes them prone to vanishing and exploding gradients (Guo and Chen, 2024; Zheng et al., 2021). These issues arise from unrolling the computational graph over simulated time steps. This compounds the recurrent weights and causes gradients in (2) to become excessively small or large, hindering effective weight updates. Hoyer et al. (2022) propose the biologically-inspired e-prop algorithm, demonstrating its effectiveness for stable sequential MNIST classification over approximately 800 time steps, despite lower overall accuracy than backpropagation-through-time. Future work could explore incorporating the e-prop learning rule into spiking autoencoder training, to determine its effectiveness for learning reconstructions of temporally-rich datasets such as DVS.

This study focused exclusively on spike-count rate encoding and decoding of MNIST data into spiking representations. This is a simplistic approach to neural coding. Numerous alternatives exist, such as temporal, delta, and population rate coding methods. Temporal coding is particularly prevalent in the SNN literature (Comşa et al., 2021; Walters et al., 2024; Stratton et al., 2020), and has been observed in neural regions like the thalamus (Reinagel and Reid, 2000; Wang et al., 2008). It is widely theorised that rate and temporal codes are highly synergistic (Mehta et al., 2002; Ainsworth et al., 2012). Future research could explore the impact of diverse coding schemes on SAE representations to elucidate the interplay between the two mechanisms.

Although this study did not examine the network's learned weight distributions, such an analysis could offer valuable insights. Comşa et al. (2021) analysed weight distributions in an SAE trained on temporally coded data, identifying distinct preferences for inhibitory and excitatory connections. A similar investigation of a rate-coded network could reveal differences in weighting preferences between the two encoding mechanisms, potentially enhancing our understanding of how these networks learn and generalize.

4 Methods

4.1 Datasets

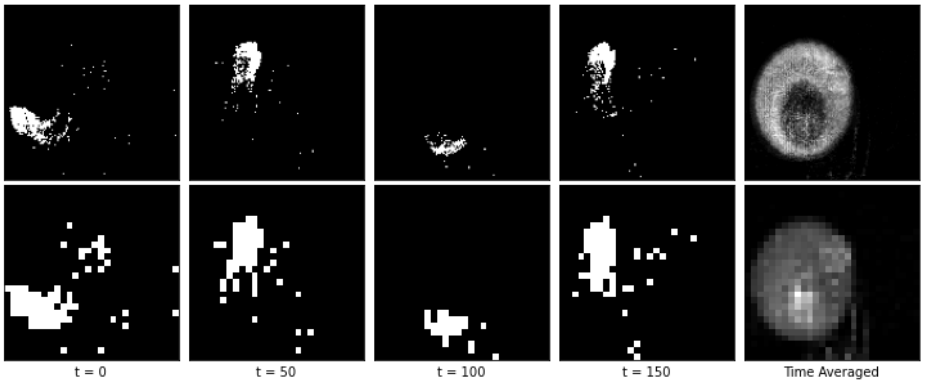
4.1.1 MNIST

The MNIST dataset (Deng, 2012) consists of 70000 (60000 training, 10000 testing) greyscale images of handwritten numerals from 0 to 9 with a 28×28 pixel dimension (Figure 13). A subset of 10% was sampled uniformly to reduce training times on available hardware, for final train and test set sizes of 6000 and 1000 respectively.

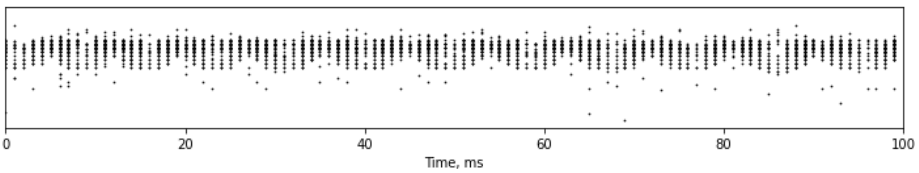
Figure 13: A sample of the ten numerals in the MNIST dataset.

4.1.2 DVS Gesture

The DVS Gesture dataset was used to check the impact of various parameters on data with temporal dynamics. The DVS (Amir et al., 2017) dataset consists of neuromorphic data collected by a DVS128 Event Driven Camera. It contains 11 different gestures performed over 122 trials each, for a total size of 1342 samples. The DVS camera recorded in a native spatial resolution of 128×128 pixels, and microsecond temporal resolution. The data was preprocessed in this work to ease the computational load of training, by reducing the spatial resolution to 28×28 pixels, with millisecond temporal resolution (Figure 14).



(a) Image Slices



(b) Image Raster

Figure 14: (a) A sample of the DVS Gesture dataset is shown at its original resolution of (128, 128) on the top row, and its reduced resolution of (28, 28) on the bottom row. The sample depicts a right-hand waving motion by the subject. Averaging the spike train over time produces the trajectory of the motion, which is shown as the circular pattern in the right-most image of each row. (b) The temporal dynamics within the DVS data are indicated by the oscillatory envelope of the spike trains.

4.2 Spike Encoding

The MNIST dataset was converted to spiking representation using spike-count rate encoding (Figure 15) after normalising the pixel values of each image to the range $[0, 1]$. Each normalised pixel value was treated as the probability of a spike firing under a binomial distribution, sampled at each time-step for T time-steps.

$$S_i^{(t)} \sim Bin(T, p_i) \tag{4}$$

where

$S_i^{(t)} \in \{0, 1\}$ is the random variable describing spike generation from neuron i at time t

T is the total number of time-steps in the spike train

p_i is the probability of a spike occurring from neuron i

A constraint upon the minimum and maximum mean firing rate was imposed upon the encoding scheme. These rates were initially set as $f_{off} = 1$ Hz and $f_{on} = 75$ Hz. Pixel values were mapped to spiking probability as follows:

$$p_i = \Delta t[(f_{on} - f_{off})x_i + f_{off}] \tag{5}$$

where

Δt is the size of the time-step for the simulation

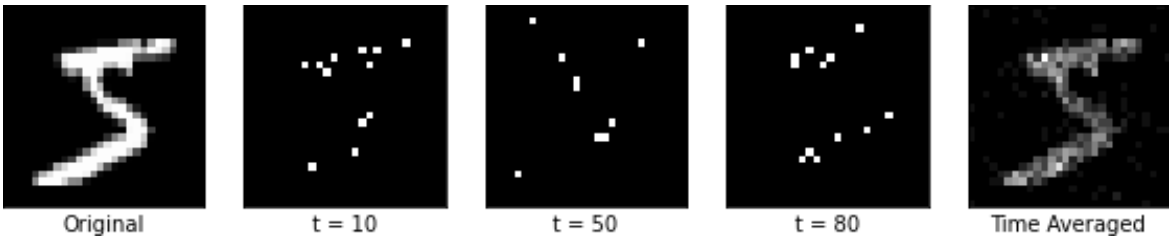
x_i is the normalised value of the ith pixel in the input image

$f_{off} \in [0, \frac{1}{\Delta t}]$ is the minimum firing rate in Hz

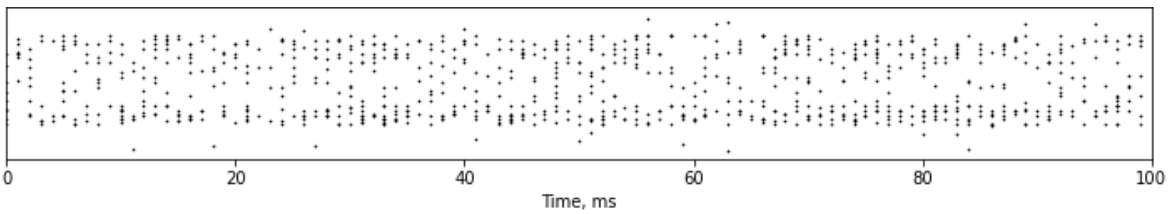
$f_{on} \in [0, \frac{1}{\Delta t}]$ is the maximum firing rate in Hz

The expected spike count from a neuron, μ_f, with a given firing probability was therefore the mean of the distribution which is:

$$\mu_f = pT \tag{6}$$



(a) Encoded image slices



(b) Encoded image raster

Figure 15: (a) The original MNIST image was encoded into spike trains with a total duration of 100 ms. Temporal slices of the image spike train are shown, along with the decoded representation by temporal averaging. (b) The encoded images can also be expressed through a raster plot showing neuron activations over the encoding period.

4.3 Neuron Model

The LIF neuron is a computationally efficient model which preserves key behaviours of a biological neuron, such as membrane potential dynamics and spiking (Stein, 1967; Wang et al., 2014). This makes it a popular choice for spiking neural networks. In this report, the LIF is utilised in the spiking autoencoder. The membrane potential dynamics are governed by the equation:

$$\tau_m \frac{dU(t)}{dt} = -[U(t) - U_{rest}] + R_m I_{ext}(t) \tag{7}$$

where

τ_m is the membrane time constant

$U(t)$ is the membrane potential as a function of time, t

U_{rest} is the resting potential

$I_{ext}(t)$ is the external current supplied

R_m is the membrane resistance

The solution of (7) for a constant current input is:

$$U(t) = R_m I_{ext}(t) + [U_0 - R_m I_{ext}(t)] e^{\frac{-t}{\tau_m}} \tag{8}$$

where U_0 is the initial membrane potential at $t = 0$.

A spike is emitted from the neuron when the membrane potential exceeds a set threshold. After a spike, the membrane potential is reset to a lower potential, which may differ from the resting potential. These dynamics are described by the equations:

$$S_{out}(t) = \begin{cases} 1, & U(t) \geq U_{th} \\ 0, & \text{otherwise} \end{cases}$$
$$\text{if } U(t) \geq U_{th}, \text{ then } U(t) \rightarrow U_{reset} \tag{9}$$

where

$S_{out}(t) \in \{0, 1\}$ is the spiking output of a neuron

U_{th} is the spiking potential threshold

U_{reset} is the reset potential following a spike

The discrete approximation of (8) can be obtained using the forward Euler method:

$$U[t] = \beta U[t-1] + (1-\beta) I_{ext}, \text{ where } \beta = e^{\frac{-\Delta t}{\tau_m}} \tag{10}$$

where β is the decay rate of $U[t]$ and Δt is the simulation time-step size. Note that spiking neural networks possess implicit recurrent dynamics due to the dependence of $U[t]$ on $U[t-1]$.

Equation 9 is non-differentiable and so can't be used directly in backpropagation calculations (Eshraghian et al., 2023). To facilitate backpropagation, a fast-sigmoid surrogate gradient was substituted in the backward pass, as described by equation:

$$S(U) \approx \frac{U}{1 + \alpha |U|} \tag{11}$$

and,

$$\frac{\partial S(U)}{\partial U} = \frac{1}{(1+\alpha|U|)^2} \tag{12}$$

where

$S(U)$ represents the spiking output as a function of the membrane potential.

α is the slope parameter of the sigmoid

4.4 Network Architecture

The general topology of the autoencoder is shown in Figure 16, while specific elements such as embedding (code layer) size and kernel size were varied throughout this work. A more detail breakdown on network architecture is shown in Table 1. A non-spiking variant was also created with the same overall structure, however the LIF neuron activations were replaced with the ReLU activation function.

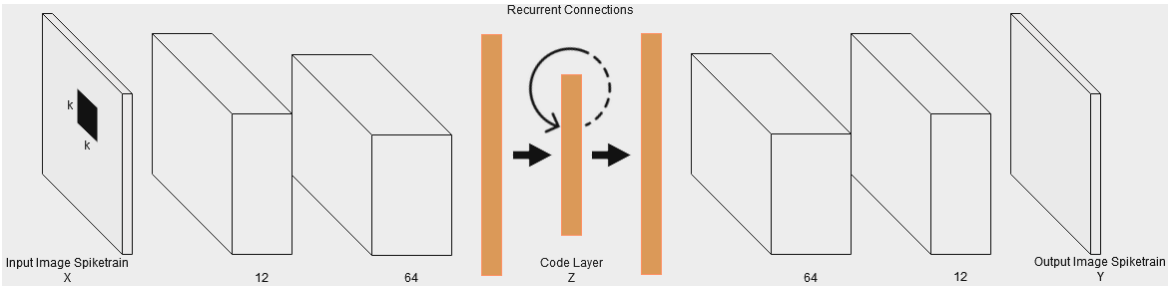


Figure 16: The spiking convolutional autoencoder had two convolutional layers, three fully connected layers, and two transposed convolutional layers. The code layer of the network was deployed with and without recurrent connections. The network accepts tensors whose first dimension describes the evolution of the spike train over a time period, T.

#	Layer	Layer Description	Output Size
0	Input Shape	N/A	$[N, 1, 28, 28]$
1	Convolutional	$[1, 12, 3 \times 3]$	$[N, 12, 26, 26]$
2	Convolutional	$[12, 64, 3 \times 3]$	$[N, 64, 24, 24]$
3	Flattening	$[24 \times 24 \times 64] \rightarrow [36864]$	$[N, 36864]$
4	Fully Connected	$[36864, 100]$	$[N, 100]$
5	Fully Connected (Recurrent)	$[100, 100]$	$[N, 100]$
6	Fully Connected	$[100, 36864]$	$[N, 36864]$
7	Unflattening	$[36864] \rightarrow [24 \times 24 \times 64]$	$[N, 64, 24, 24]$
8	Transposed Convolutional Layer	$[64, 12, 3 \times 3]$	$[N, 12, 26, 26]$
9	Transposed Convolutional Layer	$[12, 1, 3 \times 3]$	$[N, 1, 28, 28]$

Table 1: A detailed description of network architecture with a kernel size of 3×3 and an embedding size of 100. This represents a specific instance of the network and that these parameters were varied throughout this work. The notation used in convolutional layer descriptions is [Input Channels, Output Channels, Kernel Size], while the notation used for fully connected layer descriptions is [Input Channels, Output Channels]. The output sizes are given in the form [Batch Size, Channels, Height, Width]. N in this table corresponds to the batch size used in training and was also varied.

4.5 Loss Function

Choosing an appropriate loss function is crucial for training a machine learning model via gradient descent. In this work, all autoencoder loss functions were applied to the input and output spike counts. The total spike count per neuron i, X_i, at the input or output, was defined as a function of the spike trains $S_i^{(t)}$:

$$X_i = \sum_{t=0}^{T} S_i^{(t)} \tag{13}$$

where T is the total number of time-steps in the spike train.

The root mean-squared error (RMSE) is a common loss metric in machine learning, which, when applied to spike counts, is expressed as:

$$L(\theta)_{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(X_i - \hat{X}_i)^2} \tag{14}$$

where

$L(\theta)_{RMSE}$ is the RMSE loss as a function of the network parameters.

N is the number of neurons.

X_i is the input spike count of the ith neuron.

$\hat{X}_i$ is the output spike count of the ith neuron.

However, RMSE is limited when applied to spike counts under varying frequency encoding schemes. Higher frequency encodings naturally increase spike counts, leading to a heightened error estimation despite high reconstruction fidelity. To address this, a normalized root mean-squared error (nRMSE) was defined by dividing (14) by the standard deviation of the input spike counts:

$$L(\theta)_{nRMSE} = \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(X_i - \hat{X}_i)^2}}{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(X_i - \bar{X})^2}} = \frac{L(\theta)_{RMSE}}{\sigma_X} \tag{15}$$

where

$\bar{X}$ is the mean input spike count.

σ_X is the standard deviation of the input data.

This normalization makes the loss comparable across different datasets, adjusting for natural variations and decoupling the loss from specific input characteristics. This approach yields a test loss of approximately 1 if the autoencoder output approximates the mean of the input data, and 0 for a perfect reconstruction. Consequently, it enables direct comparisons between models trained on diverse input data, including between spiking and non-spiking models.

Taking the spike count over the entire simulation period is suitable for static data like MNIST, but is not sufficient to capture temporal dynamics in datasets like DVS. In such cases, nRMSE loss was computed using spike counts across discrete temporal chunks of length $k \leq T$:

$$X_i^{(n)} = \sum_{t=nk}^{(n+1)k} S_i^{(t)} \tag{16}$$

where $X_i^{(n)}$ is the spike count of neuron i between time nk and time $(n+1)k$.

When training the network on DVS data, the total loss was calculated by summing the losses over each chunk in the sequence:

$$L_{total} = \sum_{i=1}^{N} \sum_{n=0}^{K} L_{nRMSE}(X_i^{(n)}, \hat{X}_i^{(n)}) \tag{17}$$

where

$X_i^{(n)}$ is the spike count of neuron i over a chunk n

$\hat{X}_i^{(n)}$ is the spike count of output neuron i over a chunk n

$K = \frac{T}{k} - 1$ is the number of chunks in the full sequence of length T with a chunk size of k

4.6 Experiments

4.6.1 Baseline Hyperparameter Grid Search

A hyperparameter grid search was conducted (Table 2), and the parameters corresponding to the lowest test loss selected as the foundation for all future test configurations, except where explicitly varied.

Parameter	Test Set
Batch Size	{8, 16, 32, 64, 128}
Learning Rate	{1e-2, 1e-3, 1e-4, 1e-5, 1e-6}
Kernel Size	{3, 5, 7, 9, 11, 13}
Subset Size	10%
Latent Layer Size	$\{x \in \mathbb{N} \mid 2 \leq x \leq 1000\}$
Epochs	10
Membrane Time Constant, τ_m	24 ms
Sigmoid Surrogate Gradient Slope, α	25
Membrane Potential Decay Rate, β	0.959
Voltage Threshold, U_{th}	1 V
Temporal Chunk Size, k	20

Table 2: A grid search was conducted across various learning rates and batch sizes to establish values for minimising the error and runtime for the base model. Note that for the Latent Layer Size, the specific subset that was used is shown in Section 2.4. Additionally, note that β was calculated from $\beta = e^{\frac{-\Delta t}{\tau_m}}$ where $\Delta t = 1$ ms.

4.6.2 Analysing the Effect of Varying Frequency Coding Schemes

Model performance was evalauted across a selection of minimum and maximum encoding firing rates for the input data (Section 4.2). First, the test loss was recorded for various maximum encoding frequencies ({1, 10, 20, 30, 40, 50, 75, 100, 125, 150, 175, 200} Hz) while maintaining a constant minimum frequency of 1 Hz. To reflect the baseline activity observed in biological neurons, the minimum firing rate was always set above zero.

Next, the full range of encoding frequencies was analysed by assessing performance for various f_{on} and f_{off} ratios ({1, 50, 100, 150, 200} Hz). These constraints were applied solely to the initial encoding, with no restrictions on the firing rates in the hidden layers.

4.6.3 Introducing Recurrent Neurons into the Latent Space

Biological neurons are interconnected in complex structures which can exhibit feedforward, lateral, and feedback connections. Fully connected recurrent LIF dynamics were introduced into the code layer of the network, which summed the output spikes with the input spikes for the following time-step. The snnTorch (Eshraghian et al., 2023) implementation of recurrence includes the insertion of a dense or convolutional layer between the output and input of the recurrent layer (Figure 17).

This implementation of recurrence introduces additional $N(N+1)$ parameters into the network where N corresponds to the size of the recurrent layer. To isolate recurrent spiking as the only influence on performance, rather than the additional learnable parameters, an extra fully connected layer of size N was inserted into the non-recurrent version of the network. This ensured that the total number of parameters remained constant between architectures, and only the topology of the connections varied. Three recurrence structures were analysed in this report over a variety of kernel sizes and training targets.

1. **Recurrence (RLeaky)**: Explicit recurrence implemented with snnTorch's RLeaky neuron type.
2. **No Recurrence (1 Layer)**: Implicit recurrence with a single code or hidden layer.
3. **No Recurrence (2 Layer)**: Implcit recurrence with a second linear + LIF layer. This method keeps the total number of parameters equal to the RLeaky method.

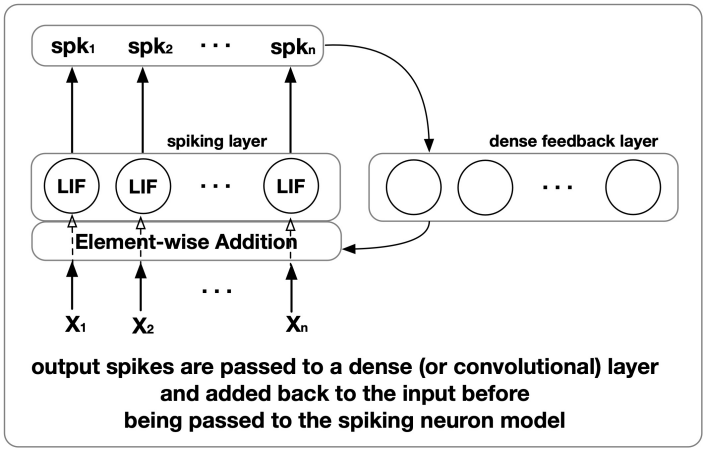


Figure 17: Diagram from Eshraghian and Henkes (n.d.) outlining their implementation of the recurrent LIF neuron. Henkes et al. (2022) presents the recurrent LIF neuron in additional detail.

The three types of recurrence were evaluated over four different network structures and training regimes.

1. SAE trained on MNIST data with varying kernel size (3, 5, 7)
2. SAE trained on DVS data with varying kernel size (3, 5, 7)
3. SAE trained on DVS data with varying training targets (Past, Present, Future)
4. Classifier trained on DVS data with varying kernel size (3, 5, 7)

The different training targets for item 3 refer to the position of the target chunk, $\hat{X}^k$, relative to the input chunk, X^k, in the loss function (17). Loss functions for past (L_{Past}), present ($L_{Present}$), and future (L_{Future}) targets are shown as follows:

$$\begin{aligned} L_{Past} &= L(X_{t:t+k}, \hat{X}_{t-k:t}) \\ L_{Present} &= L(X_{t:t+k}, \hat{X}_{t:t+k}) \\ L_{Future} &= L(X_{t:t+k}, \hat{X}_{t+k:t+2k}) \end{aligned} \tag{18}$$

where $\hat{X}_{t:t+k}$ denotes a chunk of autoencoder reconstruction starting at time-step t and including the following k time-steps.

The classifier was created by removing the decoder portion of the SAE, taking the output at the code layer. The output membrane potential at each time-step was used as logits in the softmax function, and the cross-entropy taken between the membrane potential and target class label. The cross-entropy loss was summed over all time-steps to encourage the neuron corresponding to the correct class to have the highest membrane potential at all time-steps.

4.6.4 Analysing Model Performance for Different Embedding Sizes

The number of neurons in the latent layer has a direct impact on how much of the information in the input is preserved during the encoding process. The network was trained using a variety of different latent space sizes (Table 2) to assess the resulting impact on the test loss following training on MNIST and DVS datasets. A qualitative assessment of the reconstructed images was also performed.

References

Acharya, U. R., Oh, S. L., Hagiwara, Y., Tan, J. H., Adam, M., Gertych, A. and Tan, R. S. (2017), 'A deep convolutional neural network model to classify heartbeats', *Computers in Biology and Medicine* **89**, 389–396. 2

Ainsworth, M., Lee, S., Cunningham, M. O., Traub, R. D., Kopell, N. J. and Whittington, M. A. (2012), 'Rates and rhythms: a synergistic view of frequency and temporal coding in neuronal networks', *Neuron* **75**(4), 572–583. 5, 16

Alkabaa, A. S., Taylan, O., Yilmaz, M. T., Nazemi, E. and Kalmoun, E. M. (2022), 'An investigation on spiking neural networks based on the izhikevich neuronal model: Spiking processing and hardware approach', *Mathematics* **10**, 612. 3

Amir, A., Taba, B., Berg, D., Melano, T., McKinstry, J., Di Nolfo, C., Nayak, T., Andreopoulos, A., Garreau, G., Mendoza, M., Kusnitz, J., Debole, M., Esser, S., Delbruck, T., Flickner, M. and Modha, D. (2017), A low power, fully event-based gesture recognition system, *in* '2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)', pp. 7388–7397. 17

Anthony, L. F. W., Kanding, B. and Selvan, R. (2020), 'Carbontracker: Tracking and predicting the carbon footprint of training deep learning models'. 3

Beaulieu-Jones, B. K. and Moore, J. H. (2017), Missing data imputation in the electronic health record using deeply learned autoencoders, WORLD SCIENTIFIC, pp. 207–218. 2, 7

Ben-Shaul, I., Shwartz-Ziv, R., Galanti, T., Dekel, S. and LeCun, Y. (2023), 'Reverse engineering self-supervised learning'.
URL: *https://arxiv.org/abs/2305.15614* 15

Benmeziane, H., Ounnoughene, A. Z., Hamzaoui, I. and Bouhadjar, Y. (2023), Skip connections in spiking neural networks: An analysis of their effect on network training, *in* '2023 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)', pp. 790–794. 15

Bi, G.-Q. and Poo, M.-M. (1998), 'Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type', *The Journal of Neuroscience* **18**, 10464–10472. 4

Bobrowski, O., Meir, R. and Eldar, Y. C. (2009), 'Bayesian filtering in spiking neural networks: Noise, adaptation, and multisensory integration', *Neural Computation* **21**, 1277–1320. 3

Bouanane, M. S., Cherifi, D., Chicca, E. and Khacef, L. (2023), 'Impact of spiking neurons leakages and network recurrences on event-based spatio-temporal pattern recognition', *Frontiers in Neuroscience* **17**. 10, 15

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I. and Amodei, D. (2020), 'Language models are few-shot learners'. 3

Brunel, N. and van Rossum, M. C. W. (2007), 'Lapicque's 1907 paper: from frogs to integrate-and-fire', *Biological Cybernetics* **97**, 337–339. 3

Burbank, K. S. (2015), 'Mirrored stdp implements autoencoder learning in a network of spiking neurons', *PLOS Computational Biology* **11**, e1004566. 6

Byerly, A., Kalganova, T. and Dear, I. (2020), 'No routing needed between capsules'. 5

Cao, Y., Chen, Y. and Khosla, D. (2015), 'Spiking deep convolutional neural networks for energy-efficient object recognition', *International Journal of Computer Vision* **113**, 54–66. 6

Caporale, N. and Dan, Y. (2008), 'Spike timing–dependent plasticity: A hebbian learning rule', *Annual Review of Neuroscience* **31**, 25–46. 4

Chaturvedi, S. and Khurshid, A. (2011), Review of spiking neural network architecture for feature extraction and dimensionality reduction, IEEE, pp. 317–322. 3

Cheng, Z., Sun, H., Takeuchi, M. and Katto, J. (2018), Deep convolutional autoencoder-based lossy image compression, IEEE, pp. 253–257. 2, 7

Chiang, H.-T., Hsieh, Y.-Y., Fu, S.-W., Hung, K.-H., Tsao, Y. and Chien, S.-Y. (2019), 'Noise reduction in ecg signals using fully convolutional denoising autoencoders', *IEEE Access* **7**, 60806–60813. 2, 15

Comşa, I.-M., Versari, L., Fischbacher, T. and Alakuijala, J. (2021), 'Spiking autoencoders with temporal coding', *Front. Neurosci.* **15**, 712667. 5, 6, 16

Cramer, B., Stradmann, Y., Schemmel, J. and Zenke, F. (2022), 'The heidelberg spiking data sets for the systematic evaluation of spiking neural networks', *IEEE Trans. Neural Netw. Learn. Syst.* **33**(7), 2744–2757. 15

Dampfhoffer, M., Mesquida, T., Valentian, A. and Anghel, L. (2022), *Investigating Current-Based and Gating Approaches for Accurate and Energy-Efficient Spiking Recurrent Neural Networks*, pp. 359–370. 3

Deng, L. (2012), 'The mnist database of handwritten digit images for machine learning research', *IEEE Signal Processing Magazine* **29**(6), 141–142. 17

Dhar, P. (2020), 'The carbon impact of artificial intelligence', *Nature Machine Intelligence* **2**, 423–425. 3

Ding, J., Dong, B., Heide, F., Ding, Y., Zhou, Y., Yin, B. and Yang, X. (2022), 'Biologically inspired dynamic thresholds for spiking neural networks'. 3

Dobson, J. E. (2023), 'On reading and interpreting black box deep neural networks', *Int. J. Digit. Humanit.* . 15

Dreiseitl, S. and Ohno-Machado, L. (2002), 'Logistic regression and artificial neural network classification models: a methodology review', *Journal of Biomedical Informatics* **35**, 352–359. 2, 3

Eom, J., Park, I. Y., Kim, S., Jang, H., Park, S., Huh, Y. and Hwang, D. (2021), 'Deep-learned spike representations and sorting via an ensemble of auto-encoders', *Neural Networks* **134**, 131–142. 2

Eshraghian, J. K. and Henkes, A. (n.d.), 'Regression with snns: Part ii'.
URL: *https://snntorch.readthedocs.io/en/latest/tutorials/tutorial*$_{r}egression_{2}.html$23

Eshraghian, J. K., Ward, M., Neftci, E. O., Wang, X., Lenz, G., Dwivedi, G., Bennamoun, M., Jeong, D. S. and Lu, W. D. (2023), Training spiking neural networks using lessons from deep learning, pp. 1016–1054. iii, 3, 4, 6, 15, 19, 22

Gehrig, M., Shrestha, S. B., Mouritzen, D. and Scaramuzza, D. (2020), Event-based angular velocity regression with spiking networks, IEEE, pp. 4195–4202. 3

Gerstner, W., Kempter, R., van Hemmen, J. L. and Wagner, H. (1996), 'A neuronal learning rule for sub-millisecond temporal coding', *Nature* **383**, 76–78. 4, 6

Gerstner, W., Kreiter, A. K., Markram, H. and Herz, A. V. M. (1997), 'Neural codes: Firing rates andbeyond', *Proceedings of the National Academy of Sciences* **94**, 12740–12741. 5

Greeshma, K. V. and Sreekumar, K. (2019), 'Hyperparameter optimization and regularization on fashion-mnist classification', *International Journal of Recent Technology and Engineering (IJRTE)* **8**, 3713–3719. 5

Guo, Y. and Chen, Y. (2024), 'Take a shortcut back: Mitigating the gradient vanishing for training spiking neural networks'. 15, 16

Han, B. and Roy, K. (2020), *Deep Spiking Neural Network: Energy Efficiency Through Time Based Coding*, pp. 388–404. 4, 6

Hao, Y., Huang, X., Dong, M. and Xu, B. (2020), 'A biologically plausible supervised learning method for spiking neural networks using the symmetric stdp rule', *Neural Networks* **121**, 387–395. 4

Henkes, A., Eshraghian, J. K. and Wessels, H. (2022), 'Spiking neural networks for nonlinear regression'.
URL: *https://arxiv.org/abs/2210.03515* 23

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**, 500–544. 3

Holm, S. (1979), 'A simple sequentially rejective multiple test procedure', *Scandinavian Journal of Statistics* **6**, 65–70.
URL: *https://api.semanticscholar.org/CorpusID:122415379* 11, 30

Hoyer, M., Eivazi, S. and Otte, S. (2022), *Efficient LSTM Training with Eligibility Traces*, Springer Nature Switzerland, p. 334–346.
URL: *http://dx.doi.org/10.1007/978-3-031-15934-3*$_2$8 16

Hübotter, J. F., Lanillos, P. and Tomczak, J. M. (2021), 'Training deep spiking auto-encoders without bursting or dying neurons through regularization'.
URL: *https://arxiv.org/abs/2109.11045* 16

Iakymchuk, T., Rosado-Muñoz, A., Guerrero-Martínez, J. F., Bataller-Mompeán, M. and Francés-Víllora, J. V. (2015), 'Simplified spiking neural network architecture and stdp learning algorithm applied to image classification', *EURASIP Journal on Image and Video Processing* **2015**, 4. 3, 4, 5, 6

Izhikevich, E. (2003), 'Simple model of spiking neurons', *IEEE Transactions on Neural Networks* **14**, 1569–1572. 3

Johnson, E. C., Jones, D. L. and Ratnam, R. (2016), 'A minimum-error, energy-constrained neural code is an instantaneous-rate code', *Journal of Computational Neuroscience* **40**, 193–206. 5

Juárez-Lora, A., García-Sebastián, L. M., Ponce-Ponce, V. H., Rubio-Espino, E., Molina-Lozano, H. and Sossa, H. (2022), 'Implementation of kalman filtering with spiking neural networks', *Sensors* **22**, 8845. 3

Kamata, H., Mukuta, Y. and Harada, T. (2022), Fully spiking variational autoencoder, Vol. 36. 6

Kheradpisheh, S. R., Ganjtabesh, M., Thorpe, S. J. and Masquelier, T. (2018), 'Stdp-based spiking deep convolutional neural networks for object recognition', *Neural Networks* **99**, 56–67. 4, 6

Kingma, D. P. and Welling, M. (2019), 'An introduction to variational autoencoders', *Foundations and Trends® in Machine Learning* **12**, 307–392. 2, 7

Lava (n.d.), 'Dynamics, neurons, and spikes — lava documentation'.
URL: *https://lava-nc.org/lava-lib-dl/slayer/notebooks/neuron_dynamics/dynamics.html* 3

Ledinauskas, E., Ruseckas, J., Juršėnas, A. and Buračas, G. (2020), 'Training deep spiking neural networks'. 15

Lee, C., Panda, P., Srinivasan, G. and Roy, K. (2018), 'Training deep spiking convolutional neural networks with stdp-based unsupervised pre-training followed by supervised fine-tuning', *Frontiers in Neuroscience* **12**. 4, 6

Lee, J. H., Delbruck, T. and Pfeiffer, M. (2016), 'Training deep spiking neural networks using backpropagation', *Front. Neurosci.* **10**, 508. 15

Li, H., Xu, Z., Taylor, G., Studer, C. and Goldstein, T. (2017), 'Visualizing the loss landscape of neural nets'. 15

Liu, T., Wang, J., Liu, Q., Alibhai, S., Lu, T. and He, X. (2023), 'High-ratio lossy compression: Exploring the autoencoder to compress scientific data', *IEEE Transactions on Big Data* **9**, 22–36. 2, 7

Maass, W. (1997), 'Networks of spiking neurons: The third generation of neural network models', *Neural Networks* **10**, 1659–1671. 2, 3

Malcolm, K. and Casco-Rodriguez, J. (2023), 'A comprehensive review of spiking neural networks: Interpretation, optimization, efficiency, and best practices'. 4

Marblestone, A. H., Wayne, G. and Kording, K. P. (2016), 'Toward an integration of deep learning and neuroscience', *Frontiers in Computational Neuroscience* **10**. 6

McInnes, L., Healy, J. and Melville, J. (2018), 'Umap: Uniform manifold approximation and projection for dimension reduction'. cite arxiv:1802.03426Comment: Reference implementation available at http://github.com/lmcinnes/umap.
URL: *http://arxiv.org/abs/1802.03426* 9

Mehta, M. R., Lee, A. K. and Wilson, M. A. (2002), 'Role of experience and oscillations in transforming a rate code into a temporal code', *Nature* **417**(6890), 741–746. 16

Neftci, E. O., Mostafa, H. and Zenke, F. (2019), 'Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks', *IEEE Signal Processing Magazine* **36**, 51–63. 4, 10

Nocentini, O., Kim, J., Bashir, M. Z. and Cavallo, F. (2022), 'Image classification using multiple convolutional neural networks on the fashion-mnist dataset', *Sensors* **22**, 9544. 5

Prescott, S. A. and Sejnowski, T. J. (2008), 'Spike-rate coding and spike-time coding are affected oppositely by different adaptation mechanisms', *The Journal of Neuroscience* **28**, 13649–13661. 5, 15

Putra, R. V. W. and Shafique, M. (2024), 'A methodology for improving accuracy of embedded spiking neural networks through kernel size scaling'.
URL: *https://arxiv.org/abs/2404.01685* 15

Ravi, S. (2017), 'Projectionnet: Learning efficient on-device deep networks using neural projections'. 5

Reinagel, P. and Reid, R. C. (2000), 'Temporal coding of visual information in the thalamus', *J. Neurosci.* **20**(14), 5392–5400. 16

Rice, K. L., Bhuiyan, M. A., Taha, T. M., Vutsinas, C. N. and Smith, M. C. (2009), Fpga implementation of izhikevich spiking neural networks for character recognition, IEEE, pp. 451–456. 3

Rifai, S., Vincent, P., Muller, X., Glorot, X. and Bengio, Y. (2011), Contractive auto-encoders: explicit invariance during feature extraction, *in* 'Proceedings of the 28th International Conference on International Conference on Machine Learning', ICML'11, Omnipress, Madison, WI, USA, p. 833–840. 2

Ryu, S., Kim, M. and Kim, H. (2020), 'Denoising autoencoder-based missing value imputation for smart meters', *IEEE Access* **8**, 40656–40666. 2, 15

Schultz, W., Dayan, P. and Montague, P. R. (1997), 'A neural substrate of prediction and reward', *Science* **275**. 6

Shen, J., Liu, J. K. and Wang, Y. (2021), 'Dynamic spatiotemporal pattern recognition with recurrent spiking neural network', *Neural Computation* pp. 1–25. 6

Sinha, A., Singh, M. and Krishnamurthy, B. (2019), *Neural Networks in an Adversarial Setting and Ill-Conditioned Weight Space*, pp. 177–190. 5

Song, S., Miller, K. D. and Abbott, L. F. (2000), 'Competitive hebbian learning through spike-timing-dependent synaptic plasticity', *Nature Neuroscience* **3**, 919–926. iii, 4, 5

Specht, D. (1991), 'A general regression neural network', *IEEE Transactions on Neural Networks* **2**, 568–576. 2, 3

Stein, R. (1967), 'Some models of neuronal variability', *Biophysical Journal* **7**, 37–68. 3, 19

Stratton, P., Wabnitz, A. and Hamilton, T. J. (2020), A spiking neural network based auto-encoder for anomaly detection in streaming data, *in* '2020 IEEE Symposium Series on Computational Intelligence (SSCI)', pp. 1981–1988. 16

Tal, D. and Schwartz, E. L. (1997), 'Computing with the leaky integrate-and-fire neuron: Logarithmic computation and multiplication', *Neural Computation* **9**, 305–318. 3

Tavanaei, A. and Maida, A. S. (2017), Multi-layer unsupervised learning in a spiking convolutional neural network, IEEE, pp. 2023–2030. 6

Toğaçar, M., Cömert, Z. and Ergen, B. (2021), 'Intelligent skin cancer detection applying autoencoder, mobilenetv2 and spiking neural networks', *Chaos, Solitons and Fractals* **144**. 2

Walters, B., Cai, Z., Kalatehbali, H. R., Amirsoleimani, A., Genov, R., Eshraghian, J. and Azghadi, M. R. (2024), Spiking auto-encoder using error modulated spike timing dependant plasticity, *in* '2024 IEEE International Symposium on Circuits and Systems (ISCAS)', pp. 1–5. 16

Wang, X., Lu, T., Bendor, D. and Bartlett, E. (2008), 'Neural coding of temporal information in auditory thalamus and cortex', *Neuroscience* **154**, 294–303. 6, 16

Wang, Z., Guo, L. and Adjouadi, M. (2014), 'A generalized leaky integrate-and-fire neuron model with fast implementation method', *International Journal of Neural Systems* **24**, 1440004. 3, 19

Wu, J., Chua, Y., Zhang, M., Li, H. and Tan, K. C. (2018), 'A spiking neural network framework for robust sound classification', *Frontiers in Neuroscience* **12**. 3

Wu, X., Zhao, Y., Song, Y., Jiang, Y., Bai, Y., Li, X., Zhou, Y., Yang, X. and Hao, Q. (2023), 'Dynamic threshold integrate and fire neuron model for low latency spiking neural networks', *Neurocomputing* **544**, 126247. 3

Yan, Z., Zhou, J. and Wong, W.-F. (2021), 'Energy efficient ecg classification with spiking neural network', *Biomedical Signal Processing and Control* **63**, 102170. 3

Yin, B., Corradi, F. and Bohté, S. M. (2021), 'Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks', *Nature Machine Intelligence* **3**, 905–913. 6

Zhang, W. and Li, P. (2019), Spike-train level backpropagation for training deep recurrent spiking neural networks, *in* H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox and R. Garnett, eds, 'Advances in Neural Information Processing Systems', Vol. 32, Curran Associates, Inc.
URL: *https://proceedings.neurips.cc/paper_files/paper/2019/file/f42a37d114a480b6b57b60ea9a14a9d2-Paper.pdf* 6

Zheng, H., Wu, Y., Deng, L., Hu, Y. and Li, G. (2021), 'Going deeper with directly-trained larger spiking neural networks', *Proc. Conf. AAAI Artif. Intell.* **35**(12), 11062–11070. 15, 16

A Data for Recurrence Assessment in Section 2.3

Test Loss ($\mu \pm \sigma$)		Explicit Recurrence			*H*-statistic	*p*-value
Dataset	**Kernel Size / Training Target**	**False (2 Layers)**	**False (1 Layer)**	**True (RLeaky)**		
Encoded MNIST	3	0.920 ± 0.170	0.881 ± 0.205	0.806 ± 0.200	0.96	0.619
	5	0.678 ± 0.184	0.654 ± 0.197	0.638 ± 0.203	2.66	0.264
	7	0.571 ± 0.068	0.506 ± 0.013	0.496 ± 0.012	3.62	0.164
DVS[a]	3	0.530 ± 0.010	0.506 ± 0.013	0.497 ± 0.007	5.46	0.065
	5	0.545 ± 0.008	0.519 ± 0.015	0.507 ± 0.012	9.78	0.008
	7	0.572 ± 0.014	0.518 ± 0.013	0.507 ± 0.009	8.24	0.016
DVS[a]	Past	0.530 ± 0.010	0.506 ± 0.013	0.497 ± 0.007	10.26	0.006
	Present	0.545 ± 0.008	0.519 ± 0.015	0.507 ± 0.012	11.18	0.004
	Future	0.572 ± 0.014	0.518 ± 0.013	0.507 ± 0.009	9.98	0.007
DVS[c]	3	0.530 ± 0.010	0.506 ± 0.013	0.496 ± 0.007	3.86	0.145
	5	0.545 ± 0.008	0.519 ± 0.015	0.507 ± 0.012	6.62	0.037
	7	0.572 ± 0.014	0.518 ± 0.013	0.507 ± 0.009	9.74	0.008

Table A1: The table shows the mean and standard deviation of the test loss over 5 replicates for different datasets, kernel sizes, and training targets across recurrence treatments. The degrees of freedom was 2 in each case. The DVS dataset was analysed using an autoencoder (DVS[a]) and classification (DVS[c]) network. Significant results are highlighted and investigated further using Dunn's test.

Kernel Size	Comparison	p-value	
		DVS[a]	DVS[c]
5	1 vs 3	0.039	0.071
	2 vs 3	0.009	0.071
7	1 vs 3	0.322	0.006
	2 vs 3	0.014	0.154

(a) Kernel Sizes

State	Comparison	p-value
Present	2 vs 3	0.004
Past	2 vs 3	0.003
Future	1 vs 3	0.047
	2 vs 3	0.007

(b) Training Targets

Table A2: Dunn's Test was applied as a post-hoc assessment following application of the Kruskal Wallis test to identify pair-wise significance between data summarised in Table A1. A Holm-Bonferroni adjustment (Holm, 1979) was applied to reduce the likelihood of Type I errors arising due to multiple comparisons. Statistically significant results are highlighted. The values of 1, 2, and 3 for the comparison column correspond to the different recurrence treatments in the order shown in figure 9 legends.