Visual feature extraction through brain inspired algorithms: Towards efficiency, accuracy and continual learning

Alex Vicente Sola

A thesis submitted for the degree of

Doctor of Philosophy

Neuromorphic Sensor Signal Processing Lab

Centre for Signal and Image Processing

Department of Electronic and Electrical Engineering

University of Strathclyde

Glasgow

2025

Declaration

Signed: Alex Vicente Sola
Date: 30th August 2024

Acknowledgement

First, I want to express my gratitude to my PhD supervisor Dr Gaetano Di Caterina, for his guidance, support, and the trust he always placed in me. He was always there whenever I needed anything and always found time for his PhD students.

I want to thank Dr Trevor J Bihl for his constant involvement in this PhD, his guidance and his eagerness to collaborate, which he always approached with an open mind and an interest for new research questions.

My appreciation also goes to Dr Marc Masana, for lending me his brains in our collaboration, which I really enjoyed, and for the priceless help and guidance he was always keen to give me. Also extended thanks to TU Graz University for having me as visiting researcher during this collaboration.

To Dr Paul Kirkland, for being every single day there, I will always be grateful. His passion and character inspired everyone in the lab, and he always went the extra mile to help all of us. Together with future Doctors David Vint and Davide Manna, they were my gang in the lab, and made my experience in Glasgow a memory I will forever cherish.

Finally, deepest appreciation also to my family, who shared this journey with me and endured me through highs and lows. Them, and all those friends, new and old, who were there for me through these years, have really made a difference.

This research was funded by AFOSR under grant number FA8655-20-1-7037.

Abstract

Machine learning and Artificial intelligence have already revolutionised the world we live in. Nevertheless, these technologies are expected to progress even further and advance living standards far beyond today's reality. To achieve this revolution, several limitations of current AI need to be addressed. Demands on computing resources and energy supply are a major obstacle, which ultimately limit the capabilities of the AI we can deploy at the edge. Furthermore, current systems are ill-suited for continual learning (CL) on real world data, as they require to train with all available data as an independent and identically distributed (i.i.d.) set. This thesis focuses on these problems by working on visual feature extraction, and contributing to more efficient and accurate algorithms, with the capacity for CL.

With the objective of developing energy efficient feature extraction, a major part of the thesis is focused on Spiking Neural Networks (SNNs). SNNs have become an interesting alternative to conventional artificial neural networks (ANN) thanks to their temporal processing capabilities and energy efficient implementations in neuromorphic hardware. However, the challenges involved in training them have limited their performance in terms of accuracy and thus their applications. Improving learning algorithms and neural architectures for a more accurate feature extraction is therefore a priority. Contributing towards this aim, this work presents a study on the key components of modern spiking architectures, an in-depth study on the possible implementations of spiking residual connections, and a novel spiking version of the successful residual network architecture. Additionally, the effect of different state of the art techniques are empirically compared in image classification tasks to provide a state of the art guide to SNN design. Finally, the proposed network outperforms previous SNN architectures in multiple datasets, while using less parameters.

In order to exploit SNNs for more efficient AI, it is also of interest to understand the full scope of their exploitable properties. These networks are characterised by their unique temporal dynamics, but the properties and advantages of such computations are still not fully understood. In order to provide answers, in this work it is demonstrated how spiking neurons can enable temporal feature extraction in feed-forward neural networks without the need for recurrent synapses, and how recurrent SNNs can achieve comparable results to LSTM with a smaller number of parameters. This shows how their bio-inspired computing principles can be successfully exploited beyond energy efficiency gains, and evidences their differences with respect to conventional artificial neural networks. These results are obtained through a new task, DVS-Gesture-Chain (DVS-GC), which allows, for the first time, to evaluate the perception of temporal dependencies in a real event-based action recognition dataset. Furthermore, this setup allows to reveal the role of the leakage rate in spiking neurons for temporal processing tasks and demonstrated the benefits of "hard reset" mechanisms.

Finally, the focus is switched to the capacity for training feature extractors in continual learning scenarios, a major milestone for the development of truly autonomous systems and artificial general intelligence. The challenge in this setup is to avoid catastrophic forgetting, where artificial systems forget previous knowledge if they are trained in new data without revisiting the old. Often, the methods used to alleviate forgetting make use of either rehearsal buffers, pre-trained backbones or external indication of the task to solve. However, these requirements result in severe limitations regarding scalability, privacy preservation and efficient deployment. This work explores how to eliminate the need for such requirements and proposes a new method, Low Interference Feature Extraction Sub-networks (LIFES). Additionally, the study breaks down the Catastrophic Forgetting (CF) problem into 4 causes, allowing to better understand the effect of CL methods. The proposed LIFES algorithm achieves competitive results in standard incremental learning scenarios, providing an alternative to approaches with more restrictive requirements. Moreover, it provides solutions for specific causes of the CF problem, making it complementary to other methods.

Contents

Chapter 1

Introduction

The major factor for human progress and the advancement of living standards, has historically been technological breakthrough [1, 2]. While social, political and environmental factors mediate this progress, its ultimate potential is defined by the limits of what humans can build. In the present day, the advent of a new era of growth is becoming increasingly apparent, where machine intelligence will enable an exponential increase of human capabilities. The improvement of artificial intelligence (AI) will keep pushing forward the level of automation in production processes and daily tasks, while allowing to develop superhuman intelligence: first in specialised tasks [3, 4] and ultimately in the form of artificial general intelligence (AGI) [5, 6].

To achieve this promised revolution, several limitations of modern AI systems need to be addressed. Compared to biological brains, Artificial Neural Networks (ANN) lack behind in robustness, generalisation capabilities, energy efficiency, and capacity for online continuous learning, among others [7–9]. Specifically, when focusing on the needs for future autonomous systems, demands on computing resources and energy supply are a major obstacle, which ultimately limit the capabilities of the AI we can deploy at the edge. Furthermore, current systems are ill-suited for continual learning on real world data, as they require to train with all available data as an independent and identically distributed (i.i.d.) set, hampering online adaptation to new environments and making the scaling to life-long learning setups unfeasible in terms of data storage [10].

Research towards this future AI encounters a unique situation which is not common in other engineering disciplines. The realisation of the desired systems already exists: biological brains implement intelligence in an efficient and sustainable way, demanding low energy consumption and allowing continuous learning from new environments, while demonstrating robustness and consistency in their performance.

Consequently, neuroscience has historically served as reference in AI development [11], prompting the invention of many modern ANN components, sometimes in the form of strict design guidelines and, more often, in a looser sense as source of inspiration. Still, despite its logical nature, this research perspective is not free from controversy. A part of the community has dedicated efforts to maximizing the similarities between AI and real brains, in an attempt to decipher the ways in which biology implements those properties of intelligence which AI is lacking. On the contrary, dissenting scholars argue that neuroscience might serve as inspiration, but the evolution of AI is likely to diverge from the one of organic intelligence, as it is not bounded to the same constraints, and therefore imitating it imposes limits which hinder development [12].

Often, advocates for the divergence of neuroscience and AI recall the "biological vs. mechanical flight" argument, where aeroplanes are supposed to achieve the same purpose as birds, flight, but through different means: instead of imitating flapping wings, the human-engineered solution defied gravity by means of fixed wings and propulsion engines. Still, this argument is an oversimplification, as the two solutions to flight achieve very different purposes: birds are more nimble and more energy efficient than a plane, while planes can cover large distances at high speed while carrying heavy weight loads. This same logic applies to biological inspiration in AI, for use cases where the system is not subjected to the same constraints as biological ones, development should be free to diverge from organic intelligence. In contrast, autonomous systems with limited energy supply and requiring fast responses and online adaptation, are likely to benefit from neuroscience insights. These systems intersect with biology in multiple requirements, while diverging in others due to their implementation in silicon, access to batteries, communication networks and more. Therefore, optimal solutions are likely to combine bio-inspired principles with "silicon-specific" ones.

In this thesis, the approach of brain inspiration is taken in order to develop more efficient and adaptable machine learning. This "inspiration" label encompasses many levels of neuroscience influence. In the case of this work, the objective is to identify benefits in neuroscience insights and port them to AI, while hard "biological plausibility" constraints are avoided. This means that the objective is not to replicate biology, but to distill useful principles.

Specifically, the work focuses on visual feature extraction with ANNs. Computer vision applications are typically built on top of the best performing feature extractors, therefore, advancements on this core component are key, as they define the performance ceiling of all the rest. Examples of such applications include object detection [13], semantic segmentation [14], action recognition [15], image and video generation [16] among many others.

From an energy efficiency perspective, an increasingly popular approach has been the use of Spiking Neural Networks (SNN). These are a kind of ANN which closely replicate the workings of real neurons in the brain, porting their benefits to the artificial domain. The first two contribution chapters of this thesis (4 and 5) focus on these networks.

SNN have demonstrated benefits in energy efficiency when implemented in specialised neuromorphic hardware, but their limitation has often been the difficulty added in training them and their lower accuracy compared to conventional ANNs. In order to contribute to their usability as feature extractors, the work presented in Chapter 4 studies the key strategies for the development of the SNN architectures, including a novel study on residual connections, and then uses them to propose a novel network, the Spiking ResNet (S-ResNet), which achieved state of the art results at the time of publication. After that, Chapter 5 follows by studying the unexplored advantages of SNN beyond energy efficiency. Specifically, this piece of the work demonstrates how the bio-inspired computing paradigm of SNNs can lead to advantages when processing temporal data, proposing a novel task to evaluate them and studying their capacity for spatio-temoporal feature extraction.

Finally, Chapter 6 switches the focus to the limitations of ANNs for continual learning. Learning dynamically from natural non-i.i.d. distributions of data is major need for machine learning, but avoiding catastrophic forgetting when doing so has been a long-standing challenge [10, 17, 18]. Multiple solutions have been proposed to reduce this undesired forgetting, but the most successful approaches have requirements that impose severe limitations regarding scalability, privacy preservation, and efficient deployment. The most common being the storage of old data or the indication of which task is the system solving in a multi-task scenario. These requirements are not found in real brains; therefore, the work in this chapter proposes a system which does not use them, bypassing them by creating specialised sub-networks that collaborate within the ANN. At a high level, this sub-networks perspective can be considered biologically inspired, as it is observed in neuroscience [19]. The resulting study breaks down catastrophic forgetting into 4 distinct causes, and proposes solutions to mitigate 3 of them. Moreover it clearly delineates the remaining challenges in catastrophic forgetting prevention, paving the way for more scalable and efficient continual learning.

1.1 Summary of Original Contributions

This thesis presents the following contributions to the machine learning field:

1. **A novel study on spiking residual connections:** In certain configurations, spiking neurons can diminish the benefits of residual connections, a kind of connection that has been key for the development of deep architectures [20]. The proposed study analyses three different ways of implementing residual connections for SNNs, highlighting their properties in terms of accuracy, network activity, characteristics of their derivatives, and implications in hardware requirements. Two of these implementations already existed [21, 22], where one of them was modified with respect to the version found in previous work. The third one was proposed in this work. This analysis demonstrates benefits and drawbacks of each kind of connection, regarding, and allows to make the optimal choice depending on application requirements.

2. **A new SNN architecture, the S-ResNet:** Through empirical experimentation, testing multiple strategies, the configuration of a spiking residual network was optimised. The resulting network is obtained by combining the conclusions extracted from the study on residual connections with a combination of the best performing methods from literature. At the time of publication [23], the results demonstrated state of the art accuracy in multiple visual datasets with a smaller number of parameters than previous approaches.

3. **A new task for event-based action recognition:** Rigorous evaluation of spatio-temporal feature extraction in event-based data is challenging due to the lack of suitable datasets. To solve this, a new task, DVS-Gesture-Chains (DVS-GC), is proposed. Unlike previous tasks, DVS-GC explicitly evaluates the perception of temporal order in chains of actions, which were created as chains of human gestures recorded with a neuromorphic camera.

4. **A novel study on the exploitable properties of spiking neurons for spatio-temporal feature extraction:** By means of the new DVS-GC task, the capacity of SNNs for spatio-temporal processing are demonstrated, showing how, unlike conventional ANNs, they enable temporal feature extraction without recurrent synapses and how their computations are comparable to those of long short-term memory (LSTM) cells, but requiring less synaptic weights. This demonstrates how the use of spiking neurons

enables temporal feature extraction at no extra cost in computation; how SNNs can serve as alternative to more complex memory cells when computational requirements are a constraint; and how feed-forward networks can be enabled to perform temporal feature extraction. Additionally, these comparisons highlight similarities and differences between ANN and SNN computations, facilitating the combination of principles of one into the other in future research.

5. **A new definition of catastrophic forgetting:** Given a classification task, the causes of catastrophic forgetting are broken down into 4 distinct mechanisms. Mathematical formulation is provided, formalising the problem in a way that allows to evaluate these mechanisms in isolation, and understand which of these forgetting mechanisms are addressed by continual learning methods. Such factorisation of the problem had not yet been formalised in literature, therefore, doing so provides a guide for further research, as it highlights the specific mechanisms which need addressing in order to prevent forgetting.

6. **A new continual learning method:** Given the limitations imposed by the requirements of existing continual learning methods, a new approach is proposed (Low Interference Feature Extraction Sub-networks) which bypasses them. The effects of catastrophic forgetting in this system are studied and 3 novel components are proposed to alleviate its effects: Regularisation with lateral classifiers, weight standarisation, and sub-network interference connection pruning. The resulting system is compared to multiple state of the art approaches and demonstrates competitive results with minimal requirements. Additionally, it can be combined with other approaches, as explained in Chapter 6, contributing to the development of more scalable approaches to continual learning.

1.2 Thesis organisation

The remainder of this thesis is organised as follows. Chapter 2 provides a literature review on feature extraction in conventional ANNs and SNNs. Chapter 3 reviews the state of the art for continual learning and presents contribution 5, as it is used to better describe existing CL methods. Chapter 4 presents contributions 1 and 2 towards improved feature extraction with SNNs. Chapter 5 presents contributions 3 and 4, analysing the exploitable properties of SNN for temporal tasks. Then, Chapter 6 presents the novel CL algorithm described in contribution 6. Finally, Chapter 7 closes the thesis with conclusions and directions for future work.

Chapter 2

Feature extraction and Spiking Neural Networks

2.1 Neural Networks and data-driven feature extraction

In recent years, the field of machine learning has become almost completely dominated by the models known as Artificial Neural Networks, even coining a new term, deep learning, for machine learning performed with deep neural networks [24]. These ANN systems define simple computing units, the neurons, and create networks by linking them with weighted connections. Their power lies in their capacity to learn from large datasets, potentially approximating any mathematical function [25] and thereby creating any necessary input to output relationship.

The pivotal moment marking the advent of ANNs as the dominant approach is often attributed to the success of the AlexNet network [26] which won the ImageNet Large Scale Visual Recognition Challenge [27] in 2012. Prior to this, the state of the art was often defined by handcrafted feature extraction, which was then combined with approaches such as Support Vector Machines for classification [28]. In contrast, ANNs are data driven approaches, which do not require expert knowledge to define the features to extract, instead; they learn features directly from the data. This is most often done by defining a task that involves mapping a specific input to a known output (supervised learning), and then optimizing the network configuration to minimise the error in this mapping for the given training dataset [24]. With few exceptions, this minimisation is done by means of the Backpropagation of error algorithm [29]. This algorithm formulates the output error with a differentiable loss function and then calculates the gradient of the loss with respect to the network weights by applying the chain rule, therefore propagating the error backward from the output layer to the input layer. Then, by means of gradient descent, the weights are adjusted iteratively, using a fraction of the data (mini-batch) in each descent step. Like this, the value of the loss function is minimised thereby improving the accuracy of the model.

2.1.1 Visual feature extraction

Convolutional Neural Networks

This work is focused on visual feature extraction. For this data modality, arguably the biggest cornerstone was the development of Convolutional Neural Networks (CNNs) [30]. Visual data is characterised by its high dimensionality, high spatial redundancy and its correlation to the physical world. The visual representation of a scene changes if objects move, but machine intelligence needs to understand that this change is just in location, but not in content. CNNs made this understanding easier for deep learning by defining local connectivity patterns for the network weights, which enable translation invariant feature extraction.

Specifically, CNNs are designed to process data with a grid-like topology (such as images). Their core building block is the convolutional layer, which consists of a set of learnable filters (or kernels) that are convolved with the input data. Each kernel K is a two dimensional array of dimension $D_{k1} \times D_{k2}$ and it is applied across the entire input image to produce a feature map, hence the translation invariance of the process, as the same set of weights will be applied to all spatial locations.

The convolution operation for a single filter K over a 2D input I can be described by:

$$(I * K)(x, y) = \sum_{m}^{D_{k1}} \sum_{n}^{D_{k2}} I(x+m, y+n) \cdot K(m, n) \tag{2.1}$$

Where $I(x, y)$ is the pixel value at position (x, y) of the input and $K(m, n)$ is the kernel value at position (m, n).

Then, a convolutional layer is defined for an input of $S_h \times S_w \times N$ dimensions, where N corresponds to the number of channels or input dimensions (e.g., 3 for an RGB image). The output O_k dimensions will be $S'_h \times S'_w \times M$, where M is the number of kernels (filters) used in the convolutional layer. Each of the M output feature maps is generated by convolving each of the N input channels with a corresponding set of M kernels and then summing their outputs:

$$O_k(x,y) = \sum_{c=1}^{N} \sum_{m}^{D_{k1}} \sum_{n}^{D_{k2}} \mathbf{I}_c(x+m, y+n) \cdot K_i(m,n) \tag{2.2}$$

The dimensions of the output feature map can be computed based on the input dimensions, the kernel size, stride s, and padding p. For an input of size $S_h \times S_w$ with a kernel size of $D_{k1} \times D_{k2}$, the output size $S'_h \times S'_w$ is given by:

$$S'_h = \left\lfloor \frac{S_h - D_{k1} + 2p}{s} \right\rfloor + 1 \tag{2.3}$$

$$S'_w = \left\lfloor \frac{S_w - D_{k2} + 2p}{s} \right\rfloor + 1 \tag{2.4}$$

Here, $\lfloor \cdot \rfloor$ denotes the floor function, which ensures the output size is an integer.

These layers are then stacked hierarchically in a feed-forward manner [30], as done for fully connected layers or any other variant. During training, each kernel k_i learns how to detect a distinct feature, and the number of channels in the layer are referred to as the layer width, which defines the number of the feature maps generated and the cost of the layer in parameters.

Spatio-temporal feature extraction

While convolutional layers allow for spatial feature extraction in visual data, they do not address the extraction of temporal features, which is necessary when the input changes over time such as in the case of video.

Extracting temporal features means combining information from different moments in time in order to extract meaning. For this, a property of cognitive systems that is considered essential is working memory, which holds information from previous events and allows to relate it to those perceived later [31, 32]. In the field of neural network engineering, working memory has historically been implemented by recurrent connections [33], and their memory capabilities have been further enhanced by the use of advanced memory cells such as the widespread Long Short-Term Memory (LSTM) [34]. Specifically, LSTM (Fig. 2.1) defines an internal state C_t for each neuron which retains old information, called the cell state, and controls the forgetting or retention of this information by means of three gates, the Forget gate (f_t), Input gate(i_t), and Output gate(o_t), which are implemented by an additive attention computation with Sigmoid activation σ. This additive attention is controlled by a trainable layer, therefore, the network can learn to interpret when to forget old information depending on the input and its internal state.

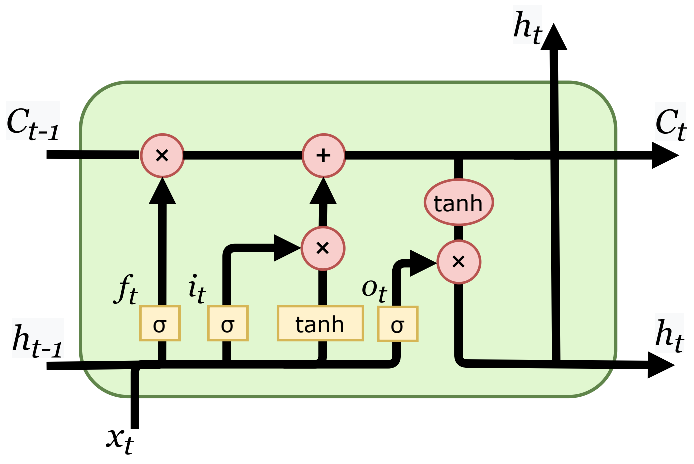

Fig. 2.1: Diagram of an LSTM memory cell. Yellow boxes indicate trainable layers, red circles indicate operators. C_t is the cell state, h_t the hidden state and output, the yellow *tanh* is a layer of synaptic weights with Hyperbolic Tangent activation. σ stands for the gating layer with Sigmoid activation.

More recently, temporal processing tasks have also been solved by the increasingly popular Transformer architectures [35] (further explained in the following subsection). When using these networks, temporal events are not presented in a succession as they happen; instead, multiple time-steps are accumulated (or the whole sequence in many cases) and then processed offline by the system. These approaches can be considered to implement working memory outside of the neural network, by accumulating stimuli over time and then feeding them to the network together as a single input. Transformers have achieved state of the art accuracy in the majority of temporal tasks [36–38], but their computational and memory complexity scales as $O(L^2)$ with the sequence length. Hence, research in recurrent architectures is still of interest in order to create lighter systems with dynamic memory management.

State of the art architectures

Since the advent of ANNs for visual feature extraction, a large body of work has focused on improving their architectures, giving rise to a plethora of methods which kept building on top of each other further enhancing performance.

For computer vision, the AlexNet breakthrough set the precedent for the use of feed-forward CNNs when performing feature extraction on images. Later improvements came with networks such as the widespread VGG [39], which achieved first and the second places in the localisation and classification tracks of the 2014 ImageNet challenge. These accuracy improvement was obtained by empirically finding a better dimensionality configuration for convolutional layers and creating deeper networks, up to 19 layers.

In 2015, a significant advancement in the performance of feed-forward ANNs was achieved through Batch Normalization (BN) [40]. This module eases training by reducing the internal covariance shift, where the distribution of each layer's inputs changes during training, making it harder to train effectively.

In order to do so, BN normalizes the output of a previous activation layer by subtracting the batch mean and dividing by the batch standard deviation, and then scales and shifts the normalized output using learned parameters. This normalization is performed for each mini-batch during training. Given the mean μ_k and standard deviation σ_k across the batch and the learnable weights γ_k and β_k, for an input of d dimensions $x_t = (x_{1,t}...x_{d,t})$, the method normalises each feature k (or convolutional channel in the case of CNNs) independently:

$$BN(x_k) = \gamma_k \frac{x_k - \mu_k}{\sqrt{(\sigma_k)^2 + \epsilon}} + \beta_k \tag{2.5}$$

In 2016, after the success of the first CNNs such as AlexNet and VGG [39], the next big improvement came with the addition of residual connections. As demonstrated in [20] with their ResNet architecture, residual connections allow to successfully train much deeper architectures, up to 152 in their paper, giving rise to a more accurate and efficient family of networks. They achieve this by adding the original input x to the output of each layer block $F(x)$:

$$H(x) = F(x) + x \tag{2.6}$$

This reformulates the problem to learning residual mappings that modify the input x. Doing so makes it easier for the learning algorithm to build identity mappings, where the input to a layer block is equal to the output $H(x) = x$, as it can be accomplished just by setting the weights in the layer to zero $(F(x) = 0)$. This allows the network to easily ignore unnecessary layers, therefore avoiding accuracy degradation when the architecture becomes many layers deep. Alternatively, when the optimal solution is not an identity mapping it might still be closer to it than to a zero mapping, making for a better initialization [20].

Later work [41] has proven that, in order for residual networks to be effective, either Batch Normalization (BN) [40] or alternative strategies which replicate its benefits (Weight Standardization [42]) are necessary. This is because BN makes the contribution of the residual path increasingly smaller through depth compared to the skip path at initialisation. This makes the network effectively shallow at earlier training stages, and allows it to give deeper layers increasingly more influence by increasing the variance in the weights as training goes on. Additionally, it also neutralizes the mean shift activation caused by the Rectified Linear Unit (ReLU) activation [43], increases the maximum affordable learning rate and acts as an implicit regulariser.

After residual architectures, the next breakthrough came through attention computations, which enabled improved accuracy for CNNs [44] in the form of additive attention [45]. The self-attention blocks defined in [44] are still used today in the EfficientNet family of networks [46], which are the best performing CNNs to date, thanks to their optimised architecture which was found by means of a reinforcement learning approach for neural architecture search [47]. In parallel, attention also enabled the development of Transformers [35], which use multi-head dot product attention layers interleaved with fully connected ones.

The essence of an attention computation is simply to define multiplicative relationships between neuron outputs instead of the more common additive ones. In most implementations , one of these values is constrained to a $[0, 1]$ range by means of a sigmoid [44, 45] or softmax function [35], acting as attention coefficient, which modulates the output of a layer. These coefficients are obtained through a fully connected layer in additive attention, while dot product attention obtains them as the dot product between two feature vectors.

Regarding Transformer architectures, in the present day they have set the state of the art results for most large datasets, and have become the standard approach for massive computation applications. Still, their limitation, as previously mentioned, lies in their computational requirements, as they require inputs to be broken down in a sequence of ”pieces” (time frames for temporal data or image blocks for visual data), then their computational and memory complexity scales as $O(L^2)$ with the sequence length.

2.2 Spiking Neural Networks

First coined in Carver Mead's 1989 publication *Analog VLSI and neural systems* [48], the term neuromorphic engineering has become synonym of the intersection between AI and neuroscience. Originally, the concept was introduced as "the use of very-large-scale integration (VLSI) systems containing electronic analog circuits to mimic neuro-biological architectures present in the nervous system". In recent years, neuromorphic engineering has brought together computational neuroscientists and researchers on deep learning and computing hardware, in a field that encompasses research trying to understand the brain by means of AI, and research applying neuroscience insights to improve AI [8, 49].

The common denominator in this field is the use of Spiking Neural Networks, a kind of ANN that closely replicate the inner workings of real neurons [50]. From an engineering perspective, SNNs' sparse and asynchronous computations have been demonstrated great gains in energy efficiency when implemented in specialised neuromorphic hardware [51], which, unlike traditional von Neumann architectures, can exploit activation sparsity, only transmitting signals coming from active neurons. Serving as example, [52] demonstrate how the same network consumes 11 times less energy in a neuromorphic implementation than its ANN counterpart, while [51] finds a reduction of up to 100 times less.

2.2.1 Spiking Neurons

The defining factor of SNNs, which makes them differ from traditional ANNs, is their neuron model, the spiking neuron, which replicates the behaviour of real neurons. In short, biological neurons are characterised by possessing an electrically charged cell membrane that accumulates voltage received from other neurons in the network. When this voltage surpasses a set threshold, ion channels open, further increasing the neuron's voltage and triggering the emission of an action potential or "spike". This spike is sent through the synapses which connect to other neurons in the network, modifying their membrane potentials and thereby communicating information.

Simulation of this behaviour can be achieved at multiple degrees of biological realism. Hence, multiple neuron models with different levels of complexity have been proposed [53]. This research uses the Leaky Integrate-and-Fire (LIF) model [54]. Despite their simplicity, LIF neurons found great success in many state of the art systems [22, 55, 56], while the use of more complex versions incurs in a higher computational cost with no clear functional benefits.

The LIF model can be formulated as the differential equation seen in (2.7), where $U(t)$ is the membrane potential, U_{rest} the resting potential, τ is the time constant and $I(t)$ is the input current. When the voltage $U(t)$ surpasses a set threshold U_{th}, the neuron emits a spike and the potential is reset by subtraction.

$$\tau \frac{du}{dt} = -(U(t) - U_{rest}) + RI(t) \tag{2.7}$$

In order to easily program this behaviour in machine learning models, explicit iterative versions of this differential equation are used. Let i be a post-synaptic neuron, $u_{i,t}$ is its membrane potential, $o_{i,t}$ its spiking activation and λ the leak factor. The index j belongs to the pre-synaptic neuron and the weights $w_{i,j}$ dictate the value of the synapses between neurons (Fig. 2.2). Then, the iterative update of the neuron activation is calculated as follows:

$$o_{i,t} = g\left(\sum_j (w_{ij} o_{j,t}) \; + \lambda \cdot u_{i,t-1}\right) \tag{2.8}$$

where $g(x)$ is the thresholding function, which converts voltage to spikes:

$$g(x) = \begin{cases} 1, \text{ if } x \geq U_{th} \\ 0, \text{ if } x < U_{th} \end{cases} \tag{2.9}$$

After spiking, a reset is performed by the subtraction $u^*_{i,t} = u_{i,t} - U_{th}$, where $u^*_{i,t}$ is the membrane potential after resetting.

Notice that when this neuron model is used without the leakage factor, it is referred to as an Integrate-and-Fire (IF) model. This version will also be used for this research.

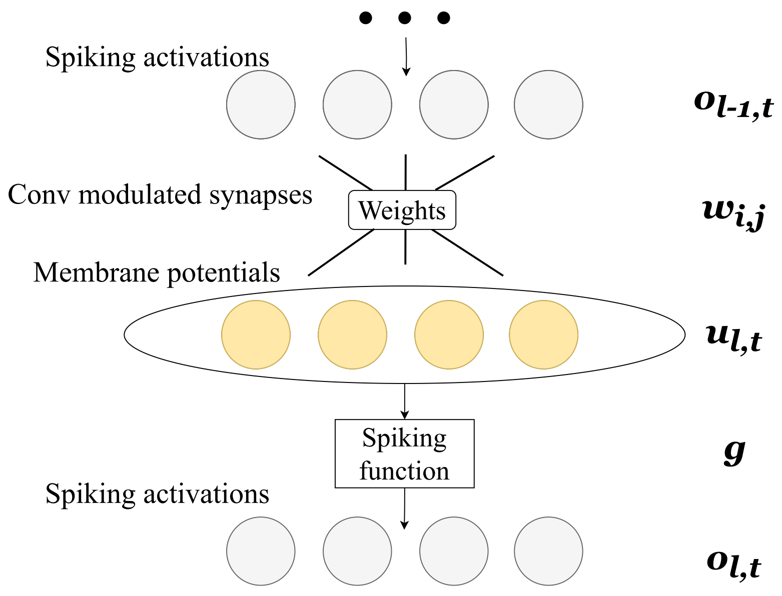

Fig. 2.2: Diagram of the operations performed in a spiking layer.

2.2.2 Spiking Neural Network architectures

SNN architectures have been developed for multiple AI tasks, including reinforcement learning [57], control systems [58], time series processing [59], and even text generation [60]. Still, the most prevalent use case has been computer vision. In this domain, the mainstream tasks used to develop and evaluate new feature extractors are image classification and action recognition. The objective of these tasks is simply to assign the correct classification label to each data sample, images for the former and video for the latter. The extracted features will create data representations that allow to discriminate between different classes, therefore higher classification accuracy will be linked to a feature extraction process that better represents the properties of the data classes. Hence, these tasks are considered a useful evaluation metric for feature extraction quality and will be used in this thesis as benchmarking tool.

Image classification

SNN architectures for image classification have followed a similar path as conventional ANNs; still, their alternative computing paradigm has posed a challenge when porting certain deep learning structures to spiking format, making them lag behind. On the other hand, these differences have also motivated the development of alternative solutions, usually more biologically plausible.

Up to 2021, the highest SNN image classification accuracies were reported with networks basing their topology on VGG [61–63]. This was achieved by reusing the original network topology and substituting the conventional neurons and their ReLU activation with IF or LIF spiking neurons.

The first attempts to replicate ResNet architectures in SNNs, are attributed to Lee et al. [21] and Zheng et al. [55], which implemented the first trainable spiking ResNets. These networks achieved competitive results, but were still outperformed by non-residual approaches in some scenarios. In 2021, Spiking ResNets achieved state of the art results through the work presented by Fang et al. [22], and the one presented by the first contribution of this thesis [23] (Chapter 4). After that contribution, in 2023, further accuracy improvement has been reported in literature by implementing Visual Transformers [64] in spiking format [56]. Performance comparisons can be found in Section 2.2.4

Action recognition and temporal processing

The state of the art for SNNs in temporal tasks has been historically based on Recurrent Neural Network (RNN) architectures, which implement recurrent connections that create a path between a layer's output and its input. The authors in [65] proposed Recurrent SNNs (RSNNs) of Leaky integrate-and-fire (LIF) neurons with neuronal adaptation, a process that reduces the excitability of neurons based on preceding firing activity. Their resulting network is tested in the Sequential MNIST (S-MNIST) [66] and TIMIT tasks [67]. Subsequent work applied LSTM cells to SNN networks, achieving higher performance in S-MNIST [68].

Still, for the processing of visual event-based datasets such as DHP19 [69] or DVS-Gesture [70], the state of the art has been set by feed-forward SNNs with no recurrency [22, 55, 71]. The question is then whether these feed-forward SNNs implement working memory or, on the contrary, the aforementioned tasks do not require a network with temporal feature extraction. Chapter 5 answers this question, proving how both statements are true.

Finally, with the recent efforts on developing Transformer architectures in spiking format, since 2023 some examples have appeared where spiking Transformers achieve state of the art results in temporal tasks. The authors in [56] obtain comparable performances to the best systems on DVS-Gestures by means of a spiking Transformer, while in [60], the "SpikeGPT" network recorded the best SNN performance for language generation. Given that SNNs are usually employed to achieve computational efficiency, the future adoption of these SNN Transformer systems is not yet guaranteed due to their higher computational cost. The next few years will define whether these research direction meets the desired computational limits in target applications.

2.2.3 Training methods for spiking neural networks

The binary and sparse nature of SNN gives great computational advantages, but at the same time these properties create some challenges at training time. That is why the development of learning algorithms for these networks is a very active field.

Conventional non-spiking neural networks owe most of their success to the backpropagation (BP) of error algorithm [24], where the optimal synaptic weights are found by minimising an error function with respect to these same weights.

In order to apply the BP algorithm, the whole network needs to be expressed as a differentiable function. In the case of SNNs, the spiking behaviour inside the neurons creates a non-differentiable function, therefore learning through BP requires additional workaround strategies. Moreover another difference between SNNs and regular Deep Learning is the fact that SNNs use the time dimension for their computations, therefore the neuronal states also have time dependencies. In the case of non-spiking ANNs, the time dimension is used when processing temporal streams, but it is not used for time invariant tasks such as image classification or object detection.

Given these properties and their challenges, several approaches to SNN training have been proposed, each with their own advantages and disadvantages. In computer vision, there are three main categories: direct supervised training, conversion methods and unsupervised training.

Direct supervised training

Directly training the SNN allows one to exploit all its valuable properties, including its sparsity and its capacity to process asynchronous inputs. However, the challenge then becomes to successfully train it given that gradient descent based methods cannot be applied to non-differentiable spiking functions. The most common strategy in state of the art methods is the use of surrogate gradients [63, 72], a method where the spiking function is used in the forward path, but when calculating its derivative in the backwards path, a continuous tractable function is used, which tries to approximate the behaviour of the real derivative.

Another option is to use a version of the SNN model that is directly differentiable. Some examples can be found in [73]. We can find models using soft non-linearities [74], probabilistic models [75] or latency-based networks [76].

Alternatively, supervised learning can also be performed without the differentiation of the whole network. Some examples use local approaches with algorithms such as [77], where the loss is computed locally in each neuron, or by using three factor learning rules [78].

When talking about final task accuracy, surrogate gradient BP is the best performing method so far. All the best SNN feature extractors consistently use this method [55, 56, 79], but the BP implementations and the surrogate functions they use vary between them.

Concerning the BP implementation, different variations can be found among the best performing networks. Most works choose to use Backpropagation Through Time (BPTT) [80], the mainstream algorithm also used for the training of RNNs and other time-dependent networks. BPTT "unrolls" the network in time, meaning that, given a fixed number of time-steps for which training will be performed, a copy of the network is created for each of them and the temporal dependencies between time-steps are explicitly defined. Considering an integrate and fire model, as previously defined, where the output $o_{l,t}$ of a layer l depends on the output of a previous layer $o_{l-1,t}$ and its own membrane state $u_{l,t-1}$ in the previous time-step $t-1$, then the derivative of the loss function E can be unrolled as follows:

$$\frac{\partial E}{\partial W} = \frac{\partial E}{\partial o_{l,t}} \frac{\partial o_{l,t}}{\partial u_{l,t}} \frac{\partial u_{l,t}}{\partial W} \tag{2.10}$$

where the gradient $\frac{\partial u_{l,t}}{\partial W}$ depends on previous states which are computed recursively:

$$\frac{\partial u_{l,t}}{\partial W} = \frac{\partial u_{l,t}}{\partial u_{l,t-1}} \frac{\partial u_{l,t-1}}{\partial W} + \frac{\partial u_{l,t}}{\partial o_{l-1,t}} \frac{\partial o_{l-1,t}}{\partial W} \tag{2.11}$$

This expression shows how computing this gradient requires to access the state of the network at all time-steps t, hence its cost in memory requirement.

Alternatively, a slightly different implementation is found in [21], where the authors use a Spike-based BP algorithm which proposes a novel way of accounting for the leak factor of Leaky Integrate-and-Fire (LIF) neurons. Finally, there are also BP approaches where the input spikes are convolved with spike response kernels like in [81], which allows for convenient spike response implementations at the cost of saving more spike time-stamps in memory.

For the choice of surrogate functions, there is no consensus on a better choice. The derivatives of any continuous function that approximates a step function can be a valid surrogate. Some examples are triangle shape surrogates in [63], rectangular shaped in [55], and arc-tangent shaped in [22, 79].

The main disadvantage of BPTT is its computational cost, as with its "network unrolling" it requires to explicitly define all temporal dependencies and store the neuron states at every point in time. In order to make the process more efficient, approaches such as e-prop [82] or Eventprop [83] have been proposed, which approximate the optimal credit assignment obtained by BPTT without storing all activation history. This comes at the cost of less precise updates and therefore, for most cases, lower final accuracy.

Conversion methods

In most machine learning tasks, the highest accuracy figures are reported by non-spiking deep learning networks which outperform their spiking counterparts.

Therefore, conversion methods propose to train a non-spiking neural network and then convert it to spiking format to make them suitable for implementation in neuromorphic hardware and reduce energy consumption.

There are different methods that allow such a conversion. These techniques create an approximation of the original network that is accurate up to a certain approximation error, and with a certain cost in terms of floating point operations per second (FLOPS) and latency. This approximation needs to reconstruct each non-spiking neurons in the original network through a set spiking neurons, and therefore the key challenge is to represent continuous activation values using the binary outputs of spiking neurons.

Comparing to direct training of an SNN, the spiking system resulting from the conversion has higher latency and energy consumption. As an example, the authors in [21] demonstrated how high performance was achieved with 30 time-steps for directly trained networks, while conversion based ones needed more than 100. Another disadvantage is that one cannot perform online training or low power training directly in a neuromorphic device, as the network can only be deployed onto the neuromorphic device once it is trained and converted. On-chip training is a topic of interest in current research [84, 85], which is key for applications requiring further learning after deployment.

Finally, also the temporal resolution of the system is affected, as the training is done with discrete time frames instead of a flow of input spikes. This is likely to cause under-performance in neuromorphic datasets as proved by [86].

Nonetheless, the advantage of using conversion techniques is the possibility to approximate the accuracy performance of non-spiking deep learning networks. If the non-spiking ANN can perform the task, converting it to SNN provides guarantees of success without having to manage the more complex SNN training.

Rate based methods – This is the most common conversion approach [87, 88], where the rate of spiking mimics the activation value that the original non-spiking neuron would have had. This approach can only be used to convert networks which employ the ReLu activation function, and the conversion is performed just by switching the ReLU activation function for the spiking function. The weights and connectivity are kept the same. Then, the input is fed to the network for N time-steps. The firing rate of each neuron becomes an approximation of the original activation value if enough time steps are allowed. In a feedforward architecture, the output approximation of a given layer is affected by its presynaptic neurons, therefore until the previous layer converges, the next one will not be able to do so; and then, the deeper the architecture, the more time steps the system needs and the more approximation error is accumulated.

Rueckauer et al. [89] show how a converted SNN with reset by subtraction (the alternative to a hard reset to 0) can approximate the original i^{th} activation a_i^1 with an additive error, which is time dependent and proportional to the threshold, as in (2.12).

$$r_i^1(t) = a_i^1 r_{max} - \frac{V_i^1(t)}{t \cdot V_{thr}} \tag{2.12}$$

Instead, Kugele et al. [90] propose to perform conversion by first constraining the original ANN to work in a streaming rollout fashion [91]. Typically RNNs are considered to to have a time delay of one time step in their recurrent connections, meaning that the recurrent output will come to the neuron's input in the next time step. At the same time, the feedforward connections are considered to be instantaneous without any time delay. This model is called a sequential rollout. Alternatively, in a streaming rollout, feedforward and skip connections are also considered to take time when propagating information. This work is able to mimic the time based dynamics of SNN in the training of the original ANN. This, together with DenseNet as architecture of choice, allows to process neuromorphic sequences, give early approximates, and to obtain one of the most accurate systems for the DVS gestures dataset.

Time based methods – Stockl et al. [92] propose an alternative to the typical rate based conversion with the "At Most One Spike per neuron" (AMOS) method. Rate based methods perform a one to one conversion, where each original neuron is approximated by one spiking neuron. Alternatively, AMOS approximates one neuron with a circuit of spiking neurons. By summing the contribution of several neurons, this circuit is able to approximate the original continuous activation value using the binary outputs of spiking neurons.

In contrast with rate based conversions, where only the rectified linear unit activation function can be approximated, AMOS can work with any activation function, as the parameters of the circuit can be optimised to approximate any function. This is why the authors are able to convert the best performing CNN architecture to the spiking domain, EfficientNet [46], and beat the 2019 state of the art in image classification. Moreover this method has a higher throughput than rate methods, as each neuron spikes just once per input, and therefore a new input can be fed after each timestep. On the other hand, the resulting converted network has a higher complexity and memory cost compared to the original one or rate based conversions.

Unsupervised training

Thanks to their biologically plausible computations, SNNs can implement Spike-timing-dependent plasticity (STDP), a learning algorithm observed in the brain, that uses spike timings to modulate the weight of the synapses between neurons. This method is the standard for unsupervised training in SNN [93] and, for shallow architectures, it can compete in terms of accuracy with self-supervised methods such as auto-encoders in non-spiking deep learning [94].

The STDP algorithm modulates synaptic plasticity in a neural network, using the Hebbian learning principle ”neurons that fire together, wire together”. This means that, in the scenario with presynaptic neuron feeding its output to a postsynaptic neuron, a correlated activation of the pre and post neurons results in a strengthening of their synapse. In the case of STDP, the timing of the spikes dictates this correlation: if the presynaptic neuron spikes, and after a window of time the postsynaptic neuron also spikes, a causality relation is assumed and Long Term Potentiation (LTP) occurs, where the synapse is strengthened. On the contrary, if the postsynaptic neuron spikes before the presynaptic one does, Long Term Depression (LTD) occurs and the synapse weight decreases.

This methodology allows a network to statistically learn patterns on a source of information, for example visual information. While the system does not have an explicit guide of what its purpose is, the learning rule implicitly forces it to extract patterns from the information it receives.

On top of the base STDP principle, successful implementations need to implement regulatory mechanisms to prevent runaway synaptic dynamics, which is a cycle of indefinite increase or decrease in synaptic weight [95]. This mechanisms find a stable operating regime achieving homeostasis. The most common methods are intrinsic plasticity, adjusting the excitation threshold of the neurons, or synaptic scaling, adjusting synaptic efficacy. Additionally, ”winner takes all” schemes are usually employed in feed-forward networks so that the neurons are forced to specialise.

2.2.4 Benchmarking of spiking neural networks

To finalise the prerequisites on the state of the art for SNN feature extraction, in this section the performances of the most notable SNN methods are presented. Table 2.1 presents image classification accuracy on MNIST [96], N-MNIST [97], Cifar10, Cifar100 [98], DVS-CIFAR10 [99] and Imagenet [100]. Notice that DVS-CIFAR10 is a dataset resulting from a screen recording of the original frame-based

CIFAR10, acquired with a neuromorphic camera which performed regular motion.

Architecture and training method are briefly specified for each of them. For brevity, commonly known architectures are referred by name, full specifications can be found in the citation. Conversion training is specified for SNNs, as well as other unconventional training procedures. Those with no reference to training methodology indicate that the network was trained through BPTT. Additionally, the table also presents performances for non-spiking ANNs (in blue) in order to showcase the gap in performance and compare the differences regarding architectures used.

This summary is specially relevant for the contribution described in Chapter 4, serving as context and motivation for the developement of the novel S-ResNet network. This work was published in 2022 [23] motivated by the state of the art up to late 2021. Examples of the most relevant systems after the date of publication have also been added for completeness.

Up to 2021, the best performing methods were usually trained by conversion. Later works, including the one in this thesis, obtained competitive results by directly training the SNN with BPTT and surrogate gradient and by proposing adaptations of the ResNet architecture to spiking format. In 2023, the visual transformer architecure was adapted to spiking networks [56], porting the superior accuracy from these architecture to SNN. Apart from that, it is also noticeable how Qiu (2024) [101] obtained high accuracy with a ResNet architecture, by changing the encoding procedure to their Gated Attention Coding, which applies additive attention to the first convolution, which encodes the continuous image values into spiking format.

Ultimately, the table also shows how non-spiking networks still outperform Spiking ones in terms of accuracy. These networks are not constrained to binary communication and are easier to train for image classification tasks. Bridging this gap is an important objective for SNN research, but it is still important to realise that image classification is more suited for non-spiking ANNs than SNNs. Spiking computations encode information in the temporal dimension, making up for the reduced expressiveness of binary communication, this means that their training complexity is the same in image classification than when processing dynamic data such as video, and their expressiveness is bound to the amount of time they run for. On the contrary, ANNs can perform inference for static data in a single time-step, easing training and defining this same expressiveness with the resolution of their floating point operations. Additionally the differentiability issues in SNNs make training less precise than in their conventional counterparts.

These differences seem to indicate that SNNs might never outperform ANNs in accuracy for mainstream tasks such as image classification. Instead, they represent a more efficient embodiment of the Neural Network computing paradigm. As previously discussed, this embodiment has its constraints, which impose limitations, but in exchange other benefits arise. Energy efficiency is the most notable, while further research is trying to unveil more of them. An example is Chapter 5, which demonstrates benefits for spatio-temporal feature extraction.

2.3 Conclusions

In this section, the preliminaries on neural networks and feature extraction have been established, followed by a review on the state of the art for SNN applied to visual feature extraction.

The parallelism between SNN and ANN developments have often been alluded to in this review, as SNNs are just one possible embodiment of the ANN paradigm. As discussed in Section 2.2.4, ANNs are easier to design and train, making for more accurate feature extractors. Still, spiking computations have exploitable benefits due to their sparsity and asynchronous computing. Hence, it is a priority to further develop SNN network architectures in order to make their performance competitive (Chapter 4), while it is also of interest to discover and demonstrate more of their exploitable benefits (Chapter 5).

Tab. 2.1: Image classification accuracy for supervised methods. Blue coloured text indicates a non-spiking method. Method lists the architecture used and any non-conventional training procedure. If no training specification is given, regular BP / BPTT was used.

Model	Method	Dataset	Accuracy
Byerly (2020) [102]	Multi-path + capsule networks	MNIST	99.84%
Rueckauer (2017) [89]	7-layer |Conversion	MNIST	99.44%
Fang (2021) [79]	2Conv 2Fc + PLIF Neurons	MNIST	99.72%
Kaiser(2020) [77]	3-layer |Deep continuous local learning	N-MNIST	99.04%
Kugele (2020) [90]	Densnet |Conversion	N-MNIST	99.54%
Wu (2019) [103]	AlexNet + NeuNorm	N-MNIST	99.53%
Fang (2020) [79]	2Conv 2Fc + PLIF Neurons	N-MNIST	99.61%
Dosovitskiy (2021)[64]	Visual Transformer	CIFAR-10	99.5%
Kolesnikov [104]	Big Transfer	CIFAR-10	99.37%
Rueckauer (2017) [89]	AlexNet Conversion SNN	CIFAR-10	90.85%
Wu (2019) [103]	CifarNet + NeuNorm	CIFAR-10	90.53%
Kim(2020) [105]	VGG9 + BNTT	CIFAR-10	90.5%
Lee(2020)[106]	Spiking ResNet11	CIFAR-10	90.95%
Esser (2016) [87]	9-layer CNN |Conversion	CIFAR-10	89.32%
Zheng (2020) [107]	ResNet18 + Fc + tdBN	CIFAR-10	93.15%
Lee (2020) [21]	10-layer Residual SNN + FC	CIFAR-10	90.95 %
Zheng (2020) [55]	19-layer Residual SNN + FC	CIFAR-10	93.15 %
Fang (2021) [79]	CifarNet + PLIF neurons	CIFAR-10	93.50%
Wu (2021) [108]	VGG-11 |conversion	CIFAR-10	91.24%
Sengupta (2019)[62]	VGG-16 |conversion	CIFAR-10	91.55%
Deng (2021)[109]	ResNet20 Conversion	CIFAR-10	93.58%
Han (2020)[61]	VGG16 Conversion	CIFAR-10	93.63%
This work (2022)	S-ResNet38	CIFAR-10	94.14%
Zhou (2023) [56]	Spiking Transformer	CIFAR-10	95.19%
Qiu (2024) [101]	MS-ResNet18 + Gated Attention Coding	CIFAR-10	96.46%
Kugele (2020) [90]	Densnet |Conversion	DVS -CIFAR-10	65.61%
Wu (2019) [103]	CifarNet + NeuNorm	DVS -CIFAR-10	60.50%
Kim(2020) [105]	VGG7 + BNTT	DVS -CIFAR-10	63.2%
Samadzadeh(2020) [110]	18-layer CNN	DVS -CIFAR-10	69.2%
Zheng (2020) [107]	ResNet18 + Fc + tdBN	DVS -CIFAR-10	67.8%
Fang [22] (2021)	Wide-7B-Net	DVS-CIFAR10	74.4%
Fang [79] (2021)	CifarDVSNet + PLIF Neurons	DVS-CIFAR10	74.8%
This work (2022)	S-ResNet38	DVS-CIFAR-10	72.98%
Zhou [56] (2023)	Spiking Transformer	DVS-CIFAR-10	80.90%
Kolesnikov [104]	Big Transfer	CIFAR-100	93.51%
Esser (2016) [87]	9-layer CNN |Conversion	CIFAR-100	65.48%
Kim(2020) [63]	VGG9 + BNTT	CIFAR-100	66.6%
Han [61] (2020)	VGG16 Conversion	CIFAR-100	70.97%
Deng [109] (2021)	VGG-16 Conversion	CIFAR-100	72.34%
This work (2022)	S-ResNet38	CIFAR-100	74.65%
Zhou [56] (2023)	Spiking Transformer	CIFAR-100	77.86%
Qiu [101] (2024)	MS-ResNet18 + Gated Attention Coding	CIFAR-100	80.45%
Touvron(2020) [111]	FixEfficientNet-L2	ImageNet	88.5%
Dosovitskiy (2021) [64]	Visual Transformer	ImageNet	88.36%
Tan and Lee (2019) [112]	EfficientNet-B7 + RandAugment	ImageNet	85%
Stockl (2020)[92]	Efficientnet-B7 |AMOS Conversion	ImageNet	80.97%
Fang (2021) [113]	SEW ResNet50 + PLIF neurons	ImageNet	63.55%
Zheng (2020) [107]	ResNet34(large) + tdBN	ImageNet	67.05%
Stockl (2020)[92]	ResNet50	ImageNet	75.22%
Stockl (2020)[92]	ResNet50 |AMOS Conversion	ImageNet	75.10%
Szegedy (2015)	Inception V3	ImageNet	78.8%
Rueckauer (2017) [89]	Inception V3	Subset-ImageNet	76.12%
Rueckauer (2017) [89]	Inception V3 |Conversion	Subset-ImageNet	74.60%
Fang [22] (2021)	SEW-ResNet152	ImageNet	69.26%
Zhou [56] (2023)	Spiking Transformer	ImageNet	74.81%
Yao [114] (2024)	Spiking Transformer 24	ImageNet	77.07%
Qiu [101] (2024)	MS-ResNet34 + Gated Attention Coding	ImageNet	70.42%

Chapter 3

Continual Learning

Artificial Neural Networks have achieved great success when trained in independent and identically distributed (i.i.d.) data, where data points are sampled with identical probability as independent events. However, this requirement for i.i.d. inputs limits the adaptability of AI systems, as it becomes unfeasible to train them on dynamic streams of data where, over time, older data might become unavailable, new classes might appear, or domains might shift their distribution [115].

Overcoming this constraint is one of the great challenges in machine learning, and one of the current priorities. Humans (and most animals) are able to continuously learn through life, accumulating knowledge from all past experiences and only forgetting that which is not often used or considered unimportant [116]. Building this knowledge base improves generalisation capabilities, as knowledge from previous experiences can transfer to new ones, while it also allows for faster learning (few shot learning [117]). These are key properties for any future AI systems, which would enable them to adapt to new situations and achieve higher overall intelligence. However, under the i.i.d. data constraint, this requires to collect all this life-long learning data into one dataset and randomly sample it. This poses a scalability problem, due to data storage and training cost, alongside with privacy preservation violations when data cannot be stored. Moreover, this is also a limitation to deployment: If the system is placed in a new scenario and needs to learn online, training on new data without revisiting examples of the old one would cause undesired forgetting. This problem is commonly known as Catastrophic Forgetting (CF) [10, 17, 18].

In order to address this limitation and promote more scalable models, there is a need to develop systems capable of learning and adapting to new tasks while maintaining previously acquired knowledge. Therefore, the main objective of continual learning (CL) approaches is to find a balance for the stability-plasticity dilemma: preserving performance in previous tasks, while still allowing the system to learn new ones [118].

3.1 Incremental learning

In order to evaluate the capacity of machine learning systems in continual learning scenarios, a common approach is framing the problem as an Incremental Learning (IL) challenge. In IL, the system tries to learn a sequence of tasks, one at a time, without access to data from previous or future ones [10, 17, 119, 120]. In this thesis, incremental learning with image classification tasks is used, and the setup is defined without overlap, meaning that each class only appears in a single task. During training, since labelled data is available, access to the task-id is granted. However, during evaluation, IL methods can be classified as task-aware and task-agnostic. The first assumes that the task-id is known at test time, while the latter does not allow its use (at test time) [121]. In this scenario, some of the best performing methods are the ones using rehearsal, which store a subset of samples from each task and class in a memory buffer and access them throughout the training sessions [119, 120]. In recent years, the usage of these exemplars has been questioned, with a growing preference for more privacy preserving methods, which also can scale better over large number of classes [115]. However, as explained in following sections, this exemplar-free incremental learning approaches struggle to tackle some of the causes of CF.

The causes of catastrophic forgetting

As one of the original contributions of this thesis, a definition of catastrophic forgetting is proposed, which allows to identify the specific mechanism that cause accuracy degradation in previous classification tasks. This allows for a better understanding of the contribution of different continual learning approaches and highlights which issues still need addressing when preventing CF. Therefore, this contribution is presented early in the thesis so that the nomenclature it defines can be used when reviewing CL methods in the literature.

The definition decomposes the problem into the following sub-components:

- Weight degradation: including **weight over-writing** and **weight interference**.

- Representation interference: including **representational overlap** and **class-energy imbalance**.

Consider a typical classification problem, where the target distribution for a cross-entropy loss function is a one-hot encoded vector y_c with value 1 for the correct class c and 0 elsewhere. The unormalized output o_c of the network is passed through a softmax function $\hat{o_c} = \frac{e^{o_c}}{\sum_i e^{o_i}}$ and the resulting distribution $\hat{o_c}$ is forced to match y_c by the loss.

Given the activation of the last layer of the network $a_L(x)$, and a classifier layer ψ, the network output for a class c is calculated as the inner product between the classifier weights connected to its logit ψ_{o_c} and the activation $a_L(x)$:

$$o_c = \psi_{o_c} \cdot a_L(x) = \mid \psi_{o_c} \mid\mid a_L(x) \mid cos(\psi_{o_c}, a_L(x)) \tag{3.1}$$

In an incremental learning scenario, after learning the mapping from x to $\hat{o_c}$, later training can modify these output probabilities $\hat{o_c}$, causing catastrophic forgetting when it makes $P(\hat{o_c} \mid x)$ and $P(y_c \mid x)$ diverge.

Considering an incremental learning sequence with t tasks (or training sessions) the activation of a given layer l can be defined as:

$$a_l(x) = x \cdot (W_l^{\in t} + W_l^{\notin t}) \, , \tag{3.2}$$

where $W_l^{\in t}$ are the weights trained during task t and $W_l^{\notin t}$ the weights which were not active while learning that task (because they were deactivated or because they were added in a later stage). Then, in later training sessions, catastrophic forgetting will be caused by **weight overwriting** if $W_l^{\in t}$ is modified and by **weight interference** if $W_l^{\notin t}$ becomes active.

Furthermore, since the value of o_c will be defined by the product in equation (3.1), the **representation overlap** problem is identified in the cosine similarity calculation, as the overlap between representations of different classes will translate into higher logit values for the incorrect ones, increasing the probability of misclassification. Classes learnt in different training sessions are the main cause of this problem, as they are not trained together and, therefore, their separability is not taken into account.

Finally, the **class-energy imbalance** problem can be found in the $\mid \psi_{o_c} \mid$ term, as some classes might have higher weights, promoting interference by scaling the effect of the representational overlap.

3.2 Families of methods

Through the years, many approaches have been proposed to alleviate CF. Comparing them is not always straightforward, given that continual learning scenarios can be defined with different constraints depending on the desired application. The use of exemplars and task-id, as previously mentioned, are examples of requirements that can be made available to the system in certain scenarios, but in others, we might want to avoid. Other examples of properties that influence the usability of methods are: Extra computation during training or inference, growing of network parameters through training or the requirement for pretrained models, among others. To better categorise methods alongside those of similar requirements, the following subsections categorise them in "families of methods" where the strategy is based on a common principle.

3.2.1 Regularisation

Regularisation methods add additional terms to the loss function which promote preservation of previous knowledge. Usually they require storing a copy of the model which serves as checkpoint, then, updates on the model are limited by the regularisation loss, which promotes staying close to the checkpoint model. The more weight this loss has, the more stability and less plasticity the model will have.

These approaches show promising results in task-aware IL, but usually require to be paired with rehearsal [122] or external datasets [123] when not having access to the task-id, as reported in [119].

Weight regularisation

One approach is weight regularisation, which calculates an importance measure for each weight in the network, indicating how critical they are for the performance of previous tasks. Then, the regularisation loss penalises changes to the weights with respect to the network checkpoint proportionally to this importance metric. Popular examples of this approach are Elastic Weight Consolidation (EWC) [124], which estimates importance as the diagonal approximation of the Fisher Information Matrix; Memory Aware Synapses (MAS) [125] which do so by means of the magnitude of the gradient; or Path Integral (PathInt) [126], which accumulates the changes applied to each parameter, which is correlated to the gradient value.

Equation 3.3 exemplifies this with the EWC loss function. Given the current weight matrix θ_i and the weights of the checkpoint $\theta^*_{t-1,i}$, saved at task $t-1$, the regularisation term calculates the diagonal of the Fisher information matrix F for each parameter i.

$$\mathcal{L}(\theta) = \mathcal{L}_t(\theta) + \sum_i \frac{\lambda}{2} F_i \left(\theta_i - \theta^*_{t-1,i}\right)^2 \tag{3.3}$$

Where F is calculated by means of the expectation $\mathbb{E}[\cdot]$ over training data of the derivative of the log-likelihood of the output y given an input x (3.4). As explained in [127], F is equivalent to the second derivative of the loss near a minimum while it can be computed from first order derivatives and is guaranteed to be positive semi-definite. Therefore, it is used to estimate the influence of each parameter in the loss, making the loss give more cost to the changes in those parameters with higher influence.

$$F_i = \mathbb{E}\left[\left(\frac{\partial \log p(y \mid x, \theta)}{\partial \theta_i}\right)^2\right] \tag{3.4}$$

These approaches usually create the network checkpoint when training for a task is completed and a new one starts. This can represent a limitation for scenarios where task boundaries are not defined, therefore [128] proposes to adapt MAS to such setup by estimating when the network has reached a learning plateau through the mean and standard deviation of the loss.

Activation regularisation

Activation regularisation imposes its penalty on activation changes, instead of weight values. The most widely used method for this is Learning Without Forgetting (LWF) [129]. The strategy is based on a method originally proposed for network distillation [130], where the logits of a teacher network are used as objective for a student one, in order to transfer knowledge from the former to the latter.

LWF creates a checkpoint of the network upon task transition and uses it as teacher network for the updated model, constraining the new model to solutions that are close to the logits that the old network returns for the current data.

The loss then adds a L_{distill} term which calculates a modified cross-entropy loss with the targets $y_o'^{(i)}$ being the recorded probabilities.

$$L_{\text{distill}}(y_o, \hat{y}_o) = -\sum_{i=1}^{l} y_o'^{(i)} \log \hat{y}_o'^{(i)} \tag{3.5}$$

where a temperature parameter T is used to smooth logit values as:

$$y_o'^{(i)} = \frac{\left(y_o^{(i)}\right)^{1/T}}{\sum_j \left(y_o^{(j)}\right)^{1/T}}, \quad \hat{y}_o'^{(i)} = \frac{\left(\hat{y}_o^{(i)}\right)^{1/T}}{\sum_j \left(\hat{y}_o^{(j)}\right)^{1/T}}. \tag{3.6}$$

Knowledge distillation has since been added to multiple methods as a knowledge preservation measure [131–133] such as Co2L [133], which learns more transferable representations through contrastive learning and performs knowledge distillation to preserve feature representations close to the ones from the checkpoint.

3.2.2 Replay

Instead of accumulating all data seen in a life-long learning process, replay methods propose approaches which store memory buffers with a limited number of examples and use them to prevent forgetting in the distributions they represent.

Experience replay

The most obvious approach for this is experience replay, which adds the exemplars to the training batch [134]. The challenge then becomes how to select the most representative exemplars, how to update them, augment them and compress them for efficiency. Examples of this are the exemplar selection strategy used in the Icarl method [122], which selects those closer to the class mean in feature space, or the Adaptive Quantization Modules [135] for exemplar compression, which encodes them with discrete autoencoders.

Gradient constraint

Instead of using exemplar buffers for replay, the Gradient Episodic Memory (GEM) [136] constrains the learning process so that the minimisation of the current loss does not increase the loss for the saved exemplars. Both this method, and its later and more efficient version A-GEM [137], alleviate catastrophic forgetting while allowing for backward transfer, as increasing the loss for previous tasks is restricted but minimising it is allowed.

Still, storing buffers of data, both for experience replay and gradient constraint, imposes limitations: Scalability is limited, as the number of exemlpars will continuously grow with as new data arrives or, alternatively, if the exemplar pool is limited in space, the method's performance will decay as the data seen increases. Additionally, when data cannot be stored due to privacy preservation issues, these methods are also not suitable.

Generative replay

An alternative approach to explicit exemplars has been to train generative models which are able to learn the distribution of previous data and generate examples to add as rehearsal examples [138]. These approaches take the problem of catastrophic forgetting into the generative model, which still needs to be updated. Some approaches try to prevent this by applying weight regularisation [139, 140] or using exemplar buffers [141].

Prototype replay

For those scenarios where the use of exemplar memories needs to be avoided, some recent approaches have chosen to substitute the use of data exemplars with pseudo-feature representations, which are generated by storing the statistics of the representation of each class after being encoded by the feature extractor. Then these statistics are used to generate synthetic pseudo-samples, which represent previous distributions and are added to the training batch [142–145]. As with the other replay based methods, these examples allow to avoid representational overlap, as they include samples from previous distributions which will allow the classifier to learn more discriminative classification boundaries.

To further enhance the effectiveness of these prototype based approaches, PASS [142] proposes a self-supervised class augmentation by rotating images 90°. This increases the number of classes by a factor of four, encouraging orientation-robust features during training. Similarly, IL2A [144] applies the self-supervision technique by combining two samples within a batch to form an augmented learnable class. SSRE [143] proposes to apply a selection mechanism to the prototypes, which decides when a sample is learned only with the cross-entropy loss (plasticity) or also with a distillation loss (stability). Furthermore, they temporarily increase the model capacity during training and afterwards introduce a reparametrization to reduce back to the original model size. PRAKA [145] extends PASS by solving some of its shortcomings, introducing prototype reminiscence which replaces the prototypes being represented from a normal distribution, to a more dynamic one that takes into account the position of other classes in the latent representation.

Finally, FeTrIL [146] proposes a prototype variant that relies in a translation of the features $f(c_n)$ of the new classes C_n into the centroid position of an older class C_p by subtracting the mean of the current distribution $\mu(C_n)$ and adding the one from the older class to generate $\mu(C_p)$, as seen in equation (3.7). When multiple classes are available in the current training session, the method shows the best results by selecting the source class C_n which is closest in centroid distance to the target class $\mu(C_n)$.

$$\hat{f}^t(c_p) = f(c_n) + \mu(C_p) - \mu(C_n) \tag{3.7}$$

FeTrIL reports superior performance to other prototype based approaches such as SSRE and PASS in incremental learning scenarios of varying length.

Still, the intrinsic limitation of all these approaches is that the representation of older classes is required to be constant for the prototypes to remain valid. Therefore, many methods use a distillation regularization to minimise this difference [142, 143], while FetrIL completely freezes the backbone. Additionally, the synthetic generation process requires to assume a certain distribution for the old classes, such as a Gaussian distribution centered around their mean. Therefore, these methods will be subject to an approximation error caused by the difference with respect to the real distribution.

3.2.3 Prompt learning

It is a well-known fact in machine learning that, when training for a new task, a pretrained model can serve as a better initialisation, reducing training time and achieving higher overall performance (forward transfer) [147]. This happens when the pretrained weights extract features that are usable for the new task, something that will be bound to happen if the data distribution used for the pretraining stage overlaps with the one for the new task. In the field of natural language processing this has been taken even further, where a large foundational model can be use as knowledge source, and it can be adapted to new tasks by appending a prompt to the input [148], without need for retraining. This strategy is applied to transformer architectures, where the prompt is an additional piece of input, in the form of additional tokens, which provides the context of the task to solve, allowing the network to retrieve the correct answer given its knowledge base. The approach works well when there is high overlap between pretraining tasks and the one to solve, but it has inferior performance compared to fine tuning [149].

Inspired by this approach, recent work has adapted prompting strategies to continual learning, leveraging large pretrained models. Specifically, they use prompt learning, which learns the optimal prompt for the current task and leaves the backbone unchanged.

Learning to Prompt [150] and its successor DualPrompt [151] were the first instances of this approach. In these methods, a query vector is calculated for each input as its encoding after going through the backbone. Then, given a set of randomly initialised prompts and a key vector associated to each of them, the cosine distance between the prompt keys and the query is calculated. Finally the N prompts with lowest distance are appended to the input and trained to reduce the task loss (and the cosine distance of the keys with respect to the query are also minimised). Later improvement came with [152], which made the training process fully differentiable and added a learnable attention vector to the query, to boost adaptability by attending to the relevant parts of this fixed representation.

3.2.4 Parameter isolation

Parameter isolation approaches explicitly define task-specific parameters, associating parameters to a given task and, therefore, allowing to preserve performance on those classes by preventing weight overwriting in them. Additionally, this separation of parameters can also make interference between parameters learnt for different tasks more obvious, facilitating its mitigation.

Methods such as [153–155], often referred to as 'architecture growing approaches', choose to add additional parameters to the network when a new task needs to be learnt, growing the network size and allowing the allocation of an arbitrary size of knowledge as long as the memory and computational requirements are affordable. On the contrary, mask-based parameter isolation, such as HAT [156], PackNet [157], TFM [158] or WSN [159], fix the network size and dynamically allocates parts of the network to different tasks.

Taking HAT as example, for each task t, every layer l learns a sub-network by defining a mask m_l^t over its activations a_l^t. The mask acts as a gating mechanism, defining which activations are inhibited or active (by means of the point-wise multiplication defined in equation 3.8):

$$\tilde{a}_l = a_l \odot m_l \tag{3.8}$$

Where m_l is generated by means of a learnt embedding e_l which is passed through an anhealed sigmoid function σ.

$$m_l = \sigma(s \cdot e_l) \tag{3.9}$$

The scaling factor s is calculated as a function of the N total number of training epochs for the task, the current epoch n, and a maximum scaling value $s_{\max}$ following $s_n = (n \cdot s_{\max})/N$.

This procedure solves the scalability problem of continuously growing networks, at the cost of being bounded by the limits of the network's capacity [157, 158]. This is because, if the number of tasks grows past a certain limit, the number of parameters in this fixed network might not be enough to allocate sub-networks of the optimal size for new tasks, and performance will start declining until no more learning is possible.

A key aspect to make this methods more parameter efficient is feature reuse between tasks. Allowing sub-networks to reuse features from previously learnt tasks reduces the capacity requirement of these later tasks and can boost performance thanks to knowledge sharing. The influence learning in old tasks has on later tasks is known as forward transfer, which is a desirable property in parameter isolation. The opposite would be backwards transfer, where later training influences older tasks, but positive backwards transfer is difficult to achieve without exemplars of previous distributions.

In recent years, parameter isolation methods have been the predominant approach for task-aware IL [10]. Given than in this setup task-id can be accessed during inference, this allows to execute only the sub-system defined for the current task, achieving in many cases zero-forgetting. Furthermore, since the task-id is known, classes from other tasks can be ruled out from classification, therefore there are no issues with representational overlap between classes that were not learned together, neither there is weight interference.

Lately, there has been a growing interest in applying parameter isolation to task-agnostic setups. Deploying these methods without the use of task ID would allow to port their weight overwriting prevention to the challenging task-agnostic setup, but without task-id, the sub-network for the current task can not be selected for deployment. MORE [160] and ROW [161] are two approaches, developed in 2022 and 2023 respectively, where each sub-network selection is framed as a inter-task classification, where exemplars from previous tasks are saved and used to learn a task classifier that can asses whether a sub-network is out of distribution (OOD) or in distribution, meaning that, if the sub-network is in distribution, the data belongs to its task, therefore finding the task-id. These approaches bypass the task-id requirement at the cost of storing exemplars. Alternatively, the Supermask in Superposition (SupSup) [162] approach has targeted this same problem by defining a network of non-learnable weights, which are kept frozen from their random initialisation and defining a mask per task. Then, a combination of masks is selected as a weighted sum of all of the sub-networks, where the weight coefficients are those minimising the output entropy. The coefficients are found through gradient based optimisation, which incurs in extra computing cost and latency during inference, but, for a simple task (Permuted MNIST [163]) demonstrates results approximating task-aware performance.

Still, the deployment of parameter isolation in exemplar-free and task-agnostic setups is an unsolved problem. Currently solutions are still bounded to the use of exemplar memories or, in the case of SupSup, not allowing for backbone training, incurring in extra compute and demonstrated only for simple tasks. Hence, part of the research presented in this thesis (Chapter 6) will study the advantages and limitations of parameter isolation for exemplar-free class incremental learning without task-id, and propose a novel method. This method alleviates the limitations of task agnostic parameter isolation with a trainable backbone, without extra compute, and without extra memory requirements.

3.3 Conclusions

This chapter has introduced the the causes of catastrophic forgetting in incremental learning (which represents one of the contributions of this thesis), and reviewed the state of the art for continual learning methods, which aim at preventing this forgetting. The summaries for those methods which are referenced in later chapters were expanded, fully defining their functionality.

From the ’Families of methods’ summary, it can be seen how an ample variety of methods is available to achieve CL, where each group presents unique properties in terms of requirements, and these requirements define which scenarios the method is suitable for. This same summary also established the limitations imposed by keeping exemplar buffers and task-aware inference, justifying how, looking forward, developments on exemplar-free and task-agnostic incremental learning are key. This motivates the work presented in Chapter 6, which explores this setup and proposes new solutions.

Chapter 4

Advancing SNN feature extraction through residual networks

4.1 Introduction

As discussed in Chapter 2, SNNs provide substantial computational benefits, including energy efficiency, asynchronous processing, and more. Currently, improving their accuracy performance is a major priority, as this will ultimately define which tasks one can afford to solve with this more efficient computing paradigm. As seen in Section 2.2.2, conventional ANN are easier to design and train, therefore, they are likely to achieve higher accuracies. Still, on the downside, they are less suitable for deployment in energy constraint systems. Hence, it is necessary to develop better architectures and training algorithms for SNNs, which will contribute to closing the accuracy gap between ANNs and SNNs, allowing SNNs and neuromorphic computing to serve as the alternative to conventional deep learning when efficiency is needed.

With the objective of improving the feature extraction process of SNNs for visual tasks, the work presented in this chapter focuses on the development of residual networks in spiking format, presenting a study on the key components of modern spiking architectures, and using its conclusions to propose a novel and highly optimized SNN. Results prove how directly training SNNs can outperform training by conversion methods, allowing to exploit all the benefits of spiking computations without compromising accuracy. Additionally, the lessons learnt from the experiments can also be valuable for those designing new SNN feature extractors in the future.

Specifically, the contributions of this chapter are as follows: First, it presents an in-depth study on the possible implementations of spiking residual connections, which highlights their properties in terms of accuracy, network activity, characteristics of their derivatives and implications of the computations in hardware requirements. This study introduces a novel residual connection for SNN, which has been named the “Voltage to Voltage” connection, and a revamped implementation of the “Spikes to Spikes” connection.

Then, it provides empirical results demonstrating the effects of different network design choices on the final accuracy. These include network size, batch normalization strategies, boosting methods, spike generation for frame-based datasets, hyper-parameter optimization and fine-tuning. When designing an SNN, the conclusions drawn from these experiments allow to make optimal design choices maximising the accuracy of the system.

Finally, a new spiking network is defined, which achieves higher accuracy than the previous state of the art (in 2022 at the time of publication) in CIFAR-10 and CIFAR-100, and matching it for DVS-CIFAR10 with many less parameters than previous methods[1].

Additionally, a study on the compromise between latency and accuracy is presented. Through the experiments performed in it, results are obtained which demonstrate a relationship between the processing time and the optimal leakage factor for a leaky integrate-and-fire model.

4.2 Spiking Residual Network

In this chapter, SNNs are developed using the LIF neuron model [54], as introduced in Section 2.2.1. Then, regarding the network architecture, in order to build the most accurate SNN feature extractor, the starting point is to implement a spiking residual network (S-ResNet).

The motivation to choose this architecture is that almost all the non-spiking state of the art ANNs make use of residual connections in order to allow for the training of very deep networks [20, 35, 46, 164]. On the contrary, in the SNN domain, the state of the art was still based in VGG-like architectures for datasets such as CIFAR-10, CIFAR-100 and DVS-CIFAR10 [61, 62]. Therefore, a new S-ResNet is defined that allows to outperform the previous state of the art and justifies the use of residual connections also in the SNN domain.

[1]The code was publicly released at https://github.com/VicenteAlex/Spiking_ResNet.

4.2.1 Implementation of a spiking residual connection

In order to design the S-ResNet, the first step is to define the implementation of the spiking residual connection. The skip connection in a non-spiking network just sums the activation value of a previous layer to the activation of the current one (2.6); however when using spiking neurons, this sum can be performed in several ways.

Given a multilayered feed-forward SNN of LIF neurons, the membrane state vector $u_{l,t}$ of a layer l at time t is given by equation (4.1), where $o_{l,t}$ is the layer's spiking activation and W_l the synaptic weight matrix. These spiking activations are obtained by means of the spiking function g (4.2).

$$u_{l,t} = W_{l-1}o_{l-1,t} + \lambda \cdot u_{l,t-1} \tag{4.1}$$

$$o_{l,t} = g(u_{l,t}) \tag{4.2}$$

Then, the residual information coming from a previous layer at position $l-n$ can be integrated to the current layer l using one of the following strategies:

Spike output to membrane (S2M): The spiking output of a previous layer $l-n$ feeds the membrane potential of the neurons in layer l. A set of synaptic weights W'_{l-n} will be needed to define the amount of voltage communicated by these spikes (4.3). These weights will typically be a non-learnable parameter, then if $W'_{l-n} = U_{th}$, the residual connection will implement an identity mapping when $W_{l-1}o_{l-1,t} + \lambda \cdot u_{l,t-1} = 0$. In any other case, the final activations are not guaranteed to be $o_{l,t} = o_{l-n,t}$.

$$o_{l,t} = g(W_{l-1}o_{l-1,t} + \lambda \cdot u_{l,t-1} + W'_{l-n}o_{l-n,t}) \tag{4.3}$$

Regarding its training through back-propagation, the properties of the residual connection can be observed in the network's derivative. Consider a generic residual block where the residual input $W'_{l-n}o_{l-n,t}$ has $n = 2$ (4.3), skipping the intermediate layer $l-1$, and where $l-1$ has no residual input:

$$o_{l-1,t} = g(W_{l-2}o_{l-2,t} + \lambda \cdot u_{l-1,t-1}) \tag{4.4}$$

Then, deriving (4.3) with respect to $o_{l-2,t}$, we get:

$$\frac{\partial o_{l,t}}{\partial o_{l-2,t}} = \frac{\partial o_{l,t}}{\partial u_{l,t}}\frac{\partial u_{l,t}}{\partial o_{l-2,t}} = \frac{\partial o_{l,t}}{\partial u_{l,t}}(W_{l-1}\frac{\partial o_{l-1,t}}{\partial o_{l-2,t}} + W'_{l-2}) \tag{4.5}$$

Equation (4.5) shows how the residual connection adds an extra $W'_{l-2}\frac{\partial o_{l,t}}{\partial u_{l,t}}$ term to the gradient, a term which is not influenced by the value of the learnable weights W_{l-1}, in contrast to $W_{l-1}\frac{\partial o_{l-1,t}}{\partial o_{l-2,t}}$. This is the reason why this residual connection will alleviate the vanishing gradient problem even when W_{l-1} is arbitrarily small. Still, given that $\frac{\partial o_{l,t}}{\partial u_{l,t}}$ will be the derivative of the spiking function, the skip connection defined by this implementation will have its gradient scaled by the value of the surrogate function, which might introduce noise in the backpropagation process.

The authors in [22] argue that the surrogate derivative g' of $g(u_{l,t})$ will typically not implement a function such that $g'(W'_{l-n}o_{l-n,t}) = 1$ when $o_{l-n,t} = 1$. Therefore, scaling the derivative of the residual stream by this value could contribute to the vanishment or explosion of the gradient.

This kind of connection has previously been used in [21] with $W'_{l-n} = U_{th} = 1$ and in [55] weighted by their threshold-dependent batch normalization (potentially compromising the identity mapping). The S2M connection is represented in Fig. 4.1 as the green connection.

Spike output to spike output (S2S): The spiking output of a previous layer $l-n$ is added to the spiking output of layer l (4.6). If $o'_{l,t} = 0$, this residual connection will successfully implement an identity mapping $o_{l,t} = o_{l-n,t}$.

$$\begin{aligned} o'_{l,t} &= g(W_{l-1}o_{l-1,t} + \lambda \cdot u_{l,t-1}) \\ o_{l,t} &= o'_{l,t} + o_{l-n,t} \end{aligned} \tag{4.6}$$

Additionally, this implementation avoids applying the thresholding function to the residual path. Therefore, when using back-propagation, the contribution of the residual connection will be unaltered by the value of the surrogate function:

$$\frac{\partial o_{l,t}}{\partial o_{l-2,t}} = \frac{\partial o'_{l,t}}{\partial u_{l,t}}\frac{\partial u_{l,t}}{\partial o_{l-2,t}} + \frac{\partial o_{l-2,t}}{\partial o_{l-2,t}} = \frac{\partial o'_{l,t}}{\partial u_{l,t}}W_{l-1}\frac{\partial o_{l-1,t}}{\partial o_{l-2,t}} + 1 \tag{4.7}$$

Regarding the information flow inside the SNN, this kind of connection has some implications that are worth noticing. It is implemented as an addition between activation maps, which is a different operation than adding voltages to a membrane and needs to be supported in the substrate implementing it (or else extra synapses will be needed). Moreover, it allows for the generation of non-binary activation maps, as the sum between activations could result in a value bigger than 1. In order to implement this, it will require to either sum activation maps and communicate non-binary values in the spike activation (as some neuromorphic hardware already supports [165]) or to avoid grouping spikes in one synapse by defining multiple individual connections such that:

$$o_{l,t} + (o_{l-n,t} + o_{l-m,t}) = o_{l,t} + o_{l-n,t} + o_{l-m,t} \tag{4.8}$$

Finally, in network topologies such as the proposed S-ResNet (defined in the following section), we can find situations where the number of neurons d_1 in $o'_{l,t} \in \mathbb{N}^{d_1}$ is different than d_2 in $o_{l-n,t} \in \mathbb{N}^{d_2}$. As proposed in [20], this is solved by applying a 1×1 convolution f to $o_{l-n,t}$ such that $f : \mathbb{N}^{d_2} \rightarrow \mathbb{N}^{d_1}$. This is relevant for the S2S connection because, as seen in Equation (4.9), by applying this convolution $o_{l-n,t}$ gets now multiplied by the learnable W'', which weights the activations transforming them into non-binary voltage values. The implications of these non-binary spiking activations are no different than that of the multiple spikes, it can be implemented as graded spikes in neuromorphic hardware (which communicate non-binary values) or by defining extra synapses. The formulation for the later can be seen in equation (4.10), where the contribution of $o'_{l,t}$ and $o_{l-n,t}$ to the membrane $u_{l+1,t}$ is split as two different incoming connections.

$$o_{l,t} = o'_{l,t} + W''_{l-n} o_{l-n,t} \tag{4.9}$$

$$\begin{aligned} u_{l+1,t} &= W_l o_{l,t} + \lambda \cdot u_{l+1,t-1} \\ &= W_l o'_{l,t} + W_l W''_{l-n} o_{l-n,t} + \lambda \cdot u_{l+1,t-1} \end{aligned} \tag{4.10}$$

This kind of connection has been used in [22]. Its implementation is the same than the one in this work for maps at the same resolution, but it differs in the downsample paths. Unlike this work, the authors add a spiking neuron layer after the 1×1 convolution. This was avoided in this work in order to eliminate the effect of the thresholding in the residual path, including the reduction of granularity in the communication and the effect of the surrogate function in the derivatives of the residual path.

The S2S connection is represented in Fig. 4.1 as the purple arrow.

Voltage to voltage (V2V): The previous two implementations created a residual mapping in the activation map. This residual mapping can also be enforced at the membrane potential level if a V2V connection is defined.

Let the spiking input to a layer $l - n$ be $W_{l-n-1} o_{l-n-1}$ plus a residual input $r_{l-n,t}$. Then, in a V2V implementation, the input that feeds a layer $l - n$ will also become the residual input to the layer l (4.12). Like this, if $W_{l-1} o_{l-1,t} = 0$ and $u_{l,t-1} = u_{l-n,t-1}$ the residual will implement an identity mapping of the membrane potentials such that $u_{l,t} = u_{l-n,t}$. This will also cause $o_{l,t} = o_{l-n,t}$ if the thresholds of the two layers are the same.

$$r_{l,t} = W_{l-n-1} o_{l-n-1,t} + r_{l-n,t} \tag{4.11}$$

$$o_{l,t} = g(W_{l-1} o_{l-1,t} + \lambda \cdot u_{l,t-1} + r_{l,t}) \tag{4.12}$$

Regarding the derivative of the network, deriving with respect to $o_{l-n-1,t}$ in the same setup as before ($n = 2$) we get:

$$\frac{\partial o_{l,t}}{\partial o_{l-3,t}} = \frac{\partial o_{l,t}}{\partial u_{l,t}} \frac{\partial u_{l,t}}{\partial o_{l-3,t}} = \frac{\partial o_{l,t}}{\partial u_{l,t}} (W_{l-1} \frac{\partial o_{l-1,t}}{\partial o_{l-3,t}} + W_{l-3}) \tag{4.13}$$

As it happened for the S2M, the derivative of the residual path will also depend on the surrogate function. Still, in the context of a hierarchical network, compared to an S2M implementation, the surrogate derivative will have less influence on this residual path, as $r_{l,t}$ is a function of $r_{l-n,t}$, which does not depend on $\frac{\partial o_{l-n-1,t}}{\partial u_{l-n-1,t}}$. In the case of the S2M implementation, the residual is $r_{l,t} = W'_{l-n} o_{l-n,t}$ which fully depends on $\frac{\partial o_{l-n,t}}{\partial u_{l-n,t}}$, adding an additional spiking function into the residual path with each residual block.

Finally, notice that implementing the V2V connection will have the same effect in the information flow as S2S. This is caused by the dependency of Eq.4.11 on $r_{l-n,t}$. In Eq.4.14 this expression is unraveled in order to show how the voltage sent by the residual connection $r_{l,t}$ is just a sum of post-synaptic potentials (PSP) from previous layers $W_{l-i \cdot n-1} o_{l-i \cdot n-1,t}$. Therefore, this can be implemented either by defining $(l/n) - 1$ extra connections per each $r_{l,t}$, or by summing the PSPs together and then communicating the voltage value through graded spikes.

$$r_{l,t} = \sum_{i=1}^{(l/n)-1} W_{l-i \cdot n-1} o_{l-i \cdot n-1,t} \tag{4.14}$$

The V2V connection is represented in Fig. 4.1 as the red connection.

From an implementation point of view, this analysis showed how an S2M connection can be accomplished by a single conventional synapse, while S2S and V2V require either to define multiple synapses or to perform a special kind of computation. This computation requires to sum spiking activations together for the S2S connection and to sum PSPs together in the case of V2V. Then the resulting value is transmitted to the membrane of the target neuron. With the neuromorphic hardware available in the present day, this could be implemented

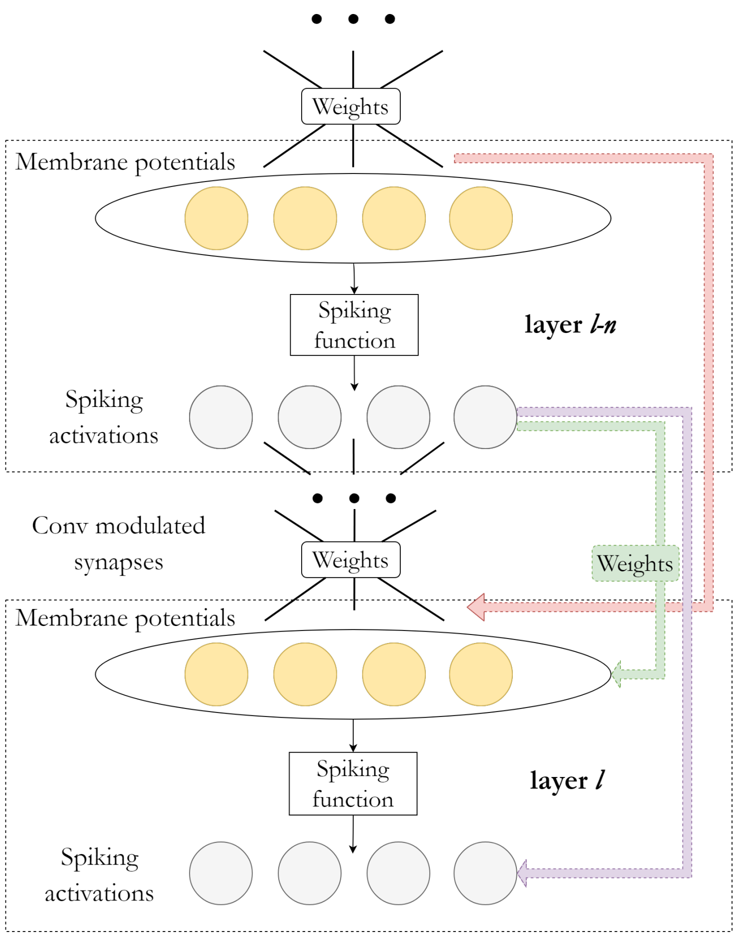

Fig. 4.1: The three possible residual connections in an SNN. In red: Membrane to membrane connection. Purple: Spike output to spike output. Green: Spike output to membrane. Note that the layers are displayed in one dimensional fashion for simplicity, but it is equivalent to a three-dimensional convolutional map if the synapses are defined by a convolutional layer.

by an intermediate neuron which performs the sum and then transmits graded spikes [165].

In this work, the three approaches are tested (section 4.3) analysing their spiking activity (Fig. 4.3) and final accuracy (Table 4.2). S2S is chosen for the final implementation, as it provides the most accurate results. This is consistent with the previous theoretical analysis, as S2S is the only solution avoiding spiking functions in the residual path.

4.2.2 Network topology

With the residual connection implementation defined, the next choice to be made is the global network architecture. In the non-spiking domain, it has already

Tab. 4.1: Table defining the CNN architecture of the original ResNet proposed for the CIFAR datasets. The variable n allows to control the depth of the network.

output map size	32×32	16x16	8x8
# layers	1+2n	2n	2n
# filters	16	32	64

been proven how the original ResNet architecture [20] outperforms feed-forward architectures without residuals; therefore, in order to test if the same principles apply to SNN, the obvious choice is to reuse the same topology.

Depending on the resolution and complexity of the dataset to target, the optimal architecture can vary; that is why in [20] the architecture used for the ImageNet dataset and for CIFAR-10 are different. CIFAR images have a resolution of 32×32, while the images are 224×224 for ImageNet (after resizing), meaning that more downsampling operations will be needed in the latter in order to have a comparable receptive field. As the targets are CIFAR-10, CIFAR-100 and DVS-CIFAR10, the chosen global network architecture is the smaller ResNet proposed for these datasets. The architecture is defined in [20] in a table, such as Table 4.1.

Regarding the batch normalization (BN) layers in the architecture, regular BN can be used in an SNN, but improved performance has been reported by using a time-dependent version of BN. As originally proposed in [166] for conventional recurrent networks, a time-varying BN can be defined where different statistics are learnt for each time-step, as the statistics of different time-steps can differ significantly. When adapted to SNNs, this approach has been called Batch Normalization Through Time (BNTT) [63].

The final architecture defined in this work uses BNTT. As proof of its benefits, Table 4.5 in Section 4.3 demonstrates the performance gains when using it compared to regular BN. A diagram of the final architecture can be found in Fig. 4.2.

To the best of the author's knowledge, this work is the first to implement the aforementioned architecture for SNN training. [21, 55] implement alternative topologies with extra fully connected layers and larger amounts of channels in convolutional layers (see the difference in parameters in Fig. 4.4 in Section 5.3). The authors in [22] define their main network for ImageNet and reuse the original ResNet's topology for this dataset which is different from the CIFAR-10 one. Additionally, they propose a residual network targeting DVS-CIFAR10. Compared to the one developed in this work, their network is wider and shallower (resulting in a larger parameter count), instead of strided convolution, it relies on max

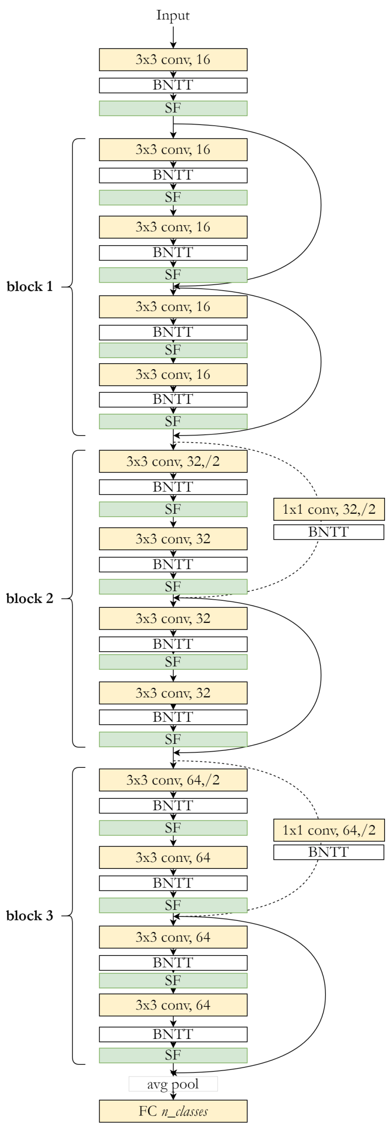

Fig. 4.2: Example architecture for an S-ResNet with $n = 2$ and 16 base filters. SF stands for spiking function.

pooling for downsampling and it processes inputs of 128×128 resolution.

Apart from that, those three networks differ from the proposed one in the normalization strategies, as they use time averaged statistics instead of BNTT, and also in the residual connection implementation.

4.2.3 Boosting strategies

Boosting techniques allow to combine the predictions of multiple weak classifiers to create a stronger one. Previous work in SNNs [79] has already applied simple versions of this strategy by converting the classification layer into a voting layer.

In this chapter, the same approach [79] was tested by adapting the last fully-connected to have $10 \times C$ neurons, where C is the number of classes. Then an average pooling layer of kernel size 10 and stride 10 reduces the dimension back to the number of classes C. This process computes the score of each class as the average of 10 neuron states, which can be seen as a voting scheme for 10 different sub-networks.

In Section 4.3, Tables 4.7 and 4.6 demonstrate the effects of adding the boosting layer. Some networks provided improved performance when using this strategy, while others did not, so the layer is kept only in those cases where it is beneficial. In the final results, only the CIFAR-10 network uses it.

4.2.4 Training framework

The networks are trained to perform image classification through supervised learning (by means of stochastic gradient descent with momentum of 0.9). In order to allow for this classification, the last neuron layer is defined with no leak and cannot spike. Then the voltage accumulated in the layer after T time-steps divided by T is considered the output value.

The output class scores are compared to the ground truth by means of a cross-entropy loss (4.15), where C is the number of classes, $u_{i,T}$ the voltage of neuron i after the last time-step, and y_i are the ground truth labels:

$$L = -\sum_{i}^{C} y_i log(\frac{e^{u_{i,T}}}{\sum_{j}^{C} e^{u_{j,T}}}) \tag{4.15}$$

With the loss defined, the weight updates for the learning process are calculated through BPTT.

The final voltage at each layer is dependent of the contribution of all previous time-steps, therefore the derivative of the loss function with respect to the network weights can be defined as the sum in (4.16), for neurons in the output layer, and as the sum in (4.17) for neurons in the hidden layers.

$$\frac{\partial L}{\partial w_{i,j}} = \sum_{t=1}^{T} \frac{\partial L}{\partial u_{t,i}} \frac{\partial u_{t,i}}{\partial p_{t,i}} \frac{\partial p_{t,i}}{\partial w_{i,j}} \tag{4.16}$$

$$\frac{\partial L}{\partial w_{i,j}} = \sum_{t=1}^{T} \frac{\partial L}{\partial o_{t,i}} \frac{\partial o_{t,i}}{\partial u_{t,i}} \frac{\partial u_{t,i}}{\partial p_{t,i}} \frac{\partial p_{t,i}}{\partial w_{i,j}} \tag{4.17}$$

where $p_{i,t}$ is the current transmitted through the synapses after applying the weights:

$$p_{i,t} = \sum_j w_{i,j} o_{j,t} \tag{4.18}$$

Then, taking into account the temporal dependency of the membrane potential along with its dependency on input spikes, we obtain:

$$\frac{\partial L}{\partial u_{t,i}} = \frac{\partial L}{\partial o_{t,i}} \frac{\partial o_{t,i}}{\partial u_{t,i}} + \frac{\partial L}{\partial u_{t+1,i}} \frac{\partial u_{t+1,i}}{\partial u_{t,i}} \tag{4.19}$$

Notice that $\frac{\partial o_{t,i}}{\partial u_{t,i}}$ requires to compute the derivative of the thresholding function, which is non-differentiable. This is addressed by using a triangle shaped surrogate gradient with $\alpha = 0.3$.

$$\frac{\partial o_{t,i}}{\partial u_{t,i}} = \alpha \max\{0, 1 - |u_{t,i}|\} \tag{4.20}$$

In practice this can be easily implemented using auto-differentiation tools such as Pytorch [167].

4.2.5 Input preprocessing

Frame-based datasets: Frame-based images need to be encoded into spikes in order for an SNN to process them. Works like [63] use a Poisson spike generation process which transforms the image frame into a sequence of spikes. Other works [55, 79] feed the unprocessed frame to the first SNN layer, making the pixel intensity equivalent to a constant input voltage for the first neurons.

The latter allows for better results, as all of the information is presented at each time-step, while the former will require many steps to represent all of the information and will add variability to the data through the randomness in its computation. Still, using a spike generation process is arguably a better representation of a scenario where the input data is spiking information (such as the data coming from event cameras), so choosing one method or another should depend on the objective of the simulation. Therefore, in this work, both approaches are used in order to compare results. The best performing networks are trained without Poisson encoder in order to maximise accuracy. Additionally, images are always normalised with respect to the statistics of the dataset.

Neuromorphic datasets: Data produced by neuromorphic cameras [168, 169] represent the changes in the scene, and these are often presented in event format. An event is a discrete package of information indicating location, timestamp and polarity (i.e. change in brightness).

Events are used to build frames containing spiking activations, which accumulate all events occurring in a time window, and where the frames have two channels: one for positive polarity and one for negative. The size of the time window is defined by the amount of time-steps one wants to have for each sequence. The process is implemented using the SpikingJelly library [170].

Data augmentation: Frame-based datasets were augmented using random horizontal flips and random crops.

4.2.6 Hyper-parameters

The performance of the proposed network depends on certain hyper-parameters, such as the leak factor of the membrane, the number of time-steps or the learning rate for training. The optimal value of these parameters varies depending on the architecture of the network, the training procedure and the task at hand. That is why in order to properly asses how useful an architecture or a training method is, it is first needed to find its optimal hyper-parameter setup.

The challenge is addressed here by using BOHB [171], a hyper-parameter optimization technique that combines Bayesian Optimization (BO) and Hyperband (HB), a multi-armed bandit strategy. Using this method, the hyper-parameters for an S-ResNet of 38 layers (S-ResNet38) in the CIFAR-100 dataset were optimised. The learning rate for this training is divided by 10 at 70%, 80% and 90% of the training process. The resulting hyper-parameters are also used for the rest of networks and datasets, as with the hardware available it could not be afforded to run an individual search per setup.

The best performing parameters are: *leak = 0.874, time-steps = 50, learning rate = 0.0268* for a batch size of 21.

Notice that the target of the search was only to optimize accuracy, therefore the number of time-steps tends to be maximized as it has a monotonically non-decreasing relationship with the accuracy. Section 4.4.2 demonstrates the effects of reducing the number of time-steps.

4.3 Experiments: Empirical tests of components and strategies

In order to maximize the accuracy of the method, a search was conducted to find the key components in state of the art architectures that allow for improved performance. In this section the empirical results, obtained from testing these components, are presented. The results from these comparisons allow to compose a network which outscores previous approaches in multiple datasets.

Residual connection implementation: In section 4.2.1 three ways of implementing residual connections in SNN were defined. Table 4.2 reports the performance of S-ResNet38 with each one of them. The highest accuracy is obtained by the S2S connection. This result is consistent with the previous theoretical analysis, as the residual path in S2S does not go through spiking functions, therefore it allows a better flow of the gradient during back-propagation. Still, the performance of the V2V implementation is very close. On the other hand, the S2M implementation has a substantially lower accuracy. This decrease in accuracy could potentially be attenuated with further hyper-parameter search and improved optimization, but it can be hypothesized that such setup is more difficult to find due to the less convenient gradient properties of S2M.

Apart from that, by adding any of these three residual connections, the network is expected to propagate more spikes to deeper layers. In order to analyse this effect, the spiking activity of the networks is averaged across the test set of the CIFAR-100 dataset (Fig. 4.3). The spiking activation obtained with a non-residual network (spiking VGG11) is also displayed for comparison.

Before starting the comparison, it is important to realise the effect of BNTT in the spiking activation. As observed by [63], by allowing to learn a different learnable weight γ per time-step, the network is allowed to scale the activation of each layer depending on the time-step. Because of this, it tends to localise the spiking activity of each layer in a certain time range. The value of this weight for each network is visualized in the second row of Fig.4.3.

When looking at the S-ResNet networks, it can be seen how there are more layers active at each time-step, as the spiking connections propagate activations to deeper layers bypassing the BNTT weighing. The effect of BNTT is more noticeable in the S2M implementation and less in V2V and S2S. Still, all of them learn a time-dependent weight distribution, indicating that, according to back-propagation, that is the optimal solution for image classification.

Apart from that, S-ResNet activity maps show a characteristic striped pattern. This is caused by how the residual connections always skip one layer, connecting only even-numbered layers (as defined in [20]).

Finally, the more abrupt changes in activation percentage localized in layer 14 and 26 are caused by the resolution change, which changes the number of total neurons in the layer and makes the residual connection go through a 1×1 convolution.

Tab. 4.2: Image classification test performance on CIFAR-10 and CIFAR-100. S-Resnet38 stands for the architecture defined in Section 4.2.2 with $n = 6$ and 32 base filters, trained for 70 epochs.

Residual connection	CIFAR-10 Accuracy	CIFAR-100 Accuracy
S-ResNet38 S2M	89.27 %	68.64 %
S-ResNet38 S2S	94.01 %	74.54 %
S-ResNet38 V2V	93.83 %	73.79 %

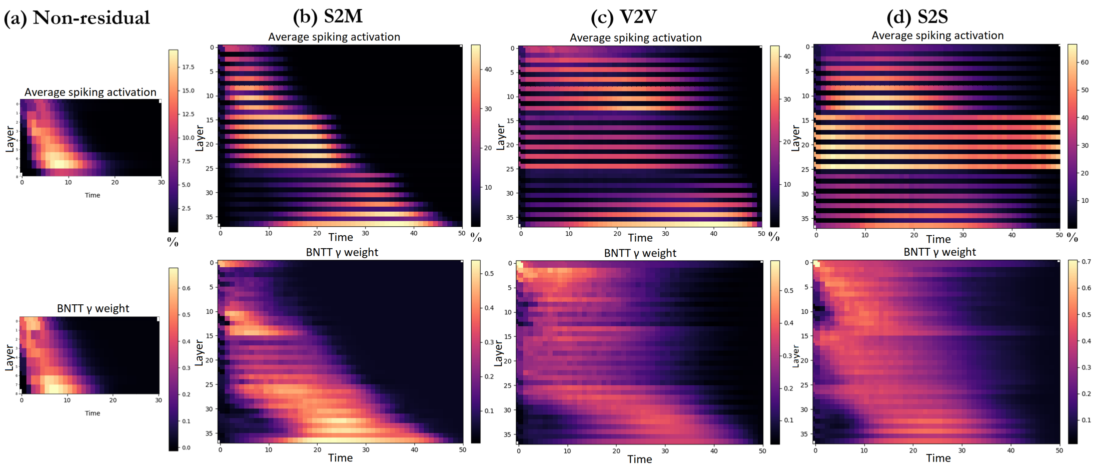

Fig. 4.3: Average activation maps of different networks in the CIFAR-100 test set (first row). The values represent the percentage of neurons active for each convolutional layer at each time. The second row displays the average value (over channels) of the learnable BNTT weight γ per each layer and time-step. Column (a) uses a non-residual VGG11 architecture, (b),(c),(d) use S-ResNet38 with 32 base filters.

Overall the contribution of the residual connections behaves as expected. It propagates the spiking activations to deeper layers, which allows the back-propagation algorithm to successfully train deeper architectures. Additionally, it can be seen how the spiking activity is higher for S2S implementations compared to V2V or S2M, as the “multiple spikes” behaviour favours sending higher amounts of voltage between layers. This can be relevant for applications which are sensible to the volume of spiking activity. In those tasks, the optimal choice for the residual implementation can vary, as there is a compromise between accuracy and volume of spikes.

In cases where a lower network activation is needed V2V poses an efficient alternative to S2S with a very similar accuracy. Regarding their implementation, S2S and V2V require to define extra synapses per residual connection or to implement spike/PSP sum, therefore, S2M is the most suitable option for applications which want to avoid this.

Network depth: The residual connections in S-ResNet allow to increase the depth of the network without the concern of catastrophic accuracy degradation. As expected, this allowed to train very deep architectures. Table 4.3 presents the classification accuracy in CIFAR-10 achieved by the S-ResNet with different depths and the same training hyper-parameters. The results show how the accuracy grows from 20 to 38 layers, but stays roughly the same from 38 to 44.

Tab. 4.3: Image classification test performance on CIFAR-10. S-Resnet stands for the architecture defined in Section 4.2.2 with 16 base filters, trained for 70 epochs.

Network	CIFAR-10 Accuracy
S-ResNet20	90.89 %
S-ResNet38	91.97 %
S-ResNet44	91.96 %

Given these results, for the rest of the experiments S-Resnet38 was taken as the default network. Still, the optimal depth of the network changes depending on the dataset and task to solve, therefore, for those researchers looking for optimal performance, it is encouraged to tune this parameter for their specific task.

Spike generation for frame-based datasets: As mentioned in Section 4.2.5, when working with frame-based datasets, two different methods were tested for the spike encoding process. One consists in transforming the intensity values into spikes by means of a Poisson spike generation process. The other consists in transforming them by means of the first convolutional layer (i.e. feeding the raw image to the network).

As expected, the results in Table 4.4 show how encoding by means of the first convolutional layer gives a better result than generating spikes as a Poisson process. In order to maximize accuracy, for the rest of experiments the encoding by convolution approach was used.

Tab. 4.4: Image classification test performance on CIFAR-100. Except for the spike generation process, both architectures and training procedures are identical. Trained for 100 epochs.

Network	CIFAR-100 Accuracy
S-ResNet38 Poisson spike generation	64.96 %
S-ResNet38 Raw image	69.03 %

Batch normalization strategies: Performances using time-dependent BN statistics versus time averaged statistics were compared. Table 4.5 shows how BNTT outperforms regular BN for the same network.

Boosting layer: As introduced in Section 4.2.3, a simple boosting layer can improve the accuracy of the system in some cases. Tables 4.6 and 4.7 show

Tab. 4.5: Image classification test performance on CIFAR-100. Except for the batch normalization module, both architectures and training procedures are identical. S-ResNet stands for the architecture defined in Section 4.2.2 with $n = 6$ and 32 base filters. Trained for 70 epochs.

Network	**CIFAR-100 Accuracy**
S-ResNet38 BNTT	74.54 %
S-ResNet38 BN time averaged	70.82 %

the effect of this component in the accuracy of the networks. In the CIFAR-10 datasets the accuracy is improved by using this technique, while in the CIFAR-100 one, where there are more classes, increasing the size of the last fully connected in order to perform boosting ends up being detrimental.

Tab. 4.6: Image classification test performance on CIFAR-100. S-Resnet38 stands for the architecture defined in Section 4.2.2 with $n = 6$ and 16 base filters. Wider architectures use 32 base filters and "boosting" indicates the use of a boosting layer (Section 4.2.3). Wider architectures trained for 70 epochs, regular architectures trained for 200 epochs.

Network	**Parameters**	**CIFAR-100 Accuracy**
S-ResNet38	639,760	68.71 %
S-ResNet38 + boosting	697,360	64.60 %
S-ResNet38 wider	2,399,776	74.46 %
S-ResNet38 wider + boosting	2,514,976	73.21 %

Tab. 4.7: Image classification test performance on CIFAR-10. S-Resnet38 stands for the architecture defined in Section 4.2.2 with $n = 6$ and 16 base filters. Wider architectures use 32 base filters and "boosting" indicates the use of a boosting layer (Section 4.2.3). Wider architectures trained for 70 epochs, regular architectures trained for 200 epochs.

Network	**Parameters**	**CIFAR-10 Accuracy**
S-ResNet38	634,000	91.97 %
S-ResNet38 + boosting	639,760	92.00 %
S-ResNet38 wider	2,388,256	92.66 %
S-ResNet38 wider + boosting	2,399,776	93.77 %

Parametric Leaky Integrate-and-Fire: The authors in [79] propose to learn the leak coefficient of the LIF neurons directly through back-propagation as another parameter of the network. By doing this they can also afford to learn a different leak value for each layer. They call this method the Parametric Leaky integrate-and-fire (PLIF) neuron. Table 4.8 shows the results after training S-ResNet38 with PLIF and with a single leak coefficient learned through hyper-parameter search.

Tab. 4.8: Image classification test performance on CIFAR-100. Except for the learnable leak factor, both architectures and training procedures are identical. Trained for 200 epochs.

Network	CIFAR-100 Accuracy
S-ResNet38 LIF	68.71 %
S-ResNet38 Parametric LIF	64.93 %

The best results are not achieved using the PLIF neuron; still, this strategy is arguably a very efficient way of finding this hyper-parameter, and it is against intuition that the best value is not found through gradient descent, a phenomenon probably related to sub-optimal gradient descent hyper-parameters. For this reason, it was tested again for the search of a shared leak value instead of calculating a different one per layer. Table 4.9 shows the difference between the leak value found through hyper-parameter search and the one found by back-propagation. It is interesting to see how the two values differ by a considerable amount, having the one found by back-propagation a slower leakage than the one found through the BOHB method.

Tab. 4.9: Optimal leak coefficient for ResNet38 in CIFAR-100 obtained through two different methods (A single coefficient shared by all layers). "Hyper-parameter search" uses BOHB algorithm to optimize the parameter. The value for this method corresponds to the mean among the 6 best performing configurations found with its corresponding standard deviation in parenthesis. "Learned through PLIF" learns the value by backpropagation during training, the value corresponds to the result after 70 epochs of training.

Method	Leak coefficient
Hyper-parameter search	0.889 ($\pm$ 0.003)
Learned through PLIF	0.986

Still, both values perform well when the network adapts its weights to work with them. The performance comparison between them can be found in Table 4.10, where the network trained with the BOHB optimized value is compared to an identical network which learned the shared leak value through PLIF.

Tab. 4.10: Image classification test performance on CIFAR-100. In "S-ResNet38 wider + Boost Single PLIF" one single leak value is learned for all layers. Except for the learnable leak factor, both architectures and training procedures are identical. Trained for 70 epochs.

Network	CIFAR-100 Accuracy
S-ResNet38 wider + Boost LIF	73.21 %
S-ResNet38 wider + Boost Single PLIF	72.44 %

Extra training data: In the deep learning domain, most state of the art performances in computer vision are achieved by means of fine tuning. This strategy consists in taking a network that has already been trained in a different dataset and then training it further for the task at hand. In the visual domain this strategy works well, as visual data has many transferable features.

This strategy was tested by pre-training the networks with CIFAR-100 and then fine-tunning for DVS-CIFAR10 and CIFAR-10. The results are presented in Table 4.11 and Table 4.12. This yields higher accuracy results in all cases except for the larger S-ResNet in CIFAR-10. Moreover, these trainings converge faster, making it a great solution for any further work building on top of these feature extractors. In the public code associated to this work, users can find the pre-trained weights to perform fine-tunning in any future system building from this one.

Tab. 4.11: Image classification test performance on DVS CIFAR-10. Pre-train column indicates if the network was trained from scratch or pre-trained with a certain dataset. S-Resnet38 stands for the architecture defined in Section 4.2.2 with $n = 6$ and 16 base filters. Wider architectures use 32 base filters and "boosting" indicates the use of a boosting layer (Section 4.2.3). Trained for 70 epochs with learning rate reduction at 50%, 70% and 90% of the training process.

Network	**Pre-train**	**DVS CIFAR-10 Acc**
S-ResNet38	No	63.3 %
S-ResNet38	CIFAR-100	70.4 %
S-ResNet38 wider + boosting	No	65.5 %
S-ResNet38 wider + boosting	CIFAR-100	69.8 %

Tab. 4.12: Image classification test performance on CIFAR-10. Pre-train column indicates if the network was trained from scratch or pre-trained with a certain dataset. S-Resnet38 stands for the architecture defined in Section 4.2.2 with $n = 6$ and 16 base filters. Wider architectures use 32 base filters and "boosting" indicates the use of a boosting layer (Section 4.2.3). S-Resnet38 Trained for 200 epochs from scratch and for 100 when fine-tuned. Wider architectures trained for 70 epochs.

Network	**Pre-train**	**CIFAR-10 Acc**
S-ResNet38	No	91.97 %
S-ResNet38	CIFAR-100	92.44 %
S-ResNet38 wider + boosting	No	93.77 %
S-ResNet38 wider + boosting	CIFAR-100	93.59 %

DVS-CIFAR10 image resolution: The event streams found in the DVS-CIFAR10 dataset were generated by recording 10,000 images from the original CIFAR10 dataset with a DVS camera while applying a repeated closed-loop smooth movement [99]. Despite the resolution of CIFAR-10 being 32×32, the DVS camera resolution was 128×128 and therefore the resulting event maps have also a 128×128 resolution. As the S-ResNet architecture is optimized for inputs of size 32×32, in the previously presented experiments, DVS-CIFAR10 frames were downsampled to that resolution.

In most datasets, downsampling the input causes information loss and therefore accuracy degradation. In order to test if this applies to the unique case of DVS-CIFAR10, the performance using 64×64 and 128×128 input resolution is also tested. The architecture of the network was adapted for the new input sizes by adding, in the case of 64×64 a stride=2 in the first convolution (c32k3s2), and in the case of 128×128 a stride=2 and kernel=5x5 in the first convolution (c32k5s2) followed by a Max Pooling of stride=2 and kernel=2 (MPk2s2).

Table 4.13 presents the test results with the three resolutions. It can be seen how the best performance is obtained when using a 64×64. No improvement was found by using the full 128×128 resolution. The best architecture for full resolution uses a bigger kernel and max pooling, similarly to how [20] handles the bigger ImageNet frames. A possible hypothesis is that this setup does not bring improved performance because the down-scaled 64×64 events already contain the necessary information and therefore the bigger 128×128 network just brings unnecessary complexity.

Tab. 4.13: Image classification test performance on DVS-CIFAR10. S-Resnet stands for the architecture defined in Section 4.2.2 with 32 base filters, trained for 70 epochs and with CIFAR100 pre-training.

Network	**Resolution**	**CIFAR-10 Accuracy**
S-ResNet38	32×32	71.80 %
S-ResNet38 c32k3s2	64×64	72.98 %
S-ResNet38 c32k5s2 MPk2s2	128×128	72.51 %

4.4 Results

4.4.1 State of the art comparison

In this section the final results are compared to the current state of the art for image classification in the CIFAR-10, CIFAR-100 and DVS-CIFAR10 datasets.

As noted in [79], most previous works train on the training set, evaluate the test set at each step, and then report the highest test accuracy obtained. This approach can be considered to be reporting validation accuracy rather than test.

For these experiments, the test set is evaluated after all the training epochs, without using its value for tuning the training. Additionally, validation accuracy is evaluated in the same manner than the previous methods in order to make a fair comparison.

The developed S-ResNet outperforms all previous SNN methods in classification accuracy for the CIFAR-10 and CIFAR-100 datasets (Table 4.14). In the DVS-CIFAR10 dataset, the validation accuracy for the best performing network outperforms it, but when measuring test score, the new S-ResNet is superior. One potential reason for the slight reduction of performance in DVS-CIFAR10 can be the choice of hyper-parameters, which were tunned for the CIFAR-100 dataset and transfered to the rest.

Before this work, in the CIFAR-10 and CIFAR-100 datasets, the most accurate network was a conversion method. These new results prove how directly training an SNN can perform better without the need of imitating non-spiking computations.

Moreover, in Table 4.15 the performance of the new S-ResNet is compared to its non-spiking ANN version. Specifically, the version with 16 and 32 base filters without boosting was used. It can be seen how the performance on the trained SNN is not far from its non-spiking counterpart, demonstrating how improvements in SNN training can push these technologies to comparable levels with conventional deep learning.

Comparing to the previous trainable SNN architectures, the newly proposed network uses many less parameters. Fig. 4.4, 4.5 and 4.6 show a map of the accuracy versus the number of parameters. The main cause for the difference in parameters is that the S-ResNet has a smaller number of channels in convolutional layers and only a single fully-connected layer. Then, even when this network is deeper than the others, it is actually lighter in terms of synaptic connections.

4.4.2 The latency - accuracy compromise

Apart from raw accuracy, the efficiency of algorithms is a major factor when deploying systems in the real world. For image classification in SNN, the number of time-steps used for prediction regulates a trade-off between accuracy and time or volume of computations.

In order to elucidate the effect of this trade-off in the system, in Table 4.16 the accuracy of S-ResNet38 is presented with different numbers of time-steps. Starting from the best network trained with 50 time-steps, accuracy degradation is tested when dropping the last 10/20/30/40 steps. Additionally, this is compared

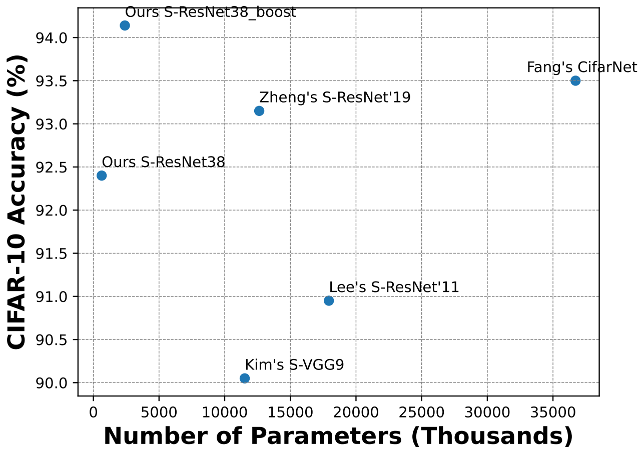

Fig. 4.4: CIFAR-10 accuracy versus number of parameters. The proposed network is compared to the best performing trainable SNNs and the other spiking ResNets. “S-ResNet38_boost” uses the wider architecture with 32 base filters. The number of parameters for other works was counted using their publicly available code.

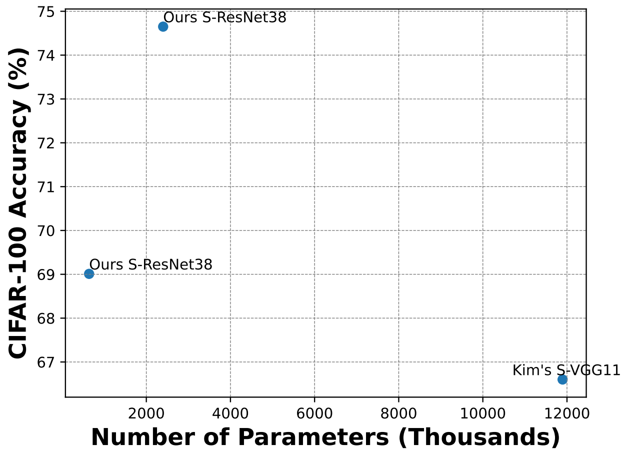

Fig. 4.5: CIFAR-100 accuracy versus number of parameters. The proposed network is compared to the best performing trainable SNN in this dataset. The two results for S-ResNet38 correspond to the same network with 16 or 32 base filters (where 32 base filters has more parameters than 16). The number of parameters for other works was counted using their publicly available code.

Tab. 4.14: Image classification validation performance on CIFAR-10, CIFAR-100 and DVS-CIFAR10. The S-Resnet38 in CIFAR-10 and CIFAR-100 stands for the wider version of the architecture defined in Section 4.2.2 with $n = 6$, 32 base filters, and boosting layer. In DVS-CIFAR10 the 16 filters version without boosting and with the pre-training step was used. S-ResNet' stands for the residual network in [55], as it follows a different architecture than this work's S-ResNet.

Network	Method	Dataset	Accuracy
Kim [63] S-VGG9	Spiking BP	CIFAR-10	90.05 %
Lee [21] Residual SNN (11)	Spiking BP	CIFAR-10	90.95 %
Zheng [55] S-ResNet'19	Spiking BP	CIFAR-10	93.15 %
Fang [79] CifarNet	Spiking BP	CIFAR-10	93.50%
Wu [108] VGG-11	SNN conversion	CIFAR-10	91.24%
Sengupta [62] VGG-16	SNN conversion	CIFAR-10	91.55%
Stockl [172] ResNet-50	SNN conversion	CIFAR-10	92.42%
Deng [109] ResNet-20	SNN conversion	CIFAR-10	93.58%
Han [61] VGG16	SNN conversion	CIFAR-10	93.63%
This work S-ResNet38	**Spiking BP**	**CIFAR-10**	**94.14%**
Kim [63] S-VGG9	Spiking BP	CIFAR-100	66.6 %
Han [61] VGG16	SNN conversion	CIFAR-100	70.97%
Deng [109] VGG-16	SNN conversion	CIFAR-100	72.34%
This work S-ResNet38	**Spiking BP**	**CIFAR-100**	**74.65%**
Kim [63] S-VGG9	Spiking BP	DVS-CIFAR10	63.2 %
Zheng [55] S-ResNet'19	Spiking BP	DVS-CIFAR10	67.8 %
Fang [22] Wide-7B-Net	Spiking BP	DVS-CIFAR10	74.4%
Fang [79] CifarDVSNet	**Spiking BP**	**DVS-CIFAR10**	**74.8%**
This work S-ResNet38	Spiking BP	DVS-CIFAR10	72.98%

to the result obtained by directly training with less time-steps.

The results show how for CIFAR-100, the network trained with 20 steps performs better than dropping the last 30 steps of a 50-step network. Still this same experiment in the CIFAR-10 dataset shows the opposite results by a close margin, indicating that the 50-step network had a more complete training.

At 10 steps, the degrading of the 50-step network becomes more obvious. Interestingly the network trained with 20 time-steps does not degrade as much, as it is only losing half of its computations and therefore still managing to extract the core visual features.

Finally, it is a plausible hypothesis that the optimal leakage coefficient for the neurons might be correlated to the number of time-steps the network is ran for. Given that the leak factor used was obtained through the hyper-parameter search process, and given that this process prioritized large amounts of time-steps, the optimal leak factor for 20-step inferences could be different from the one used in previous experiments. This was empirically tested by training the network again

Tab. 4.15: Image classification performance on CIFAR-10 comparing the ANN version of ResNet to the newly developed S-ResNet. All architectures trained for 70 epochs and the same hyper-parameters. S-Resnet38 stands for the architecture defined in Section 4.2.2 with $n = 6$ and 16 base filters. Wider architectures use 32 base filters.

Network	Method	CIFAR-10 Accuracy
ResNet38	ANN	92.33 %
ResNet38 wider	ANN	93.56 %
S-ResNet38	SNN	91.97 %
S-ResNet38 wider	SNN	92.66 %

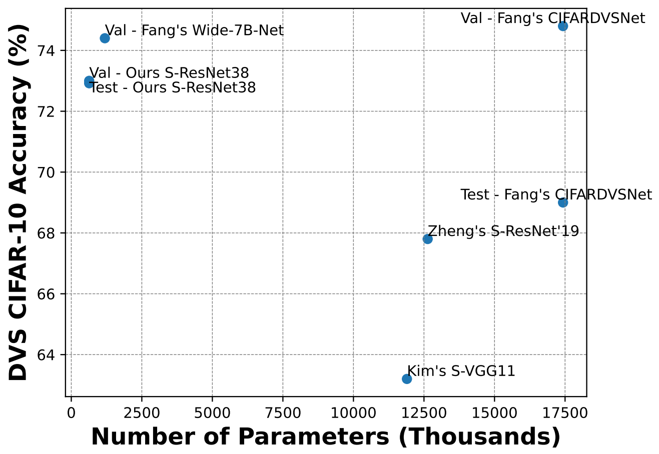

Fig. 4.6: DVS CIFAR-10 accuracy versus number of parameters. The proposed network is compared to the best performing trainable SNNs and the other spiking ResNets. The number of parameters for other works was counted using their publicly available code. The "Val" prefix stands for validation accuracy while "Test" stands for testing accuracy.

Tab. 4.16: Influence of the number of time-steps in the validation accuracy. Results of the evaluation of the best performing S-ResNet38 with boosting. Training time-steps specifies the number of steps used during training, inference time-steps the steps used for inference. If the inference number is smaller than the training one, early stopping is applied and the last N time-steps (and learned BNTT layers) are not used. For comparison, the training is reproduced also with 20 time-steps. Clarification: The architecture is the same but the results for CIFAR-100 use the weights trained in CIFAR-100 and the CIFAR-10 results use the weights trained in CIFAR-10.

Inference t-steps	Training t-steps	CIFAR-100 Acc	CIFAR-10 Acc
50	50	73.40 %	94.10 %
40	50	73.14 %	93.96 %
30	50	71.75 %	93.61 %
20	50	65.78 %	91.93 %
20	20	67.70 %	91.28 %
10	50	15.15 %	63.86 %
10	20	62.28 %	90.45 %

Tab. 4.17: CIFAR-10 validation accuracy for inferences of 20 time-steps. The first network was trained with 50 time-steps in training time, the others were trained with 20 time-steps. The first two networks use the leak value learned through hyper-parameter optimization done for the 50-step network. The third one optimizes the leak value during its training through PLIF neurons.

Leak factor	Inference t-steps	Training t-steps	CIFAR-10 Acc
0.874	20	50	91.93 %
0.874	20	20	91.28 %
0.995	20	20	92.8 %

with PLIF neurons, a process that allows to optimize the leak value in a single training run. The results, as seen in Table 4.17, prove how a better performance is obtained when the leak coefficient is optimized for the number of inference steps, confirming the hypothesis.

This study shows how the optimal solution is to perform training with the same number of time-steps that will be used at inference time and to optimize hyper-parameters such as the leak factor for this same objective. Still, the tested SNNs can withstand the effect of early stopping, retaining most of their accuracy even when big percentages of their computation steps are dropped. This allows to provide early estimates in time sensible tasks or to reduce computational cost.

4.5 Conclusions

In this chapter a new SNN architecture was presented, which outperforms the previous state of the art in different image classification datasets. This system is the product of an in-depth study on spiking residual connections and design choices based on the empirical results from multiple experiments, which demonstrate the effects of different design choices in the final performance. On top of that, the analysis performed on residual connections sheds new light on the effects of these connections in terms of network activity and hardware requirements. The lessons learned from these studies also serve as a guide for SNN design, as they allow to make informed choices when building a new SNN feature extractor.

The results of this work demonstrate how SNNs do not need to use conversion methods in order to maximize their accuracy. Additionally, they contribute to pushing their performance closer to that of non-spiking deep learning. From here, I hope that new applications can benefit from increased accuracy by fine tuning the newly developed networks and more experiments can follow in order to keep pushing the SNN state of the art.

Chapter 5

The advantage of Spiking Neural Networks for spatio-temporal feature extraction

5.1 Introduction

From an application point of view, SNN's sparse and asynchronous computations have been demonstrated to provide great gains in energy efficiency when implemented in neuromorphic hardware [51], hence becoming their major selling point. Still, their differences with respect to conventional ANNs go beyond, as their event-driven temporal dynamics provide an alternative paradigm for temporal processing. The potential advantages of this paradigm have often been overlooked, therefore, a study demonstrating its exploitable properties demands attention.

To provide such demonstration, this chapter focuses on the task of event-based action recognition. This is motivated by the fact that SNNs are naturally suited for the processing of event-based data. These networks are able to integrate input over time and their neurons are activated in an event-based manner, hence their application to event-based data has been a topic of interest [70, 88, 173]. Additionally, given the recent surge in popularity of event-based cameras, research on event-based action recognition is a major priority, making neuromorphic video [168, 169] the perfect target.

Despite the ample variety of conventional frame-based action recognition datasets, the options for event-based action recognition are very limited [70, 174], forcing many researchers to resort to artificially generated datasets, which either convert frame sequences to events [175, 176] or generate them from simulations

[177, 178]. Alternatively, those works employing real data from an event camera [79, 81, 179, 180] have mainly resorted to IBM's DVS Gesture Dataset [70].

In this chapter, it will be proven how solving the action recognition task in the DVS Gesture dataset does not require a network implementing temporal feature extraction. Accumulating events into frames and processing them with an image classifier yields >95% accuracy.

To bypass the limitation of DVS Gesture, a new task is proposed[1], DVS-Gesture-Chain (DVS-GC), which can only be solved by those systems capable of perceiving the ordering of events in time.

Perception of order is a fundamental part of many temporal tasks. Specifically, in action recognition the time dependencies defined by relative order are of critical importance. Often, actions have a sequential nature, where they are composed of a smaller set of sub-actions, and perceiving their ordering is essential to identify the overall action. Further to that, the context of an action and its relationships of causality are also based on the perception of order between actions. Consequently, evaluating this capacity is crucial when designing action recognition systems.

In order to evaluate the aforementioned capacity, DVS-GC leverages the DVS-Gesture data and combines its gestures into chains of gestures, making the chain the actual action class to recognise.

Using this new task, it will be shown how Spiking Neurons enable spatio-temporal feature extraction without the need for recurrent synapses, demonstrating a form of temporal computation which is different from the one in conventional ANNs and providing an alternative approach to time processing. Additionally, the presented study also analyses the differences between this new computing paradigm and conventional Recurrent Neural Networks (RNN), and further develops the current understanding of it by demonstrating the effects of membrane potential leak and reset mechanism. Specifically, it is shown how the reset by subtraction approach [181] can cause slow adaptation to incoming inputs. Then it is proven how this can be alleviated by voltage leak or by using a reset to zero strategy, leading to improved action recognition accuracy.

Finally, the role of temporal attention is explored, which arises when time-dependent weights and normalization are used to perceive order, and allows to illustrate the difference between time-dependent and time-invariant feature extraction.

[1]Code is made available at https://github.com/VicenteAlex/DVS-Gesture-Chain

5.2 DVS Gesture Chain

5.2.1 Event-based datasets

The event-based sensor market is still in its infancy and its still limited commercial adoption has not allowed to collect large volumes of event-based data. Currently, many of the datasets used in computer vision are artificially created from frame-based data or simulations. N-MNIST, N-Caltech101 [97] and DVS-CIFAR10 [99] are three popular datasets created through screen recordings of the original frame-based data with a neuromorphic camera. Alternatively, frame-based datasets have also been converted into events directly through software [175, 176, 182]. Finally, [178, 183] provide simulators for the generation of synthetic event data.

Still, the most desirable option for the development of event-based systems is to use data from a real-world acquisition. In the present day, most of the available natively neuromorphic datasets are still simple compared to traditional frame-based ones. For classification tasks we can find: N-CARS [184] a binary classification dataset, ASL-DVS [185] a 24 class sign language classification task, NEFER [186] with 7 classes for facial expression recognition, DailyActionDVS [187] a 12 class action recognition task and DVS-Gesture [70] the widely used 11 class action recognition dataset. Other available datasets are the DHP19 [69] pose estimation dataset, the Gen1 [188] and 1 Mpx [189] object detection datasets, and DSEC [190] for optical flow estimation.

A study on the relevance of neuromorphic datasets for SNN evaluation was presented in [191]. In this work, it was proven that collapsing all events into a single frame and performing recognition with an image classifier did not affect classification performance in N-MNIST or N-Caltech101, but did decrease it in DVS-Gestures. The results in this chapter will show how, when the event integration is done in multiple frames instead of just one, DVS-Gestures can also be solved by a non-temporal image classifier, evidencing its lack of temporal complexity.

Regarding non-classification datasets, the available options are ill-suited for rigorous evaluation of temporal processing. Despite temporal information being exploitable to gain improvements in accuracy, object detection or pose estimation tasks can still be solved by accumulating events in time-windows and performing inference in one frame. In the same way, optical flow estimation can be performed using two frames of accumulated events.

5.2.2 Defining the DVS Gestures Chain task

The objective is to define an action recognition task in event-based sequences that requires the perception of temporal dependencies, i.e., relationships where the meaning of an action is contingent to those that happened previously. To create such temporal dependencies, DVS-GC leverages the DVS-Gesture dataset and combines N of its gestures $G = \{g_1, g_2, ..., g_N\}$ into chains of gestures $G_c = \{(g_i, g_j, ..., g_k) \mid g_i, g_j, ..., g_k \in G\}$, where each g is a gesture from G. Then each of these chains is considered as its own class, meaning that (in an example with 2 gesture long chains) a chain composed of gesture A and then B will be labelled as class AB, and a chain composed of gesture B and then A will be labelled as class BA. Therefore, to correctly identify a class, it is essential not only to recognise the individual gestures but also to understand the sequence in which they occur, making DVS-GC an action recognition task that demands perception of temporal order.

Event data processing

When using neural networks to process streams of asynchronous events, it is common practice to discretise the time dimension by accumulating events in frames using a constant time window [55, 77, 79, 192]. This allows to process the sequence with an arbitrary number of discrete time-steps and to train using methods such as Backpropagation Through Time (BPTT).

Representing the event sequence as a function $e_{t,x,y,p}$, which has value 1 when the position (t_i, x_i, y_i, p_i) is active, and 0 otherwise, its discretised frame representation $F_{j,x,y,p}$ can be calculated as:

$$F_{j,x,y,p} = \sum_{t=W(j-1)}^{Wj} e_{t,x,y,p}, \tag{5.1}$$

where j is the frame index (or time-step), W the time window, t represents time, x and y are the spatial coordinates and p the polarity.

Naturally, the cost of this discretisation is that it makes it impossible to distinguish the precise timing or the relative order of occurrence of events within a frame.

This strategy is used for all experiments, both with DVS-Gestures and DVS-GC. Then, at each time-step j a new frame F_j will be fed to the network.

When this quantisation is applied to the input of an SNN, the result is an approximation of the one obtained in an asynchronous implementation with infinite time resolution, as the effect is simply a reduction of time resolution, where groups of events are collapsed into the same time-stamp. In this scenario, the SNN's voltage leak can be represented as the percentage of voltage lost from one time-step to the next, meaning that this leak coefficient will be conditioned by the number of simulation time-steps.

On the contrary, non-temporal ANNs cannot accumulate information from different time-steps, therefore collating events into frames is the only way in which they can combine information from events happening at different instants. This is further discussed in the results section.

Creation of the action classes

The creation of classes in DVS-GC is parametrised by the length of the gesture chain L and the number of gestures N used in the chain. Then the number of classes generated C will be equal to all possible combinations (5.2). Alternatively, a class generation method is also provided which does not allow to repeat the same gesture in consecutive positions of the chain, reducing the possible permutations of the chain (5.3). By using different values of N, L and the methodology with and without repetition, different datasets were created in order to evaluate the studied networks (results shown in Section 5.3).

$$C = N^L \tag{5.2}$$

$$C = N(N-1)^{L-1} \tag{5.3}$$

Chaining of events

Given the stream of 4-dimensional events (x, y, time, polarity) provided by the DVS-Gesture dataset, they are transformed into frames by accumulating events in a time window, as done in most state of the art systems [55, 77, 79, 192]. The initial number of generated frames F per sequence is user defined and will be constant for all instances in the dataset. The resulting frames have two channels, one for positive polarity and one for negative.

Given that the gesture instances are obtained from a set of users under different lighting conditions, the gesture chains are created combining the gestures from the same user under the same lighting condition. This avoids sudden changes in illumination or the appearance of the user, which could help the system identify the transition between a gesture and the next. It is also worth noting that the

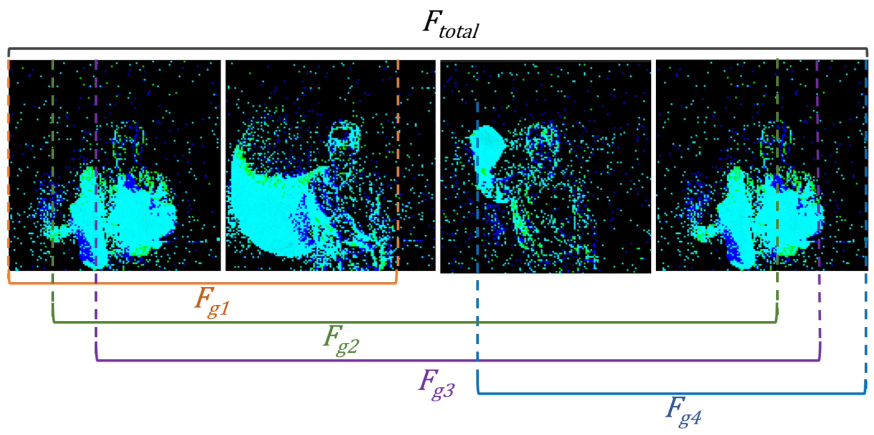

Fig. 5.1: Example gesture chain with variable F_g duration ($\alpha_1 = 0.2$ and $\alpha_2 = 1$). The coloured underscores represent, for each gesture in the chain, the temporal window in which they could appear given the values of α_1 and α_2. This allows to understand why the gesture transition is not predictable and most time-steps have no guarantees of belonging to a certain position in the chain.

subjects are split in the Training and Testing sets following the original DVS-Gesture split, therefore, users appearing in the test set do not appear in the training set.

When building the gesture chains, having a constant number of frames for each gesture can also allow the machine to know when the transition will happen. To solve this, the duration in frames of each gesture F_g is made variable. Let F be the initial number of frames that the gesture sequences have, and F_{total} the target number of frames for the final gesture chains, which is user defined. Then, as seen in equation (5.4), the duration of each gesture F_g will be a fraction of F (parameterised by the coefficients α_1 and α_2) which satisfies that the total sum is equal to F_{total}.

$$F_g \in [\alpha_1 F, \alpha_2 F] \mid \sum_{g=1}^{L} F_g = F_{total} \tag{5.4}$$

Then, the value for F_g is chosen randomly from the set that was just defined, and the resampling from F to F_g is carried out by taking the first F_g frames of the original sequence (Fig. 5.1 demonstrates this variability, visually). In the experiments, two datasets are designed with $\alpha_1 = 0.5$, $\alpha_2 = 0.7$ and one with $\alpha_1 = 0.2$, $\alpha_2 = 1$. Both work well when using the sequences in DVS-Gesture because each gesture is repeated several times per recording, and therefore it is recognisable even after discarding a substantial part of the sequence.

Finally, when targeting a specific F_{total}, the number of initial frames F that will allow the values of F_g to have a uniform distribution between $\alpha_1 F$ and $\alpha_2 F$

is given by:

$$F = \frac{F_{total}}{L} \frac{2}{\alpha_1 + \alpha_2} \tag{5.5}$$

To further clarify the generation of chains of variable gesture lenght, Algorithm 1 provides pseudo-code for an implementation of equation 5.4, which returns gesture chains.

Algorithm 1 Implementation of equation 5.4

```
1:  min_len ← $F \times \alpha_1$
2:  max_len ← $F \times \alpha_2$
3:  previous_len ← 0
4:  frames ← []                                  ▷ Initialize list to store frames for each gesture
5:  for $g = 0$ to $L - 1$ do                      ▷ Iterate over each gesture in the sequence
6:      if $g = L - 1$ then                        ▷ If last gesture, complete remaining frames
7:          current_len ← $F_{\text{total}}$ − previous_len
8:      else
9:          low_limit ← $F_{\text{total}}$ − previous_len − $(L - 1 - g) \times$ max_len
10:         high_limit ← $F_{\text{total}}$ − previous_len − $(L - 1 - g) \times$ min_len
11:         current_len ← random.uniform(max(min_len, low_limit), min(max_len, high_limit))
12:     end if
13:     current_frames ← Frame sequence of gesture $g$
14:     frames.append(current_frames[0 , current_len])
15:     previous_len ← previous_len + current_len
16: end for
17: save frames to output file
```

5.3 Results

In this section, it is first proven how a network without the capability for temporal feature extraction (ANN-BN) can solve the classification task in DVS-Gesture but fails to do so in the new DVS-GC, which demands a perception of temporal order. Then it is demonstrated how, in contrast, an SNN of the same architecture learns to perceive the temporal dependencies in DVS-GC. From there, the effects of the membrane potential reset strategy, voltage leak, time-dependent weights, and time-dependent normalization are evaluated. Finally, SNNs are compared to conventional RNNs.

5.3.1 Experimental setup: Neural network architectures

For the experiments, the S-ResNet developed in Chapter 4 is used, which in its initial form uses a LIF neuron model, reset by subtraction and BNTT.

As seen in equation (5.6), for a time-dependent input of d dimensions $x_t = (x_{1,t}...x_{d,t})$, the method defines an individual BN module per time-step. This not only normalizes each feature k (or convolutional channel in the case of CNNs) independently, as regular BN would do, but also defines independent statistics (mean $\mu_{k,t}$ and standard deviation $\sigma_{k,t}$) and learnable weights ($\gamma_{k,t}$ and $\beta_{k,t}$) per time-step t.

$$BNTT(x_{k,t}) = \gamma_{k,t}\frac{x_{k,t} - \mu_{k,t}}{\sqrt{(\sigma_{k,t})^2 + \epsilon}} + \beta_{k,t} \tag{5.6}$$

In order to compare to non-spiking ANNs, a non-spiking version of the same architecture is defined. The neuron model is substituted by the Rectified Linear Unit (ReLU) activation function, and regular BN is used instead of BNTT. With these changes, the network becomes a conventional feed-forward ResNet. These networks process the input instantaneously, without temporal dynamics, which means that, for a sequence classification task such as action recognition, they can give an output per time-step but not a global one for the whole sequence. This was solved by adding the same output layer used by the SNN, which can be seen as a voting system that accumulates the outputs for all time-steps by summing them together. In the following experiments, this network will be referred to as ANN-BN.

Additionally, in order to study the effect of the learnable weights in BNTT, an additional network is defined, the ANN-TW (ANN with temporal weight), which is a modified version of the non-temporal ANN. This version adds a learnable weight $w_{l,t} \in \mathbb{R}^1$ per time-step t at each layer l, which is used to scale the activation map after the convolution (Conv) and BN layers as:

$$y_{l,t} = BN(Conv(x_{l,t})) \cdot w_{l,t} \tag{5.7}$$

As an alternative, a version where each channel learns a different temporal weight is also defined. This network is referred to as ANN-TWC.

Finally, for the RNN vs SNN comparisons in Section 5.3.3, a different architecture is defined with the objective of disentangling temporal processing from spatial processing. A non-spiking ResNet14 acts as spatial feature extractor, then (1 or 2) fully connected ”temporal layers” of 128 features are appended before the final classification layer, to act as temporal feature extractor. As temporal

Tab. 5.1: Test performance on DVS-Gesture. SNN* was initialized with pre-trained weights as proposed in [192]. Training and testing were run for 3 times, accuracies presented as mean ± std.

Network	Normalization	DVS-Gesture Accuracy
SNN	BN	70.31 ± 3.27 %
SNN	BNTT	89.82 ± 1.50%
SNN*	BNTT	94.84 ± 1.06 %
ANN	BN	97.35 ± 0.45 %
ANN	BNTT	96.95 ± 0.61%

layers, the following are tested: feed-forward SNN layers, recurrent SNN layers (RSNN), vanilla RNN, and LSTM.

5.3.2 DVS-Gesture evaluation

As in Chapter 4, the experiments are performed by only evaluating the test set after the training is complete, without using test evaluations to tune the training.

Table 5.1 shows how both SNN and ANN achieve high accuracy in the DVS-Gestures task. As previously stated, the ANN final prediction is just a sum of the individual predictions made at each time-step. Each of these is made using the information from a frame which integrates the events received within a time window; for this set of experiments, the time window is $\frac{1}{50}$ of the total number of events for each frame.

The ANN has no way of combining information from different frames and has no notion of the timing in which they occurred, hence, it does not perceive the timing or relative order of the events. Still, it cannot be said that the features it uses are strictly non-temporal. When accumulating these events into frames, only the ones which are close in time will be integrated into the same frame, meaning that the spatial features the network will calculate are still dependent on event timing. This makes the appropriate wording a sensible matter: the ANN does not implement working memory neither does it implement temporal feature extraction; still, the spatial features it extracts in this scenario are dependent on event timing, therefore, given that a temporal feature is any attribute of the data that is explicitly related to time, these features can be considered a type of temporal feature. This explains why a network without temporal feature extraction can solve the DVS-Gestures task, and how solving this task does not require to perceive the temporal ordering of events, but only to integrate events which are close in time so that spatial features become apparent. Then, a system designed for the classification of static images can perform the task.

For completeness, the table also reports the accuracy obtained by the SNN with conventional BN (SNN-BN) and the accuracy of the ANN with BNTT (ANN-BNTT). It can be seen how the ANN does not benefit from the time-dependent computations of BNTT and obtains a very similar result. On the contrary, the SNN performance decreases when using regular BN, demonstrating how, for a system where activity statistics change through time such as SNN, timing-aware normalization is beneficial.

5.3.3 DVS-Gesture-Chain evaluation

Using the methodology described in the previous Section 5.2, three DVS-GC datasets were created, as summarised in Table 5.2. Datasets *81-p* and *96-p* define a smaller variability for the duration F_g of each individual gesture ($\alpha_1 = 0.5$, $\alpha_2 = 0.7$), while *96-u* defines a larger one, making it much harder to predict the transition between gestures in time (Fig. 5.1 demonstrates this variability visually).

All three datasets are defined with a validation set of 20% of the training data which is evaluated at every epoch. The test performance is then evaluated using the weights with the highest validation accuracy.

Tab. 5.2: Parameters per dataset. In the naming convention, **-p** stands for predictable time windows while **-u** stands for unpredictable time windows.

Name	**N**	**L**	α_1	α_2	**Repetition**	**# classes**
81-p	3	4	0.5	0.7	Yes (5.2)	81
96-p	3	6	0.5	0.7	No (5.3)	96
96-u	3	6	0.2	1	No (5.3)	96

ANN vs SNN and time-dependent weights

First, the networks are run on the *81-p* and *96-p* datasets. As seen in Table 5.3, now that the task requires distinguishing the ordering of the events, the non-temporal ANN (ANN-BN) fails to solve it. Moreover, its accuracy value implicitly reveals the computations performed by the network: taking the 81 class data set as an example, one can see how the ANN accuracy (16.91%±0.5) is higher than random chance (1.23%). This is because the network is capable of detecting the gestures present in the sequence and the number of times they appear, but is unable to perceive their ordering. With such conditions, and assuming a perfect accuracy in gesture detection, the probability of correctly classifying a sequence

Tab. 5.3: Test performance on DVS-GC. Training and testing were run for 3 times, accuracies presented as mean ± std.

Network	Normalization	Dataset	Accuracy
ANN	BN	81-p	16.91 ± 0.50 %
ANN	BNTT	81-p	99.52 ± 0.31 %
ANN	BN + TW	81-p	89.44 ± 5.74 %
ANN	BN + TWC	81-p	91.08 ± 8.10 %
SNN	BN	81-p	86.00 ± 2.38 %
SNN	BNTT	81-p	95.83 ± 0.62 %
ANN	BN	96-p	12.96 ± 0.74 %
ANN	BNTT	96-p	99.52 ± 0.55 %
SNN	BN	96-p	80.62 ± 2.76 %
SNN	BNTT	96-p	96.32 ± 0.02 %
ANN	BNTT	96-u	74.66 ± 0.64 %
SNN	BNTT	96-u	91.16 ± 1.30 %

for the 81 class dataset is $p_d = 16.05\%$. This probability can be calculated as follows:

Assuming a system with perfect gesture classification but no perception of order that is evaluated in DVS-GC, this system will know which gestures are present in the sequence and the number of times they appear, but will be unable to perceive their ordering. For this system, depending on the number of detected gestures, the candidates to be the correct output are reduced. Therefore, to calculate its accuracy, one can calculate the probability of correctly classifying each individual class x_i in the dataset and then, assuming a constant number of class examples, their average will be the final accuracy. For the 81-class DVS-GC dataset, it can be calculated as in equation 5.8.

$$p(x) = \begin{cases} 1, \text{ if gesture repeated 4 times} \\ \frac{1}{4}, \text{ if gesture repeated 3 times} \\ \frac{1}{6}, \text{ if 2 gestures repeated 2 times} \\ \frac{1}{36}, \text{ otherwise} \end{cases}$$

$$P_d = \frac{1}{C}\sum_{i}^{C} p(x_i) =$$
$$= \frac{1}{81}(3 \cdot 1 + 24 \cdot \frac{1}{4} + 18 \cdot \frac{1}{6} + 36 \cdot \frac{1}{36}) = 0.1605 \tag{5.8}$$

On the other hand, the results show how the SNN still achieves high accuracy on these same datasets, implying that its temporal dynamics allow to perceive order in time. In order to analyse which components of the network enable this capacity, the experiments also report the performance of the SNN with conventional BN (SNN-BN) and the accuracy of the ANN with BNTT (ANN-BNTT). The accuracies obtained by both systems indicate that they are successfully learning to perceive order, meaning that both, spiking neurons and BNTT can enable a neural network to recognise temporal sequences on their own. Additionally, it is also worth noticing how, for the *81-p* and *96-p* datasets, the ANN with BNTT is more accurate than the SNN.

In BNTT, the perception of order is gained by learning time-dependent values that are used to scale the activation maps of the network, providing temporal attention. In order to decorrelate this capacity from the normalization strategy, two modified versions of the non-temporal ANN were created, which use regular BN and implement learnable temporal weights, ANN-TW and ANN-TWC (introduced in Section 5.3.1).

The performance of ANN-TW (Table 5.3) proves how a single time-dependent weight per layer is enough to recognise the temporal sequences in *81-p*, and how the learned value does not need to be different between channels for temporal perception purposes. Still, the ANN-TWC obtains a slightly higher accuracy. Apart from that, the performances of both networks are lower than those of the systems using time-dependent normalization statistics, proving how these are not essential but indeed beneficial.

Finally, the performances of the networks with the *96-u* configuration are evaluated, where the variability of the duration F_g of each individual gesture is higher. The results (last two rows of Table 5.3) demonstrate how this set-up greatly decreases the performance of the ANN-BNTT, meaning that temporal attention is not enough for the task. In contrast, SNN-BNTT exhibits a smaller decrease and still solves the task with high accuracy, proving how the capacity of SNNs for spatio-temporal feature extraction goes beyond that of temporal attention. The complete analysis justifying these results is provided in Section 5.4.

Leak and reset mechanism

When using regular BN, the SNN does not have temporal weights, making voltage leak the only time-aware component in the network. This motivates us to explore its relevance to solving the task.

Table 5.4 compares the results of the same network trained with LIF neurons and IF neurons (no leak). Because the performance comparison between LIF and IF can be affected by the chosen leak coefficient, its optimal value was found through hyper-parameter search. The best results are obtained with 0.87 for the 81 class dataset and 0.80 for the 96 class one.

The original network, which uses reset by subtraction, suffers a major performance drop when not equipped with leak. After analysing this behaviour, it was found that the reason behind this is the excess of voltage in neurons reset by subtraction, which can trigger delayed spikes that slow down adaptation to newer inputs. The results table quantifies this effect by means of what was called "repetition error" (R-error). The R-error is measured as the percentage of wrong classifications where at least one of the miss-classified gestures in the chain has been predicted to be the same as the preceding one. As there are three different gestures to choose from at each position in the chain, the standard R-error is 33.3%. Values higher than this one will indicate a tendency towards repeating previous predictions. (Notice that the 96-class dataset does not allow repetition in its classes and therefore cannot present R-error, still, not clearing old voltage also decreases the performance in it)

These results demonstrate how voltage leak prevents old information from corrupting current calculations by solving the voltage stagnation problem caused by the reset by subtraction. Additionally, a reset to zero strategy was also tested as an alternative. Its results demonstrated how this reset mechanism also prevents the issue (Table 5.4), as it does not retain any voltage after spiking, and therefore does not generate delayed spikes.

Interestingly, reset to zero consistently achieves the best performance when paired with IF neurons, while implementing leak decreases its accuracy. Not implementing voltage leak will mean that neurons close to reaching the spiking threshold will remain in that state, even after the stimulus that was triggering them is long finished. This will make them prone to spiking prematurely in later processing, arguably causing noisier computations. On the other hand, leakage represents the progressive loss of the short-term memory of the network, which also has the potential to disrupt computations. The fact that leak is not beneficial for the task at hand might indicate that the former issue is not prevalent. One possible explanation can be that, when a new gesture appears, initial noise in the spiking pattern is still superseded by later detections due to data redundancy through time (the gesture can be continuously detected throughout a window of time).

Tab. 5.4: Test accuracy and R-error of the SNN-BN under different setups. IF neurons do not leak. Zero stands for reset to zero and Sub for subtraction. LIF neurons use a leakage factor of 0.87 except for LIF-Sub in 96-u, which uses 0.80. Training and testing were run for 3 times, accuracies presented as mean ± std.

Neuron	Reset	Dataset	Accuracy	R-error
LIF	Sub	81-p	86 ± 2.38 %	33.54 ± 5.20 %
IF	Sub	81-p	48.27 ± 1.38 %	70.41 ± 2.44 %
LIF	Zero	81-p	82.31 ± 2.14 %	35.30 ± 8.09 %
IF	Zero	81-p	92.59 ± 2.67 %	31.01 ± 2.15 %
LIF	Sub	96-u	68.74 ± 1.45 %	n/a
LIF	Zero	96-u	63.58 ± 3.09 %	n/a
IF	Zero	96-u	71.40 ± 4.55 %	n/a

RNN vs SNN

After demonstrating how SNNs can perform temporal computations without the need for recurrent connections, results are further validated by comparing their performance with that of RNNs. For this, as introduced in Section 5.3.1, a different architecture is defined, which is based on a non-spiking ResNet14, which acts as spatial feature extractor, and (1 or 2) fully connected "temporal layers" (SNN, RSNN, RNN or LSTM) which are added before the final classification layer as temporal feature extractor.

Table 5.5 shows how, for *96-u*, SNNs outperform vanilla RNNs while LSTMs outperform SNNs. The RSNN demonstrates a substantial improvement with respect to the SNN when using two layers, getting closer to the LSTM performance. On the other hand, in the dataset with predictable time windows, *81-p*, all networks perform at a similar level, implying that all networks manage to exploit its predictable time windows, arguably, demonstrating time-dependent feature extraction.

The performance differences between SNN, RSNN, RNN, and LSTM are well justified by their computing principles, which are analysed in Section 5.4.2.

5.4 Analysis of temporal computations

After proving through empirical results how spiking neurons and time-dependent weights enable temporal order perception, in this section the in-depth mechanics that implement this capability are analysed.

Tab. 5.5: Test accuracy in DVS-GC *81-p* and *96-u*. SNNs use IF neurons and reset to zero. Training and testing were run for 3 times, accuracies presented as mean ± std. "# TL" stands for number of temporal layers."# params" presents the number of parameters used in the temporal layers as a factor of the parameters of a 128-dimensional dense layer. *RNN presents the maximum accuracy instead of the mean, as numerous trials failed to learn.

Temporal layers	# TL	# params	Dataset	Accuracy
SNN	1	×1	96-u	61.53 ± 3.27 %
SNN	2	×2	96-u	67.09 ± 1.57 %
RSNN	1	×2	96-u	64.62 ± 0.69 %
RSNN	2	×4	96-u	77.80 ± 1.92 %
RNN	1	×2	96-u	44.08* %
RNN	2	×4	96-u	65.58* %
LSTM	1	×8	96-u	87.31 ± 2.79 %
LSTM	2	×16	96-u	88.80 ± 3.47 %
SNN	2	×2	81-p	84.60 ± 3.08 %
RSNN	2	×4	81-p	86.47 ± 1.44 %
RNN	2	×4	81-p	72.32 ± 16.91 %
LSTM	2	×16	81-p	89.83 ± 3.34 %

5.4.1 Temporal attention analysis

Networks with time-dependent weights such as ANN-TW or those using BNTT use temporal attention to store the time at which a visual detection occurred. This is achieved by constraining the activation of certain layers or channels to a time window, then the feature detected by those neurons will be known to happen within that time-window.

In order to prove how the networks are using this strategy, in Fig. 5.2, the value of the temporal weight in ANN-TW is visualised when trained in the *81-p* dataset. Notice that, when designing DVS-GC, the gesture chaining procedure was made variable in time, so that the transition between gestures does not always happen in the same time-step. Now, visualising the graph, it can be seen how the network learned to reduce the weight in the uncertainty zone of the transition and defined its detection time windows between the time-steps which are guaranteed to belong to the *n-th* gesture. Then, it can be seen how the weight restricts the last layers to only be active in time windows corresponding to specific positions in the 4-gesture chain. This specializes different layers in detecting gestures at certain positions in the chain, acting as a temporal attention coefficient. This is equivalent to associating timestamps to the detected spatial features and then combining this information in the last layer by accumulation. This last layer is the only element in the system implementing memory for the non-spiking networks.

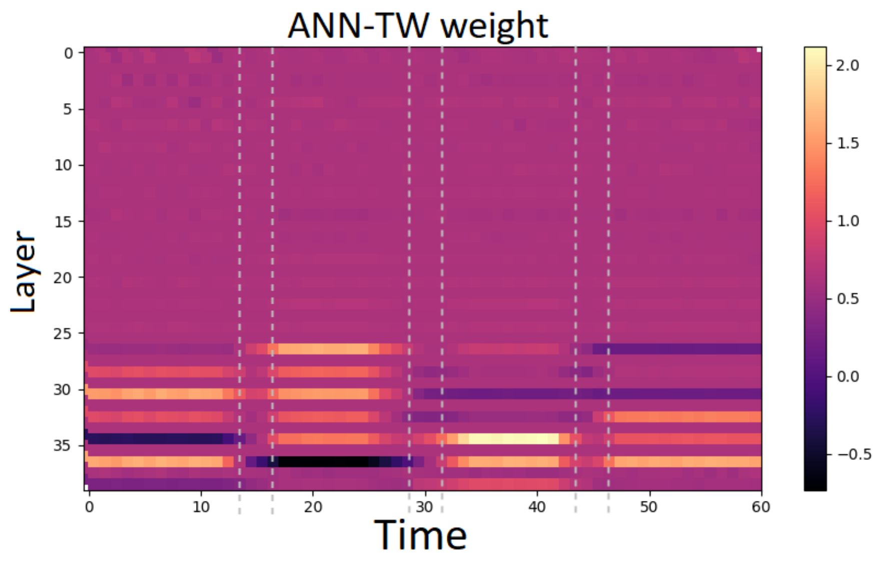

Fig. 5.2: Value of the time weight in TW-ANN. Dotted lines highlight the gesture transition zone. The last two layers of the graph correspond to the layers in the residual connection downsampling. Trained in the *81-p* DVS-GC.

For ANN-BNTT, the same mechanism can be observed. Fig. 5.3 (a) displays the value of BNTT's β weight, which scales neuron activations. Unlike the ANN-TW graph, this one does not show large changes in the coefficient value. This is because the β weight has a different value per channel, something that is not visible in Fig. 5.3 (a), as it averages through channels. Therefore, in order to visualize the temporal windowing of the BNTT weight, Fig. 5.3 (b) plots the center of mass m in the time dimension for the weights at each channel as:

$$m = \frac{1}{T}\sum_{t}^{T}(x_t - min(x))t \tag{5.9}$$

Where x is a vector that contains a weight value x_t per time-step t.

With a uniform distribution of weight through the time-steps, the center of mass would have a value equal to $T/2$, which in the case of our network would be 30. Consequently, all the values in Fig. 5.3 (b) that are far from this number are indicators of the existence of a time window. It can be seen how the centre of mass varies among different channels, demonstrating how they specialise on detecting features inside different temporal windows. This proves how BNTT hard-codes a different temporal attention for each channel in a layer.

Given this computational logic, it is then clear why in the *96-u* dataset the performance of these networks dropped. With $\alpha_1 = 0.2$ and $\alpha_2 = 1$ time-steps are not guaranteed to contain a specific position in the chain (except for the first and last gestures), since the transition zones now overlap. Therefore, in that scenario, time-dependent features calculated using temporal attention are not a reliable descriptor.

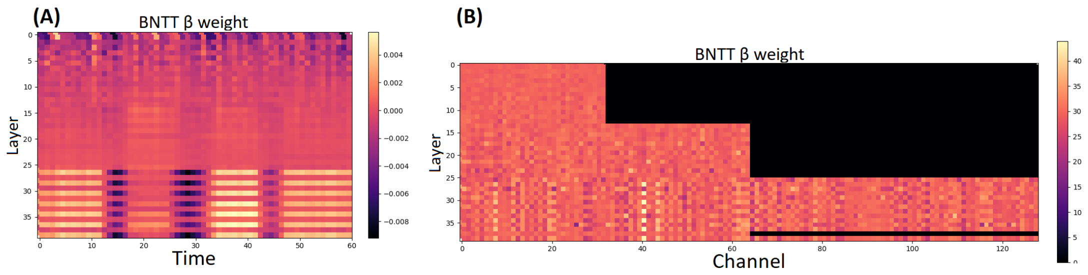

Fig. 5.3: (A): Bias weight average value across channels in the BNTT layers of ANN-BNTT. (B): Value of the center of mass in the time dimension (60 time-steps) of the bias weight of the BNTT layers in ANN-BNTT. The last two layers of all graphs correspond to the layers in the residual connection downsampling. Trained in the 81-class DVS-GC.

5.4.2 Spiking neuron analysis

Unlike temporal attention, spiking neurons achieve high accuracy in all three datasets. This shows how SNN can perform two types of spatio-temporal tasks:

1. Sequence recognition with predictable action time windows in the *81-p* dataset.
2. Sequence recognition with unpredictable time windows in the *96-u* dataset.

Task 1 can be solved by means of time-dependent features, as shown by the analysis performed on temporal weights. Moreover, as this dataset allows repetition, these kinds of features are indispensable in order to distinguish individual gestures when the same one is repeated in succession.

Task 2, to the best of the author's knowledge, can only be solved by recognising each gesture transition in the chain, as the timing of an action is not enough to find its chain location and, therefore, the relative order of appearance is the only usable information.

Following that logic, by performing successfully in Task 2, SNNs demonstrate that they can detect gesture transitions in a time-invariant manner, while in Task 1, they demonstrate the ability to localise gestures in time. Such behaviour implies that spiking neurons enable time-invariant spatio-temporal feature extraction, as well as time-dependent feature extraction.

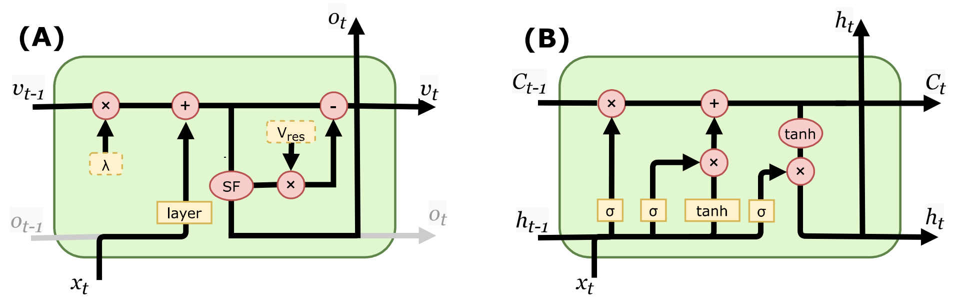

Fig. 5.4: (A): Diagram of a layer of LIF neurons. *layer* are the synaptic weights, *SF* the spiking function, V_{res} the voltage reset value. Gray lines show the architecture with recurrent connections, without them, the architecture is feed-forward. (B): LSTM diagram. C_t is the cell state, h_t the hidden state and output, the yellow *tanh* is a layer of synaptic weights with Hyperbolic Tangent activation. σ stands for the gating layer with Sigmoid activation.

Comparing SNN to RNN

A layer of spiking neurons can be seen as a recurrent cell where the current input x_t passes through a layer and is summed to the previous state v_{t-1}. The contribution of v_{t-1} is weighted by the leak factor, and the final voltage v_t is decreased by subtracting voltage in the event of a spike. Therefore, they retain memory by integrating inputs over time until the threshold is surpassed. Then, when the information is released in form of a spike, it is deleted from memory through the reset mechanism. On the contrary, in a non-spiking vanilla RNN, the neuron itself has no memory, recurrency is implemented only at layer level by defining a recurrent synapse from each neuron to itself and lateral synapses within the layer. Therefore, it is to be expected for SNN/RSNN to have better performance than RNNs.

On the other hand, LSTM adds a cell state to their computations to integrate inputs through time, like spiking neurons do, but, as seen in Fig. 5.4, it has three differences: first, the integration is weighted by two gating layers, the input and forget gates, while LIF neurons compute which information to forget through the reset mechanism and the leak factor; second, in LSTM the non-linearity is applied before integration, while the spiking neuron applies its non-linearity (thresholding) only to the output; finally, in an LSTM the output is weighted by another gating layer. This side by side comparison illustrates how spiking neurons define a computing principle similar to LSTM units, but without the use of gating layers, resulting in a lighter network. Additionally, it allows to understand how, as seen in the experiments, an internal state can be enough to calculate temporal features. Recurrent connections can be beneficial, but they are not indispensable.

5.5 Conclusions

This chapter showed how spiking neurons can be exploited to solve temporal tasks without the need of recurrent synapses. This proves how their temporal dynamics are not only a vehicle for computational efficiency, but also a tool for the extraction of temporal features. This can allow to bypass the need for recurrent connections when a lighter network is needed and to reuse feed-forward networks for temporal tasks. Moreover, the parallelism drawn between LSTM and SNN allows to appreciate how SNN computation is closer to LSTM than to vanilla RNNs. Understanding their similarities and differences allows to make informed choices when designing temporal processing systems and paves the way to distilling more biologically inspired principles into machine learning. Additionally, it also contributes to closing the gap between neuroscience and machine learning knowledge.

The performed experiments evaluated the two components that allow an SNN to clean its memory, the leak and reset mechanism. The effect of the leak factor has been previously evaluated in static data [193], but understanding its relevance for temporal computations was still necessary. The obtained results contribute to develop this understanding by showing how voltage leak prevents old information from stagnating in the network when using reset by subtraction. Looking at the reset strategy, it was found that zero-reset also solves the aforementioned problem. Reset by subtraction has been a popular option given that it prevents loss of information, and has proven especially useful in ANN to SNN conversion approaches [194]. Still, results indicate that retaining such information can come at the cost of slower adaptation to dynamic inputs. Therefore, this effect should be taken into account when designing SNNs for temporal processing tasks, and appropriately handling it will lead to improved results, as shown in the experiments.

Additionally, the analysis of temporal weights demonstrated a clear use case for temporal attention, showing how time-dependent features are learned by a network when the meaning of the events is dictated by their timing. In this work, the implementation of temporal weights and time-dependent normalization requires learning a parameter per time-step, which would be a limitation for inputs of variable length. Still, it is enough to demonstrate the aforementioned computing principle. Moreover, it serves as a tool to prove which tasks can be solved with time-dependent features without the need for time-invariant ones.

These insights were obtained thanks to the newly proposed DVS-GC, a task which was created by means of a novel chaining technique. The relevance of this task is that, first, it fulfils the current need for event-based action recognition datasets. Apart from that, it provides an approach that allows the creation of controlled scenarios in order to evaluate specific capacities of a learning system. The datasets built in this work serve as examples, where *81-p* and *96-p* could be solved by timing-aware features, and *96-u* could only be solved with time-invariant spatio-temporal features. Moreover, the chains can be made arbitrarily long, which allows to test the limits of a system's memory. Looking ahead, the results provided in the proposed DVS-GC configurations can serve as a baseline when evaluating new systems. Finally, if a more challenging task is needed, the method allows to build longer sequences with more gestures.

Chapter 6

Towards scalable algorithms for continual learning

6.1 Introduction

As previously discussed, achieving life-long continual learning is a major priority for AI. This requires the avoidance of the catastrophic forgetting problem through scalable and efficient methods, like the human brain does [195, 196]. In recent years, multiple methods have been developed to alleviate CF in incremental learning setups. Significant success has been achieved by storing samples of previously seen data in exemplar memories, and through the use of parameter isolation methods when the task-id is known. However, the requirement for exemplar memories or knowledge of the task-id result in severe limitations regarding scalability, privacy preservation and deployment. It is arguable that more efficient and sustainable solutions can be achieved, given that biological brains are known to retain knowledge seamless through life, while allowing for continuous learning. Therefore, it is of great interest to explore the feasibility of alternative AI solutions which bypass the aforementioned limitations, aiming at scalable algorithms that can lead to robust life-long learning.

To this end, this chapter proposes a method that avoids requirements which hamper scalability in incremental learning scenarios, such as exemplar memories or task-id knowledge. This method is tested on conventional ANNs, as opposed to the previous two chapters which employed SNNs. This was necessary to make results comparable to other works in continual learning, which also use non-spiking ANNs, while it also allowed to avoid the use of BPTT making training much faster and allowing for a larger volume of experiments.

At a high level, the strucuture of biological brains can be seen as a set of

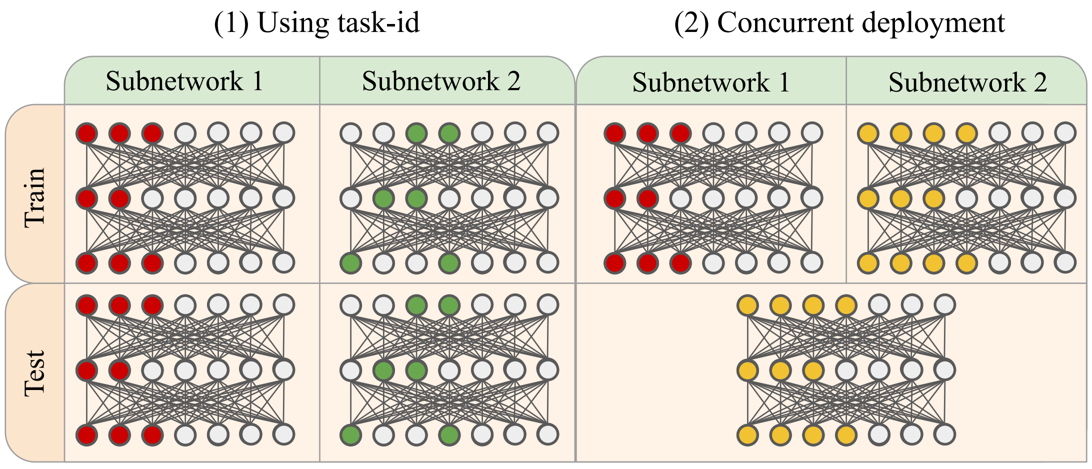

Fig. 6.1: Parameter isolation when using task-id vs the proposed concurrent subnetwork deployment. (1) Requires task-id at test time to deploy the correct subnetwork depending on the task at hand, (2) deploys a subnetwork that contains both, regardless of the task, hence ignoring task-id.

collaborating subnetworks, each specialised in different functions [19]. Such modular behaviour facilitates the preservation of consolidated knowledge, as one can choose to update only the relevant sub-system when training on new data. In the AI domain, parameter isolation methods to CL (Section 3.2.4) take this same approach, defining independent or semi-independent systems to solve each task. Still, as previously explained, these methods usually rely on knowledge of the task-id during inference time. A research direction of interest is then to enable parameter isolation methods to work without task-id or exemplars, while also bypassing the aforementioned requirements. To this end, this chapter proposes a new method: *Low Interference Feature Extraction subnetworks* (LIFES). The proposed algorithm trains subnetworks within a neural network of fixed capacity. Then, at inference time, it deploys the full network without selecting any specific subparts, hence executing all subnetworks concurrently with a single forward path (Figure 6.1).

Deploying all subnetworks concurrently represents the most ambitious scenario for parameter isolation, as it is optimal in terms of requirements and computational cost, but faces the challenge of interference between subnetworks. Therefore, the focus of this work is to study the implications of this challenge and to propose solutions that make it a viable alternative to current CL strategies. The proposed LIFES method (Figure 6.2), prevents weight overwriting, but the concurrent activation of subnetworks causes weight interference and representational overlap. To mitigate these, the following work in this chapter focuses on the feature extractor and proposes the use of Lateral Classifiers Regularisation

(LCR) and weight standarisation for the alleviation of representational overlap, and subnetwork Interference Connection Pruning (SICP) for weight interference. These components are tested through multiple ablation studies, and the final method is compared to relevant strategies in the literature [124, 129, 146, 156], reproducing them under a common framework.

Results show how LIFES achieves competitive results in class-incremental sequences on CIFAR100 [98] and Tiny-ImageNet [197], demonstrating how concurrent subnetwork deployment can be a viable alternative to task selection. Additionally, it provides solutions that alleviate two specific components of the CF problem, representational overlap and weight interference, allowing to extend them to other methods in the future, and contributing to a better understanding of the challenges that need to be addressed in task-agnostic continual learning.

6.2 The LIFES algorithm

The newly developed LIFES algorithm extends the parameter isolation approach to task-agnostic inference by learning subnetworks within a neural network model and deploying them concurrently during inference (Section 6.2.1). However, by doing so, two challenges arise: weight interference and representational overlap. These other catastrophic forgetting components are tackled by reducing subnetwork interference (Section 6.2.2), introducing weight standardization (Section 6.2.3), and through the novel lateral classifier regularization (Section 6.2.4).

LIFES is designed to have minimal requirements, prioritising computing efficiency and its adaptability to multiple incremental learning scenarios. Table 6.1 compares its properties and requirements to the most related approaches in literature. This summary clarifies how LIFES is the most flexible parameter isolation method in terms of requirements, as it does not rely on task-id for inference, exemplars, frozen backbones or extra compute during inference. This puts LIFES at the level of regularisation approaches in terms of requirements, while still being a parameter isolation method.

As presented in Section 3.1, Incremental Learning challenges will be used in order to evaluate continual learning capabilities, where the system will learn a sequence of tasks, one at a time, without access to data from previous or future ones.

Tab. 6.1: Method comparison on different requirements for incremental learning.

	Type	Rehearsal	Task-id	Frozen	Δ params	Δ compute
EWC [124], **MAS** [125]	Regulariz.	No	No[1]	No	No	No
LFL [198], **LWF** [129]	Regulariz.	No	No	No	No	No
iCaRL [122], **EEIL** [199]	Regulariz.	Yes	No	No	No	No
PASS [142], **SSRE** [143]	Proto + Reg.	No	No	No	No	No
FeTrIL[146]	Proto	No	No	Yes	No	No
HAT[156], **TFM**[158]	Param isol.	No	Yes	No	Masks (feat.)	No
PackNet[157], **WSN**[159]	Param isol.	No	Yes	No	Masks (weight)	No
ROW [161], **MORE**[160]	Param isol.	Yes	No	No	Masks (feat.)	OoD classifier
SupSup [162]	Param isol.	No	No[2]	Yes	Masks (weight)	Yes
LIFES (this work)	Param isol.	No	No	No[3]	No	No

[1]Original proposed for task-aware, but commonly adapted to task-agnostic.
[2]Proposes versions of the method for task-aware and task-agnostic.
[3]The best performing version freezes the first three convolutional layers to increase stability.

6.2.1 Concurrent subnetworks

Similarly to other task-aware mask-based approaches [156, 159, 200, 201], LIFES learns a subnetwork for each task t and layer l by defining a mask m_l^t over the activations a_l^t. The mask acts as a gating mechanism, defining which activations are inhibited or active via $\tilde{a}_l = a_l \odot m_l$. These are learnt during training as in [156] by passing an embedding e_l through an anhealed sigmoid function σ following $m_l = \sigma(s \cdot e_l)$. The scaling factor s is calculated as a function of the N total number of training epochs for the task, the current epoch n, and a maximum scaling value $s_{\max}$ following $s_n = (n \cdot s_{\max})/N$.

Previous methods such as HAT [156] use the task-id t to select and apply m_l^t when solving a task, both in training and evaluation. On the contrary, this work proposes to use a cumulative mask $m_l^{\leq t}$ (for training and evaluation) which combines the neurons activated by the current embedding and the ones active in previously learnt masks:

$$m_l^{\leq t} = \max(m_l^t, m_l^{\leq t-1}), \tag{6.1}$$

Afterwards, both $m_l^{\leq t-1}$ and m_l^t can be discarded. After all training is complete, evaluation can be performed without the need to store any extra mask since m_l^t can be used to prune the weights assigned to inhibited activations. Therefore, the method is not affected by the growing memory problem. The only increase in memory is the (strictly necessary) classifier weights, which need to be added in any class-incremental method when new classes appear.

During training, to prevent subnetworks from claiming all activations, a mask capacity regularisation term is introduced, promoting sparsity and inhibition of activations within the subnetwork. The cost of this regularisation is defined as:

$$\mathcal{L}_{\text{M}} = \frac{\sum_{l=1}^{L-1}\sum_{i=1}^{N_l} m^t_{l,i}\left(1 - m^{<t}_{l,i}\right)}{\sum_{l=1}^{L-1}\sum_{i=1}^{N_l}(1 - m^{<t}_{l,i}})\,, \tag{6.2}$$

adding all mask values across all activations i and layers l. The previously unused activation masks $(1-m^{<t}_{l,i})$ are used to nullify the cost of reusing previously trained activations.

Therefore, by learning a sparse mask with each new task and adding them to a cumulative mask, one obtains a parameter isolation strategy that allows to learn a subnetwork per task while being able to evaluate without the need of the task-id.

6.2.2 Interference Connections

Parameter isolation methods prevent weight overwriting by constraining the gradient ∇_l of the weights which are relevant to previous subnetworks. Usually, a mask per task is stored, requiring the task-id to know which one to apply [156–158]. Therefore, subnetworks learnt for later tasks will not interfere with the ones for older tasks. Thus, these connections are not frozen, and the gradient constraint is defined by:

$$\nabla'_{l,ij} = \left(1 - \min\left(m^{<t}_{l-1,i}, m^{<t}_{l,j}\right)\right)\nabla_{l,ij}\,, \tag{6.3}$$

where i is the neuron index at layer $l-1$ and j the one for layer l. In contrast, the experiments in this work (Section 6.3.3) show how when multiple subnetworks are active at the same time, these connections can cause forgetting. This is because the subnetwork from task t will train with $m^{<t}$, but after training for later tasks $t+n$, the cumulative mask $m^{<t+n}$ will activate extra neurons, which could modify the activations j in $m^{<t}_l$ if $i \notin m^{<t}_{l-1} \wedge i \in m^{<t+n}_{l-1}$. This results in *future subnetwork interference*, which represents part of the weight interference component defined in Section 3.1. With LIFES, the remaining weight interference is caused by the activations which at task t were inhibited $j \notin m^{<t}_l$, but are active in the later mask $j \in m^{<t+n}_l$. This second part cannot be easily prevented, since it is still necessary to allow the training of new subnetworks. However, future subnetwork interference can be removed by modifying the gradient constraint so that the aforementioned connections are frozen:

$$\nabla'_{l,ij} = (1 - m^{<t}_{l,j})\nabla_{l,ij}\,. \tag{6.4}$$

Moreover, since weights will still have non-zero values from the network initialisation, the proposed algorithm sets them to zero so that future pre-synaptic activations i do not affect older subnetworks. This process will be referred to as *subnetwork Interference Connection Pruning* (SICP). An intuitive representation of the effect of this constraint can be found in the coloured weights W_L in Fig. 6.2. where the dashed black arrow indicates the weights pruned by SICP (which would have been trainable in other parameter isolation approaches).

6.2.3 Weight standardisation

Weight standardisation (WS) [42] was proposed as an alternative to Batch Normalisation [40] to enable training with smaller batch sizes. For each layer activation, WS forces the weights connected to it to be normal. This provides two benefits, smoothing of the loss landscape and avoiding the creation of elimination singularities (i.e. weights/convolutional kernels under constant inhibition due to being mapped to the null gradient region of the non-linearity). Moreover, learning normalized layers promotes a more controlled signal scaling [41].

Learning feature representations under this constraint can influence the representational overlap between subnetworks in methods such as the one in this work, as shown in Section 6.3.4. Experimental results were obtained with different WS setups. As a result of this analysis, LIFES implements WS for its frozen layers, but not in the trainable ones.

The implementation of WS in frozen layers follows the original one [42]. Let W be the original weights of a given layer, μ_W its mean, and σ_W its standard deviation (both calculated across the fan-in neurons). Then, the standardized weights $\hat{W}_{i,j}$ are calculated as:

$$\hat{W}_{i,j} = \frac{W_{i,j} - \mu_{W_{\cdot,j}}}{\sigma_{W_{\cdot,j}}}\,. \tag{6.5}$$

For trainable layers, the process is adapted to suit the proposed incremental learning scenarios. For weights $W_{i,j}$ where the connection from i to j has been frozen to preserve performance in a previous task, applying equation (6.5) would cause forgetting, as the calculation of $\mu_{W_{i,\cdot}}$ and $\sigma_{W_{i,\cdot}}$ depends on all j, and some of this weights are not yet frozen and will change in later tasks. Therefore, the weights of the frozen subnetwork would be modified.

To correct this, the weights that belong to the current subnetwork are standarised only with respect to themselves, by masking $W_{i,j}$ with $m^t_{l,j}$ and calculating the mean and the standard deviation only with respect to the weights selected by this mask ($j \in m^t_{l,j}$). Weights not belonging to the subnetwork defined by $m^t_{l,j}$ are not standardized during task t training:

$$\hat{W}_{i,j} = (m^t_{l,j} \frac{(W_{i,j} - \mu_{W_{\cdot,j \in m^t_{l,j}}})}{\sigma_{W_{\cdot,j \in m^t_{l,j}}}} + (1 - m^t_{l,j}) W_{i,j}) \tag{6.6}$$

Then, after training, weights in the subnetwork defined by $m^t_{l,j}$ are frozen with standardisation applied to them. Therefore, when equation (6.6) is applied, weights which are not in $m^t_{l,j}$, but are still active because they are in $m^{\leq t}_{l,j}$, will have mean 0 and standard deviation 1, and therefore the whole $\hat{W}_{i,j}$ will too.

6.2.4 Lateral Classifier Regularisation

The main contribution of this work towards the mitigation of representational overlap is the *Lateral Classifiers Regularisation* (LCR). When minimising the loss for a set of classes, the solution space will contain different distributions for the feature representations. All of them maximise separability between the current classes being learned, however, separability with respect to classes from other tasks is not taken into account in the loss minimisation. Still, some solutions might be more prone than others to overlapping with the representations learnt for classes in other tasks.

In order to learn a distribution for current classes with lower probability of overlapping with future ones, the distribution is enforced to have more discriminative features across all layers by adding a lateral classifier to each of them. This classifier is as a linear layer ϕ_l which is only used during training. Then, given the ground truth labels y, each of these classifiers are learned through the regularisation loss $\mathcal{L}_{\text{LCR}}$, which is an average of the cross-entropy loss $\mathcal{L}_{\text{CE}}$ across all lateral classifiers:

$$\mathcal{L}_{\text{LCR}} = \frac{1}{L} \sum_{l=1}^{L} \mathcal{L}_{\text{CE}}(\phi_l \cdot a^t_l, y)\,. \tag{6.7}$$

This promotes more discriminative feature representations even in earlier layers. Intuitively, earlier layers might learn more general features, which are used to build the representations of multiple classes. With LCR, these layers will also be forced to have some class-specific neurons to aid the lateral classifier, therefore creating a more separable representation.

The results in Section 6.3.1 and 6.3.2 show how the representations learnt by LCR considerably reduce representational overlap, providing the same inter-class distance as the upper bound (joint training), and even higher in the experiments in Section 6.3.3.

6.2.5 Complete LIFES method

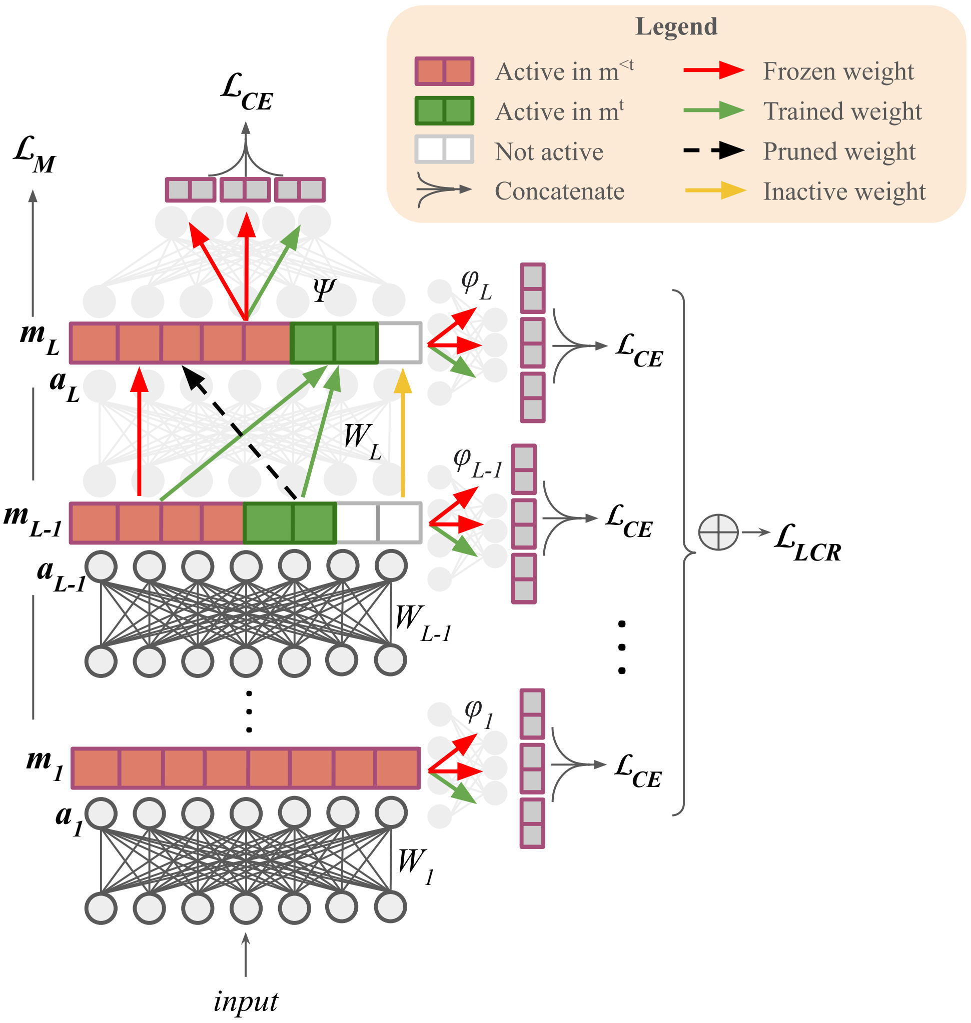

Fig. 6.2: LIFES diagram. Layer activations a_i are masked by $m_l^{<t}$ and m_l^t, and weights which could modify values masked by $m_l^{<t}$ are frozen. Weights W_L are coloured to demonstrate this, which is the result of the gradient restriction and SICP. Classifier weights (ψ and ϕ_l) are divided into groups (or classifier heads) for each set of classes corresponding to a task. Colouring in the classifier weights indicates that only the weights of the classes being learnt are modified.

The complete LIFES algorithm implements the previously explained concurrent subnetwork strategy with SICP and LCR. Then, its loss is comprised of three components, the cross-entropy loss ($\mathcal{L}_{CE}$), the mask capacity regularisation ($\mathcal{L}_M$) and the LCR regularisation ($\mathcal{L}_{LCR}$), which use λ hyper-parameters to balance their weight:

$$\mathcal{L} = \mathcal{L}_{CE} + \lambda_M \mathcal{L}_M + \lambda_{LCR} \mathcal{L}_{LCR} \tag{6.8}$$

Regarding WS, as justified by the results in Section 6.3.4, it applies it only to the first three convolutional layers. These three layers are frozen after training the first task, as the basic shapes they learn are general features which can be reused for later tasks [147, 202, 203].

A diagram of the LIFES training algorithm is presented in Figure 6.2, illustrating the process of training submasks and the integration of SICP and LCR into the process. For further clarification, Algorithm 2 depicts the order of operations upon the arrival of a new task. Notice how training $\psi_{o_{c\in t}}$ and $\phi_{o_{c\in t}}$ means that only the weights connected to the logits of the classes being trained ($o_{c\in t}$) are modified.

Algorithm 2 LIFES' incremental learning loop

```
Initialize $m^{<t}$ as all zeros
for each new task $t$ do
    • Add new classes $c \in t$ to the classifiers $\psi$ and $\phi_l$
    • Jointly train $W$, $m^t$, $\psi_{o_{c\in t}}$ and $\phi_{l,o_{c\in t}}$ subject to the gradient constraint (6.4)
    • Aggregate the current mask $m^t$ to $m^{<t}$ (6.1)
    • Prune according to SICP (Section 6.2.2)
end for
```

6.3 Experimental results

For the experiments, class-incremental scenarios are defined as a sequence of T tasks with C classes each, shortened to C×T, using the CIFAR-100 [98] and Tiny-ImageNet [197] datasets (e.g.: CIFAR-100 (10×10) for CIFAR with 10 tasks of 10 classes each). The main network architecture used in these experiments is the one proposed by [156], which adapts an AlexNet architecture for 32×32 inputs. This allows to easily compare performance to EWC [124], LwF [129] and HAT [156], which are classic references for task-aware approaches. Additionally, performance was also tested for the AllCNN [204] and ResNet [20] architectures (Section 6.3.6). Accuracy is reported over all classes at the end of the sequence, in task aware (TAw Acc) and task agnostic (TAg Acc) format. The former only uses class outputs for the current task, while the latter concatenates all class outputs without using the task-id. Task forgetting (Forg) [205] is also evaluated, calculated as the difference in performance with respect to the initial accuracy when the task was learned. Given that this work focuses on task agnostic methods, TAg Acc is the metric used to rank the performance of the methods, while TAw metrics are complementary information. Notice that in cases where the average accuracy is similar but one method has higher forgetting than the other, this indicates that the one with higher forgetting has also higher plasticity, as it obtained higher accuracy in later tasks but lost more on the earlier ones.

Finally, an inter-class distance is proposed, which is calculated by obtaining the average activation per class, calculating the pair-wise cosine distances between all classes, and reporting the mean across all of them. Notice that the standard deviation for inter-class distance is not across the multiple seeds as in the other metrics, instead, it is calculated across all the pair-wise distances between classes, and afterwards, its mean across all seeds is reported. All reported approaches are reproduced under the same framework with fixed initialisation and training batch order, for a fairer comparison.

Training across all methods is performed using stochastic gradient descent for 300 epochs per task, with a momentum of 0.9, learning rate of 0.01, batch size of 256 for CIFAR-100 and 64 for TinyImageNet, and gradient clipping with a threshold of 1000. For method specific parameters, a search was performed (manual grid-search) to find the best hyperparameter configuration: EWC with $\lambda = 500$, LWF with $\lambda = 8, T = 2$, HAT with $\lambda = 0.75$, FeTrIL with the *most-similar-new-class* pseudo-feature generation heuristic and a fully-connected as classification layer. Finally, LIFES uses $\lambda_{\mathrm{M}} = 1$ and $\lambda_{\mathrm{LCR}} = 1$. The performance of methods from the literature are consistent with the ones obtained in the original papers and previous surveys such as [206].

6.3.1 State-of-the-art comparison

To contextualize the performance of the tested methods, three baselines are provided: fine tuning, which sequentially trains each task without any CF prevention; freezing, which freezes the network backbone after training the first task, and then just trains classifier heads; and joint, which adds the data from previously seen tasks to the current task, training in i.i.d. format and therefore defining the upper bound. Continual learning methods will prove useful for a specific setup if their performance in it is superior to fine tuning and freezing, while joint serves as reference for the maximum achievable performance. Additionally, Table 6.2 and

Tab. 6.2: Results for CIFAR-100 (10×10) and CIFAR-100 (50×1 + 10×5) presented as mean (± standard deviation) across 5 random seeds.

			CIFAR-100 10×10		
Approach	**TAw Acc ↑**	**TAg Acc ↑**	**TAw Forg ↓**	**TAg Forg ↓**	**Inter-class dist ↑**
Finetuning	53.16 (± 1.88)	13.56 (± 1.14)	29.68 (± 2.30)	67.60 (± 1.31)	0.41 (± 0.18)
Freezing	63.46 (± 0.60)	21.22 (± 0.76)	0.00 (± 0.00)	14.92 (± 0.61)	0.27 (± 0.18)
Joint	83.50 (± 0.41)	56.30 (± 0.33)	-0.46 (± 0.36)	5.22 (± 0.19)	0.55 (±0.16)
EWC [124]	60.18 (± 0.86)	17.56 (± 1.26)	20.48 (± 0.92)	56.70 (±1.82)	0.45 (±0.18)
LWF [129]	27.08 (± 0.53)	9.32 (± 0.43)	57.22 (± 0.20)	72.74 (±0.39)	0.15(±0.09)
FeTrIL [146]	51.82 (± 1.59)	17.52 (± 1.11)	-0.34 (± 0.16)	24.16 (± 1.17)	0.29 (±0.17)
HAT [156]	71.30 (± 0.77)	N/A	0.00 (± 0.00)	N/A	N/A
LIFES (Ours)	73.38 (± 0.68)	31.60 (± 0.33)	0.34 (± 0.10)	7.76 (± 0.85)	0.54 (±0.17)

			CIFAR-100 50×1 + 10×5		
Approach	**TAw Acc ↑**	**TAg Acc ↑**	**TAw Forg ↓**	**TAg Forg ↓**	**Inter-class dist ↑**
Finetuning	66.08 (± 1.68)	18.98 (± 1.71)	14.60 (± 1.25)	25.18 (± 3.38)	0.47 (± 0.18)
Freezing	71.98 (± 0.83)	38.30 (± 1.36)	0.00 (± 0.00)	11.16 (± 0.45)	0.47 (± 0.17)
Joint	79.74 (± 0.55)	54.66 (± 1.10)	-0.70 (± 0.20)	0.96 (± 0.45)	0.55 (± 0.16)
EWC [124]	70.50 (± 1.25)	20.86 (± 1.20)	8.62 (± 0.62)	19.82 (± 4.26)	0.51 (± 0.17)
LWF [129]	35.53 (± 1.03)	17.30 (± 0.95)	50.23 (± 0.93)	69.13 (± 0.73)	0.19 (± 0.11)
FeTrIL [146]	67.72 (± 0.83)	36.16 (± 1.41)	-0.28 (± 0.37)	15.26 (± 0.67)	0.47 (± 0.17)
HAT [156]	70.54 (± 0.80)	N/A	0.00 (± 0.00)	N/A	N/A
LIFES (Ours)	77.80 (±0.39)	42.78 (±0.72)	0.20 (±0.09)	5.64 (±0.59)	0.55 (±0.16)

Figure 6.3 present the results obtained with other methods when reproduced with the same network and in the same training conditions. EWC is used as representation of regularisation methods, LWF for distillation, FeTrIL for prototype-based and HAT for task aware parameter isolation.

CIFAR-100 results in Table 6.2 show how the proposed method achieves competitive results, surpassing the performance of the reference task-agnostic approaches in the proposed setup. Regarding inter-class distance, LIFES demonstrates the highest amongst the methods tested. It is specially relevant how it matches the one from the joint upper bound, which trains with i.i.d. data. Following sections provide analysis on how this is achieved.

Comparing CIFAR-100 (10×10) to CIFAR-100 (50×1 + 10×5), the main difference is the performance of *freezing*, which is lower in 10×10, having lower average accuracy and less inter-class distance. This is to be expected, as the first task is the only one used to train the feature extractor in the freezing approach and it has less classes in the 10×10 setup. Moreover the task sequence is longer, giving more importance to plasticity.

Results with TinyImageNet are shown in Figure 6.3 as the average task-agnostic accuracy across the test sets of all trained tasks. This is calculated after training each task, reporting the evolution of this average accuracy as more tasks are added to the incremental learning sequence. Taking into account the

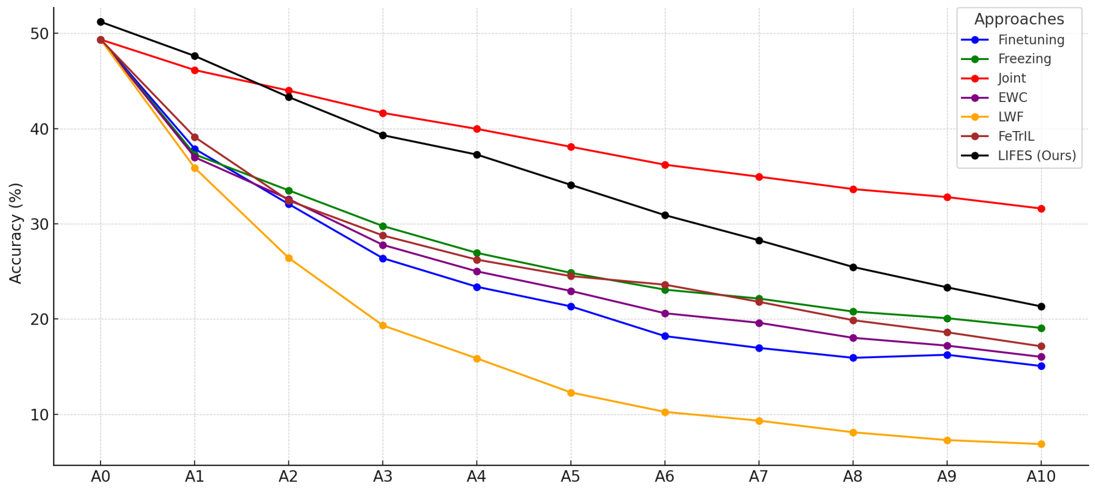

Fig. 6.3: Graphical representation of the average task-agnostic accuracy after each task (x-axis) with Tiny Imagenet (50×1 + 15×10), reshaped to 64×64 images. Results presented as mean across 5 random seeds.

accuracy achieved by the *joint* upper bound, which is considerably lower than the one achieved for CIFAR-100, the results are consistent with previous experiments and LIFES is shown to retain knowledge for longer than the other task-agnostic methods. Apart from that, it is noticeable that LIFES is the only method with a different accuracy than the rest in the first task, this is because of its two regularisation terms, $\mathcal{L}_{\text{M}}$ and $\mathcal{L}_{\text{LCR}}$, which affect how the first task is learnt.

Finally, it is worth noticing that TAw Forgetting is close to 0 in LIFES. This means that when the correct classifier head is chosen (which is equivalent to discarding the classes trained in other tasks) most interference disappears, even when all subnetworks are active concurrently. This shows that, when using the LIFES approach, it is not necessary to mask out subnetworks for irrelevant tasks. Then, the method can be used to reduce the CF problem to classifier head selection. If a reliable method for this selection is proposed, combining it with LIFES becomes a very promising approach.

6.3.2 Lateral classifier regularisation ablation

In order to analyse the influence of the proposed Lateral Classifier Regularisation loss on improving representational overlap, an ablation study is performed where LIFES is evaluated with and without this component (see Table 6.3). Furthermore, the table include results for freezing, the zero-forgetting baseline that represents maximum stability and minimal plasticity by freezing the model (except the classifier) after the first task. Additional measures are included indicating the mean and maximum activation across all non-masked neurons and the sparsity of the last layer. Sparsity is calculated as the percentage of neurons with an activation greater than a threshold. Eight different thresholds are used, calculated as a percentage p of the maximum activation value, where $p \in \{1.0, 2.5, 5.0, 7.5, 10.0, 12.5, 15.0, 20.0\}(\%)$. The reported value is the Area Under the Curve calculated by means of the trapezoidal rule.

The results show a consistent improvement in TAg accuracy when using LCR, with an improvement in plasticity at the cost of a small increase in forgetting. This is likely a result of forcing more discriminative representations in each layer, which makes the inter-class distance higher, therefore reducing representational overlap. This interpretation is further reinforced by the fact that TAw accuracy is not improved by LCR, since representational overlap between classes learnt at different training times is already solved by choosing the correct classifier head and therefore ignoring classes from other tasks. It is also noticeable how LCR achieves almost the same inter-class distance in the 10×10 and 50×1 + 10×5 setups, but without it, the longer 10×10 sequence sees a larger reduction, indicating that LCR trained representations are most useful for longer and more challenging sequences.

The remaining reported values give an idea of the difference in distribution of the feature representations. The mean activation is similar with and without LCR, but sparsity is consistently higher when using it, and so it is the mean maximum activation, indicating a distribution with more salient features.

6.3.3 Interference connection pruning ablation

An ablation study for the subnetwork Interference Connection Pruning component (Section 6.2.2) is also provided. Table 6.4 shows how for the 50×1 + 10×5 sequence the accuracy with and without SICP is very close, while for the 10×10 sequence, the improvement in stability proves most useful, and obtains a higher final accuracy. The LIFES algorithm implements this mechanism in order to prioritise performance in longer sequences and enable scalability.

Interestingly, inter-class distance is higher when SICP is not applied, due to the cost of interference connection pruning being a reduction in capacity and plasticity. The results show higher accuracy for newly learnt tasks when SICP is not used, since their subnetworks have more weights available. However, this comes at the cost of performance degradation in pre-existing subnetworks.

Finally, it is worth noting how the higher inter-class distance of LIFES without SICP is even higher than the one for the joint upper-bound. This indicates that the representations learnt through LCR are even more separable than those learnt

Tab. 6.3: Results for CIFAR-100 (10×10) and CIFAR-100 (50×1 + 10×5). Accuracies, forgetting and statistics for the LCR study. Presented as mean (± standard deviation) across 5 random initialisation seeds.

	CIFAR-100 10×10			
Approach	**TAw Acc ↑**	**TAg Acc ↑**	**TAw Forg ↓**	**TAg Forg ↓**
Freezing	63.64 (± 0.60)	21.22 (± 0.76)	0.00 (± 0.00)	14.92 (± 0.61)
LIFES w/o LCR	73.43 (±0.58)	28.65 (±0.69)	0.20 (±0.00)	5.80 (±1.38)
LIFES with LCR	73.38 (± 0.68)	31.60 (± 0.33)	0.34 (± 0.10)	7.76 (± 0.85)
	Inter-class dist ↑	**Mean**	**Max**	**Sparsity AUC**
Freezing	0.27 (±0.17)	0.04 (±0.00)	0.54 (±0.06)	4.84 (±0.11)
LIFES w/o LCR	0.37 (±0.15)	0.08 (±0.00)	1.51 (±0.10)	3.90(±0.13)
LIFES with LCR	0.54 (±0.17)	0.08 (±0.01)	3.41 (±0.07)	1.89 (±0.08)
	CIFAR-100 50×1 + 10×5			
Approach	**TAw Acc ↑**	**TAg Acc ↑**	**TAw Forg ↓**	**TAg Forg ↓**
Freezing	72.04 (±0.58)	38.08 (±0.41)	0.00 (±0.00)	12.04 (±1.43)
LIFES w/o LCR	77.44 (±0.60)	34.64 (±1.47)	0.14 (±0.08)	3.92 (±0.43)
LIFES with LCR	77.80 (±0.39)	42.78 (±0.72)	0.20 (±0.09)	5.64 (±0.59)
	Inter-class dist ↑	**Mean**	**Max**	**Sparsity AUC**
Freezing	0.47 (±0.18)	0.05 (±0.00)	0.81 (±0.03)	3.98 (±0.17)
LIFES w/o LCR	0.44 (±0.15)	0.09 (±0.00)	1.11 (±0.04)	4.90 (±0.16)
LIFES with LCR	0.55 (±0.16)	0.10 (±0.00)	2.25 (±0.11)	2.94 (±0.20)

when training all classes in the same session, but this effect is reduced when SICP is applied to prevent the aforementioned source of catastrophic forgetting.

6.3.4 Weight standardisation analysis

Weight standardisation limits the solution space of weights to those of mean 0 and std 1, reducing plasticity. In Table 6.5 different combinations of layer freezing are evaluated, showcasing how applying WS to all layers is detrimental, since it causes the task-agnostic accuracy of the first task to be preserved at the cost of having it close to zero for any subsequent tasks. When WS is applied only to the frozen layers, results show consistent improvement, with higher average TAg Acc, lower forgetting, and higher inter-class distance. This seems to indicate that, for layers that are not used to generate task-specific features (due to being frozen), the representations learnt with the WS constraint incur in less representational overlap. Consequently, for LIFES, the first three layers are frozen after learning the first task and WS is applied to them.

Tab. 6.4: Results for CIFAR-100 (10×10) and CIFAR-100 (50×1 + 10×5).

		CIFAR-100 10×10			
Approach	**TAw Acc ↑**	**TAg Acc ↑**	**TAw Forg ↓**	**TAg Forg ↓**	**Inter-class dist ↑**
Joint	83.50 (± 0.41)	56.30 (± 0.33)	-0.46 (± 0.36)	5.22 (± 0.19)	0.55 (±0.16)
LIFES w/o SICP	66.56 (± 1.29)	25.70 (± 1.13)	8.12 (± 0.77)	40.78 (± 1.35)	0.73 (±0.18)
LIFES with SICP	73.38 (± 0.68)	31.60 (± 0.33)	0.34 (± 0.10)	7.76 (± 0.85)	0.54 (±0.17)
		CIFAR-100 50×1 + 10×5			
Approach	**TAw Acc ↑**	**TAg Acc ↑**	**TAw Forg ↓**	**TAg Forg ↓**	**Inter-class dist ↑**
Joint	79.74 (± 0.55)	54.66 (± 1.10)	-0.70 (± 0.20)	0.96 (± 0.45)	0.55 (± 0.16)
LIFES w/o SICP	75.68 (± 0.82)	44.18 (± 0.60)	3.04 (± 0.62)	34.12 (± 0.82)	0.68 (±0.17)
LIFES with SICP	77.80 (±0.39)	42.78 (±0.72)	0.20 (±0.09)	5.64 (±0.59)	0.55 (±0.16)

Tab. 6.5: Results with different weight standardization (WS) configurations for CIFAR-100 (10×10) and CIFAR-100 (50×1 + 10×5) presented as mean (± standard deviation) across 5 random seeds.. Frozen describes which layers are frozen or standardized is indicated with a range of indexes, "All", or "None".

			CIFAR-100 10×10			
WS layers	**Frozen**	**TAw Acc ↑**	**TAg Acc ↑**	**TAw Forg ↓**	**TAg Forg ↓**	**Inter-class dist ↑**
None	[1, 3]	67.58 (± 0.49)	27.16 (± 0.40)	0.34 (± 0.17)	9.14 (± 1.07)	0.51 (± 0.18)
[1, 3]	[1, 3]	73.38 (± 0.68)	31.60 (± 0.33)	0.34 (± 0.10)	7.76 (± 0.85)	0.54 (±0.17)
			CIFAR-100 50×1 + 10×5			
WS layers	**Frozen**	**TAw Acc ↑**	**TAg Acc ↑**	**TAw Forg ↓**	**TAg Forg ↓**	**Inter-class dist ↑**
None	[1, 3]	74.64 (± 0.27)	40.36 (± 0.42)	0.34 (± 0.12)	6.56 (± 0.29)	0.36 (±0.16)
[1, 3]	[1, 3]	77.80 (±0.39)	42.78 (±0.72)	0.20 (±0.09)	5.64 (±0.59)	0.55 (±0.16)
All	[1, 3]	70.44 (± 0.39)	10.90 (± 0.11)	0.12 (± 0.07)	0.12 (± 0.04)	0.36 (±0.16)
None	[1]	70.16 (± 0.71)	32.38 (± 1.29)	0.28 (± 0.07)	8.24 (± 1.19)	0.52 (±0.17)
[1, 3]	[1]	71.22 (± 0.27)	31.58 (± 1.04)	0.18 (± 0.13)	2.98 (± 0.32)	0.45 (±0.16)
All	[1]	65.60 (± 2.77)	13.02 (± 8.64)	-0.02 (± 0.12)	0.70 (± 1.40)	0.43 (±0.18)

6.3.5 Capacity Analysis

When studying representation overlap, it is important to take into account the dimensionality of the representations. It is a well-known fact in High-Dimensional Computing, that the orthogonality between a set of random representations will scale with its dimensionality, and decrease the more representations are placed in its space [207]. Motivated by this observation, this section studies the effect of changing the dimensionality of the last layer of the feature extractor, projecting to the original 2048, and also to 1024 and 4096 (see Table 6.6).

Results show how there is a slight improvement in inter-class distance when dimensionality is increased, but this benefit seems to come from better plasticity, rather than a reduction in forgetting. Specifically, the improvement comes from having more capacity available to allocate into the subnetworks, since later tasks have lower accuracy as the number of non-frozen neurons becomes increasingly smaller. Therefore, growing network capacity does not show a correlation with reducing representational overlap. However, it still shows accuracy improvements,

Tab. 6.6: Performance of the method for different dimensionalities of the last layer for CIFAR-100 (10×10), (50×1 + 10×5), and Tiny Imagenet (50×1 + 15×10) . Last layer dimensionality indicated besides the name. All other layers remain the same.

		CIFAR-100 10×10			
Approach	**TAw Acc ↑**	**TAg Acc ↑**	**TAw Forg ↓**	**TAg Forg ↓**	**Inter-class dist ↑**
LIFES 1024	72.04 (± 0.55)	30.14 (± 0.33)	0.44 (± 0.10)	6.60 (± 0.49)	0.50 (±0.16)
LIFES 2048	73.38 (± 0.68)	31.60 (± 0.33)	0.34 (± 0.10)	7.76 (± 0.85)	0.54 (±0.17)
LIFES 4096	73.46 (± 0.65)	32.52 (± 0.38)	0.22 (± 0.07)	9.40 (± 0.61)	0.55 (±0.17)
		CIFAR-100 50×1 + 10×5			
Approach	**TAw Acc ↑**	**TAg Acc ↑**	**TAw Forg ↓**	**TAg Forg ↓**	**Inter-class dist ↑**
LIFES 1024	77.28 (± 0.35)	40.56 (± 0.36)	0.42 (± 0.15)	4.90 (± 0.32)	0.51 (±0.15)
LIFES 2048	77.80 (±0.39)	42.78 (±0.72)	0.20 (±0.09)	5.64 (±0.59)	0.55 (±0.16)
LIFES 4096	78.18 (± 0.64)	45.32 (± 0.35)	0.20 (± 0.09)	7.00 (± 0.46)	0.58 (±0.16)

Tab. 6.7: LIFES performance for different architectures in CIFAR-100 (50×1 + 10×5). Presented as mean (± standard deviation) across 5 random seeds.

Network	**Parameters**	**TAg Acc ↑**	**TAg Forg ↓**	**Inter-class dist ↑**
AlexNet	6.5M	42.78 (±0.72)	5.64 (±0.59)	0.55 (±0.16)
AllCNN	5.4M	42.62 (± 1.29)	10.62 (± 1.46)	0.32 (± 0.13)
ResNet20	4.8M	39.98 (±2.48)	6.94 (±0.95)	0.31 (±0.12)

as the capacity of the network is fixed, and it eventually saturates when multiple subnetworks have been learnt.

6.3.6 LIFES in alternative architectures

In order to test whether the benefits of LIFES are consistent when applying it to alternative architectures, results were also evaluated for two more networks: AllCNN [204] and ResNet [20]. As a summary, Table 6.7 presents the performance of LIFES in each of these networks along with the parameter count, demonstrating how the method works in different architectures.

LIFES with AllCNN

The fully connected layers in AlexNet architectures account for most of their parameters. Therefore, it is of interest to test whether the masking strategy proposed by LIFES behaves in the same way for networks which are fully convolutional and do not depend on densely connected layers.

To test this, the AllCNN architecture was chosen. Specifically, the configuration of Model C from the original article [204] is used, which employs 8 convolutional layers as feature extractor and a 1×1 convolutional layer followed by

Tab. 6.8: Results with the AllCNN network for CIFAR-100 (10×10) and CIFAR-100 (50×1 + 10×5) presented as mean (± standard deviation) across 5 random seeds.

			CIFAR-100 10×10		
Approach	**TAw Acc ↑**	**TAg Acc ↑**	**TAw Forg ↓**	**TAg Forg ↓**	**Inter-class dist ↑**
Finetuning	49.08 (± 1.67)	13.76 (± 0.46)	28.94 (± 2.07)	54.98 (± 2.00)	0.26 (± 0.01)
Freezing	56.74 (± 1.88)	17.26 (± 0.93)	0.00 (± 0.00)	7.20 (± 1.55)	0.13 (± 0.01)
Joint	82.68 (± 0.43)	54.90 (± 0.76)	0.28 (± 0.57)	5.52 (± 0.97)	0.37 (± 0.01)
EWC [124]	64.56 (± 2.14)	21.36 (± 2.19)	13.54 (± 1.54)	30.98 (± 3.25)	0.23 (± 0.01)
LWF [129]	42.42 (± 0.99)	13.26 (± 0.29)	35.94 (± 0.80)	57.88 (± 2.86)	0.12 (± 0.01)
FeTrIL [146]	51.26 (± 2.30)	18.28 (± 1.46)	0.60 (± 0.44)	19.00 (± 0.61)	0.13 (± 0.01)
HAT [156]	66.46 (± 1.02)	N/A	0.00 (± 0.00)	N/A	N/A
LIFES (Ours)	65.30 (± 1.78)	24.98 (± 0.69)	3.08 (± 0.56)	13.78 (± 1.21)	0.21 (± 0.01)

			CIFAR-100 50×1 + 10×5		
Approach	**TAw Acc ↑**	**TAg Acc ↑**	**TAw Forg ↓**	**TAg Forg ↓**	**Inter-class dist ↑**
Finetuning	58.44 (± 2.15)	22.34 (± 1.26)	19.84 (± 1.43)	37.12 (± 2.72)	0.27 (± 0.01)
Freezing	72.40 (± 1.06)	34.74 (± 1.68)	0.00 (± 0.00)	6.94 (± 1.18)	0.22 (± 0.01)
Joint	78.34 (± 0.67)	54.14 (± 1.16)	0.40 (± 0.71)	2.80 (± 0.98)	0.33 (± 0.01)
EWC [124]	69.52 (± 0.84)	26.00 (± 0.83)	6.52 (± 0.79)	12.82 (± 1.92)	0.26 (± 0.01)
LWF [129]	60.14 (± 1.19)	26.42 (± 0.49)	22.14 (± 0.84)	42.72 (± 2.96)	0.16 (± 0.01)
FeTrIL [146]	67.86 (± 0.89)	37.68 (± 1.39)	0.64 (± 0.50)	16.50 (± 0.71)	0.23 (± 0.01)
HAT [156]	54.20 (± 3.86)	N/A	0.00 (± 0.00)	N/A	N/A
LIFES (Ours)	77.06 (± 0.58)	42.62 (± 1.29)	2.66 (± 0.73)	10.62 (± 1.46)	0.32 (± 0.01)

global pooling as classification layer. To make comparisons fairer, the width of the feature extraction layers was doubled with respect to the dimensionality defined in [204]. This creates a 5.4M parameter network, which is closer to the 6.5M parameters of the AlexNet used for the other experiments.

Table 6.8 presents the results with AllCNN, which are consistent with those obtained with AlexNet in Table 6.2, and considerably close in average accuracy. The most notable differences are LIFES in the 10×10 sequence, where it is 7% more accurate with AlexNet than AllCNN, and LWF in the 50×1 + 10×5 sequence, where it is 9% more accurate with AllCNN than AlexNet. Apart from that, the inter-class distance is lower for all methods with AllCNN, probably due to the lower dimensionality of its last feature extraction layer (384 compared to AlexNet's 2048).

LIFES with ResNets

When designing neural network architectures, it is a well known fact that residual connections are necessary to ease the training of deep feed-forward structures, and therefore they have become the common denominator among deep network architectures [23, 35, 46, 164]. These connections reformulate the problem to learning residual mappings by adding a skip path that sums the input of a block of layers to its ouptut. This makes it easier for the learning algorithm to build identity mappings and prevent depth induced accuracy degradation [20]. Later work [41] has proven that, in order for residual networks to be effective, either Batch Normalisation (BN) [40] or alternative strategies which replicate its benefits (weight standardisation) are necessary. This is because BN makes the contribution of the residual path increasingly smaller through depth compared to the skip path at initialisation. This makes the network effectively shallow at earlier training stages, and allows it to give deeper layers increasingly more influence by increasing the variance in the weights as training goes on. Additionally, it also neutralizes the mean shift activation caused by ReLU, increases the maximum affordable learning rate and acts as an implicit regularizer.

When using these architectures in incremental learning without task-id, implementing strategies such as BN or the WS proposed in NF-ResNets [208] becomes a challenge due to the changes in data distribution. In the case of BN, activation statistics will be updated for the most recent task, degrading the performance on older distributions in the style of weight overwriting. This issue has less of an impact in task-aware methods, where the task-id is available and used to choose the statistics for the current task [158]. Even when ignored, in task-aware parameter isolation, masks are not active concurrently, neurons will be active only in their relevant tasks, therefore having more consistent statistics.

To address this issue, the main idea from NF-ResNets is adopted, and WS is adapted to the proposed incremental learning scenario. This is done as explained in the WS experiments (Section 6.2.3). To make the comparison fairer with respect to the AlexNet architectures, a wider NF-ResNet20 is tested, which uses a width multiplicative factor of $k = 4$. The number of parameters is then 4.3M compared to the 6.5M of the AlexNet used for the rest of experiments.

Results show how, consistent with the analysis done in Sec. 4.4, WS reduces plasticity 6.9. Therefore, task-agnostic accuracy when all layers are standardized is very low, and only the first task achieves high accuracy. Therefore, performance when reducing the number of WS layers was tested. Eliminating standarisation caused unstable training, due to the ResNet's need for WS or BN (as previously explained). In such conditions, the optimizer struggled to find good weight configurations and often caused forgetting due to exploding gradients.

To solve the limitation in plasticity when using WS, a fully-connected layer of 2048 units is added after the convolutional feature extractor. This expanded network has 4.9M parameters and achieves competitive accuracy, since it now has now enough plasticity. Additionally, for this network, it can be seen how freezing the convolutional layers reduces plasticity along with forgetting, achieving a better balance for this setup and therefore higher average accuracy.

These experiments highlight the challenge of adapting normalisation or standardisation in ResNet architectures to task-agnostic parameter isolation. First, WS needs to be adapted to incremental learning (as defined in Section 6.2.3) in order to prevent forgetting. Then, the remaining challenge is the limit WS imposes on plasticity, which in this work is compensated by means of an additional fully-connected layer. Looking forward, it is of interest to find alternative solutions that circumvent this limitation, as adding extra layers is arguably a sub-optimal solution.

Additionally, Table 6.9 also reports the accuracy obtained with the expanded NF-ResNet20 and other methods, including the three baselines: *Finetuning*, *Freezing*, and *Joint*. These results allow to contextualize how the accuracy for LIFES is still competitive but the aforementioned effect of WS on plasticity limits the benefit of using ResNet. Other approaches such as LWF and FeTrIL demonstrated a greater improvement with the architecture change, with FeTrIL obtaining the highest accuracy in the (50×1 + 10×5) setup. These results are to be expected, as the WS challenge does not apply when the backbone is frozen, and the higher number of classes in the first task favours its freezing strategy. Regarding LWF, with the NF-ResNet there was a great reduction in its amount of forgetting compared to when it was used in other architectures. This seems to indicate that the plasticity reduction of WS provided the necessary stability to this method, while the additional FC layer provided enough plasticity. Additionally, it is also possible that part of the benefit is due to ResNet architectures being more suited for the knowledge distillation approach.

Apart from that, it is also interesting how Finetuning improved performance with this network. A hypothesis is that the limitation in plasticity caused by WS promoted, again, more stability, making this approach, which does not implement any CF prevention, find a better stability-plasticity balance.

6.4 Conclusions

This chapter presented a method which allows to expand parameter isolation from task-aware to task-agnostic IL without the use of rehearsal. By referring to the CF definition previously proposed in this thesis, the study clarified how this new task-agnostic paradigm prevents weight overwriting, but can suffer from representation overlap and weight interference. Therefore, solutions to alleviate their effect were proposed. SICP reduced weight interference, while LCR and WS

Tab. 6.9: Results for LIFES with NF-ResNet in CIFAR-100 (50×1 + 10×5). Additional methods also reported for reference. NF-ResNet20 uses a wider architecture of $k = 4$. One extra FC layer is added when indicated in the network column. Frozen stands for frozen layers. Which layers are frozen or standardized is indicated with a range of indexes or "All".

Approach	**Network**	**WS layers**	**Frozen**	**TAw Acc ↑**	**TAg Acc ↑**	**TAw Forg ↓**	**TAg Forg ↓**
LIFES	NF-ResNet20	All	[1,14]	67.53 (± 2.74)	13.78 (± 0.60)	9.08 (± 2.72)	1.00 (± 0.34)
LIFES	NF-ResNet20	[1,18]	[1,14]	8.96 (± 0.23)	1.76 (± 0.23)	14.06 (± 1.12)	15.68 (± 1.38)
LIFES	NF-ResNet20 + FC	All	[1,14]	74.48 (± 0.47)	34.98 (± 2.77)	4.78 (± 0.73)	13.98 (± 4.42)
LIFES	NF-ResNet20 + FC	All	[1,20]	78.98 (± 0.43)	39.08 (± 0.54)	0.35 (± 0.21)	5.95 (± 0.69)
Finetuning	NF-ResNet20 + FC	All	None	74.74 (± 0.65)	27.38 (± 1.90)	10.68 (± 0.77)	23.20 (± 3.62)
Freezing	NF-ResNet20 + FC	All	[1,20]	78.94 (± 0.29)	26.06 (± 0.75)	0.00 (± 0.00)	1.08 (± 0.29)
Joint	NF-ResNet20 + FC	All	None	85.26 (± 0.43)	64.52 (± 0.79)	-0.50 (± 0.36)	3.60 (± 0.75)
HAT [156]	NF-ResNet20 + FC	All	None	65.66 (± 0.62)	N/A	0.0 (± 0.00)	N/A
EWC [124]	NF-ResNet20 + FC	All	None	75.86 (± 0.98)	23.72 (± 2.18)	9.30 (± 1.22)	17.26 (± 5.22)
LWF [129]	NF-ResNet20 + FC	All	None	78.74 (± 0.91)	38.42 (± 3.52)	0.48 (± 0.25)	5.14 (± 2.13)
FeTrIL [146]	NF-ResNet20 + FC	All	[1,20]	75.84 (± 0.24)	48.30 (± 0.50)	1.56 (± 0.22)	20.98 (± 0.31)

were proven to alleviate representational overlap. Specifically, LCR showed in some cases even higher inter-class distance than the joint training upper bound, making it a promising stand-alone component which could be added to other methods to reduce representational overlap.

Competitive results were obtained, surpassing the performance of the reference task-agnostic approaches is several setups. This proves that the concurrent deployment of subnetworks can be a way forward as long as the relevant CF components are addressed. Moreover, it demonstrates how LIFES can serve as an alternative to current IL approaches when minimal requirements are needed.

Further improvement can be achieved by addressing the class-energy imbalance problem in the classifier layer, something that was out of the scope for this work, but has been addressed by complementary approaches such as [209]. Additionally, another way forward is the extrapolation of the method to larger architectures. ResNets were analysed for this purpose, demonstrating the challenge imposed by task-agnostic normalisation / standardisation. Despite finding a solution through the addition of an additional layer, it is most likely a sub-optimal workaround, as the plasticity of the network is still limited in the convolutional layers. Solutions enabling better plasticity or the use of a different network are of interest for the future.

Finally,as previously discussed, the capacity of LIFES for reducing the problem to selection of a classifier head is arguably a promising starting point for future approaches. An algorithm working with the classifier head information, such as out of distribution detection algorithms, can potentially fulfil this role.

Chapter 7

Conclusions

Feature extraction is at the core of Artificial Intelligence, defining the way in which AI algorithms build representations that allow them to perceive the world. As it was previously argued, this defines the limits of what these AI systems can achieve, therefore making it a topic of major relevance. As seen through the literature review, the feature extraction processes in state of the art machine learning achieve impressive results, but are still far from perfect. Therefore, this thesis aimed at identifying aspects of feature extraction which require further development in the path towards a fully automated future with ubiquitous AI. With this target in mind, the work was focused on the efficiency of visual feature extraction and the capacity for continuously learning new features without catastrophic forgetting. The former is necessary for deploying larger systems at the edge and democratizing AI. The latter is critical for deployed systems which need to adapt to new scenarios, and to make training with increasingly large bodies of data a feasible feat in the path to AGI.

To argue that such improvements are attainable, biological intelligence was used as reference, given that, compared to artificial systems, real brains demonstrates superior energy efficiency and capacity for continual learning. Following this line of thought, the first two contribution chapters took on the neuromorphic computing approach, making use of Spiking Neural Networks in order to achieve more efficient ANN implementations.

When using SNNs, a major concern is often the lower accuracy they achieve compared to conventional deep learning. This is caused by the implicit limitations in spiking computations (Section 2.2.4), but also by the lack of more advanced network architectures. Hence, Chapter 4 targeted this problem, focusing in the development of better residual architectures for SNN, and contributing with both, a novel study on residual connections and a new network configuration that achieved state of the art results at the time of publication. After the publication of this work, later developments confirmed how this is indeed a topic of interest for the community, as many works kept pursuing this same objective. Finally, from the contribution in Chapter 4, it is also relevant how it analysed the requirements of different types of residual connections in terms of computing and hardware. This analysis is often omitted in SNN publications when the proposed algorithms are not tested in neuromorphic hardware. Nevertheless, providing this information is of great importance to ensure usability in future neuromorphic implementations, and to guide future hardware developments.

To make SNNs a solution towards efficient feature extraction, not only it is needed to optimise their accuracy, but also to understand the full scope of their properties, which can be then exploited to gain further benefits. Chapter 5 addressed this topic by demonstrating how SNNs have an inherent capacity for the extraction of temporal features, even without recurrent connections, enabling the design of architectures with less parameters. The work discussed how SNNs achieve this through integration of data over time in their membranes, and it demonstrated how they can calculate both, time-dependent and time-invariant temporal features (Section 5.4.2). The distinction between these two types of features is often ignored when analysing temporal tasks, but it is relevant in order to properly asses the capacity of a system for temporal processing. Time-dependent features were equated to the creation of time-stamps, which requires the quantification of time passed between events. Time-invariant features, on the contrary, require to detect sequences of events independently of their location in time, the same way convolutions are translation invariant in space. To evaluate these capacities in a neuromorphic vision dataset, it was necessary to create a new task, as preexisting ones did not allow for this analysis (Section 5.2.1). Hence, the gap was filled with the creation of DVS-GC. This novel task has been publicly released and can be used by researchers to asses the capacity of their systems for spatio-temporal feature extraction in an event-based scenario.

Furthermore, the work in Chapter 5 also studied the effect of the leak factor in the proposed task. Results demonstrated that this leakage is critical when spiking neurons are implemented with reset by subtraction, as they retain voltage after spiking. Interestingly, it also showed how an SNNs can extract the aforementioned temporal features without using leak, as the reset to zero strategy paired with IF neurons achieved high performance in DVS-GC. Still, this does not imply that other tasks different from DVS-GC might not leverage the leak factor for their temporal feature extraction. The requirements to solve DVS-GC intersect with a wide range of temporal tasks, that need to understand the order in which events happen. However, it does not represent all of them. For example, alternative tasks might implement delays between the relevant events, while in DVS-GC, when one finishes the following one starts. Alternatively, actions could be displayed with varying speeds, which might create a requirement for variable integration times in the neurons. Studying the implications of these modified scenarios is of interest for the future, and will complement the conclusions extracted in this chapter. For this purpose, other tasks can be defined, or modifications of DVS-GC can be proposed.

Following the work with SNNs, Chapter 6 switched the focus to continual learning. As discussed, this is also a core need for future AI systems, which not only need to extract features accurately and efficiently, but also need to do it in natural environments, where data is not presented as an i.i.d. distribution, and the system needs to learn from new information while retaining the relevant old knowledge. For its experiments, this chapter made use of conventional ANNs, as opposed to the previous two. This allowed to make results comparable to other works in continual learning, which also use non-spiking ANNs. Additionally, it made image classification training much faster, as SNNs still need to process images through time, requiring more expensive training algorithms such as BPTT. Training speed was a relevant factor, as the larger volume of experimental results allowed for more robust and well-proven conclusions. Nevertheless, continual learning methods, including the one developed in this thesis, can in principle be applied to either conventional ANNs or SNNs, the same way many other training algorithms are applied seamlessly to any neuron model, such as loss regularisations like weight decay [62, 210], or fine-tuning (Section 4.3).

The approach followed to design the novel continual learning algorithm started from the requirements of the method. Future systems must not depend on having an external input that indicates which task they are solving, neither can they rely on collecting examples of all previously seen distributions. Therefore, a system without these requirements was defined. Given that the brain can be seen as a modular system with specialised sub-networks, it is an attractive approach to base artificial algorithms in this same principle. The proposed LIFES algorithm instantiated this sub-network approach, executing all of them concurrently. This allowed to study what happens in such scenario, and to judge whether it can be a way forward for continual learning. The work proposed a novel definition of the causes of catastrophic forgetting (Section 3.1), which allowed to identify how concurrent sub-network deployment suffers from weight interference and representational overlap. Then, the LIFES algorithm was designed to alleviate this two causes of CF, through SICP, WS and LCR, making concurrent sub-networks a viable approach which demonstrated competitive performance in multiple incremental learning setups. Additionally, the effect of LCR in inter-class distance was noteworthy, making it a viable stand-alone regularisation for the reduction of representational overlap, which other methods can incorporate.

Overall, this thesis presented a set of contributions towards visual feature extraction, striving for efficiency, accuracy and learning adaptability to natural environments. In the short term, these contributions enable the design of more efficient, accurate and adaptable systems. In the greater journey towards ubiquitous AI and AGI, these represent just tiny incremental steps in specific research directions. Some of these research directions might be superseded and abandoned, while others may become vital parts of the path towards higher artificial intelligence. Regardless of the future, all research directions will have their value, as exploration is necessary to find what works and what does not. Hence it is the author's hope that the steps taken in this thesis will contribute to this mission and prove useful for future research.

7.1 Future work

Following the novel SNN architecture presented in Chapter 4, the energy efficiency of the S-ResNet was demonstrated in a later publication [52] by implementing a version of it in a Loihi [211] neuromorphic chip. Still, further testing on neuromorphic hardware is of interest for future work, as only one of the proposed residual connections was tested in the aforementioned work. Therefore, an efficiency comparison between the three types of residual connections would be valuable. Additionally, the requirements for S2S and V2V implementations are better suited for later neuromorphic chips such as Loihi 2 [165], hence an implementation on it is also of interest.

Regarding the work presented in Chapter 5, as previously discussed, DVS-GC defines a task with a specific set of properties. Expanding DVS-GC to create alternative tasks, such as one implementing delays between gestures or including actions with varying speeds, is a promising direction to contribute to this line of research. Doing so would allow to create new scenarios where certain components of SNNs, such as the leak factor, might have a different role. Therefore, an expanded DVS-GC would contribute to extracting additional conclusions on the capacity of SNNs for spatio-temporal feature extraction, and the role of their components.

Finally, Chapter 6 proposed the novel LIFES algorithm, and discussed how this new approach has potential to be further improved. A promising way forward is to combine it with other CL approaches tackling class-energy imbalance, as LIFES only works on the feature extraction process and this CF cause is found in the classifier weights. Alternatively, it was also proposed to combine LIFES with an algorithm for classifier head selection, as task-aware accuracy suffered almost no forgetting. Finally, its application to deeper networks is also a major necessity for future work, as the proposed solution for NF-ResNets was proven to constrain plasticity, and therefore alternative solutions with improved plasticity or alternative architectures are needed. In conclusion, the approach can be seen as the first instantiation of a new class of parameter isolation approaches, which have minimal requirements, while further work has the potential to push the performance of this new paradigm to higher levels. Such performance improvements are necessary in the AI roadmap: LIFES perform well enough to replace existing task-agnostic approaches in the tested setups, but life-long learning requires much higher robustness to forgetting, as this is the first step towards truly adaptable AI.

List of publications

Published

1. P. Kirkland, D. Manna, **A. Vicente-Sola**, and G. Di Caterina, "Unsupervised spiking instance segmentation on event data using stdp features," IEEE Transactions on Computers, 2022.

2. **A. Vicente-Sola**, D. L. Manna, P. Kirkland, G. Di Caterina, and T. J. Bihl, "Keys to accurate feature extraction using residual spiking neural networks," Neuromorphic Computing and Engineering, 2022.

3. D. L. Manna, **A. Vicente-Sola**, P. Kirkland, T. J. Bihl, and G. Di Caterina, "Simple and complex spiking neurons: perspectives and analysis in a simple stdp scenario," Neuromorphic Computing and Engineering, vol. 2, no. 4, p. 044009, 2022.

4. T. J. Bihl, P. Farr, G. Di Caterina, P. Kirkland, **A. Vicente Sola**, D. Manna, J. Liu, and K. Combs, "Exploring spiking neural networks (snn) for low size, weight, and power (swap) benefits," Hawaii International Conference on System Sciences 2024 (HICSS-57), 2023.

5. D. L. Manna, **A. Vicente-Sola**, P. Kirkland, T. J. Bihl, and G. Di Caterina, "Frameworks for snns: A review of data science-oriented software and an expansion of spyketorch," in Engineering Applications of Neural Networks, pp. 227–238, Springer Nature Switzerland, 2023.

6. P. Chaudhari, **A. Vicente-Sola**, A. Basu, D. L. Manna, P. Kirkland, and G. D. Caterina, "Sign language recognition using spiking neural networks," Procedia Computer Science, vol. 235, pp. 2674–2683, 2024. International Conference on Machine Learning and Data Engineering (ICMLDE 2023).

7. D. L. Manna, **A. Vicente-Sola**, P. Kirkland, T. J. Bihl, and G. Di Caterina, "Time series forecasting via derivative spike encoding and bespoke loss

functions for spiking neural networks," Computers, vol. 13, no. 8, 2024.

8. G. Bent, C. Davies, M.R. Vilamala, Y. Li, A. Preece, G. Di Caterina, **A. Vicente-Sola**, P. Kirkland, G. Pearson, and B. Tutcher. "A demonstration of vector symbolic architecture as an effective integrated technology for AI at the network edge". In Artificial Intelligence for Security and Defence Applications II (Vol. 13206, pp. 421-436). SPIE, 2024.

9. G. Bent, C. Davies, M.R. Vilamala, Y. Li, A. Preece, G. Di Caterina, **A. Vicente-Sola**, P. Kirkland, G. Pearson, and B. Tutcher. "The transformative potential of vector symbolic architecture for cognitive processing at the network edge". In Artificial Intelligence for Security and Defence Applications II (Vol. 13206, pp. 404-420). SPIE, 2024.

10. **A. Vicente-Sola**, D. L. Manna, P. Kirkland, G. Di Caterina, and T. J. Bihl, "Spiking Neural Networks for event-based action recognition: A new task to understand their advantage" Neurocomputing, vol. 611, p. 128657, 2024

11. D. L. Manna, G. Di Caterina, **A. Vicente-Sola**, and P. Kirkland. "An approach to time series forecasting with derivative spike encoding and spiking neural networks." Hawaii International Conference on System Sciences 2025

Under review

1. **A. Vicente-Sola**, P. Kirkland, M. Aquilina, and A. Lyons, "Single photon event-driven 3d imaging," researchsquare preprint 10.21203/rs.3.rs-4351302/v1, 2024

2. **A. Vicente-Sola**, P. Kirkland, G. Di Caterina, T. Bihl, and M. Masana, "From task-aware to task-agnostic parameter isolation for incremental learning"

References

[1] O. Galor, *Unified growth theory*. Princeton University Press, 2011. 1

[2] D. Comin, W. Easterly, and E. Gong, “Was the wealth of nations determined in 1000 bc?,” *American Economic Journal: Macroeconomics*, vol. 2, no. 3, pp. 65–97, 2010. 1

[3] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, *et al.*, “A general reinforcement learning algorithm that masters chess, shogi, and go through self-play,” *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018. 1

[4] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, *et al.*, “Highly accurate protein structure prediction with alphafold,” *nature*, vol. 596, no. 7873, pp. 583–589, 2021. 1

[5] M. R. Morris, J. Sohl-Dickstein, N. Fiedel, T. Warkentin, A. Dafoe, A. Faust, C. Farabet, and S. Legg, “Position: Levels of agi for operationalizing progress on the path to agi,” in *Forty-first International Conference on Machine Learning*, 2024. 1

[6] Y. Bengio, G. Hinton, A. Yao, D. Song, P. Abbeel, T. Darrell, Y. N. Harari, Y.-Q. Zhang, L. Xue, S. Shalev-Shwartz, *et al.*, “Managing extreme ai risks amid rapid progress,” *Science*, vol. 384, no. 6698, pp. 842–845, 2024. 1

[7] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg, *et al.*, “Sparks of artificial general intelligence: Early experiments with gpt-4,” *arXiv preprint arXiv:2303.12712*, 2023. 1

[8] D. V. Christensen, R. Dittmann, B. Linares-Barranco, A. Sebastian, M. Le Gallo, A. Redaelli, S. Slesazeck, T. Mikolajick, S. Spiga, S. Menzel, *et al.*, “2022 roadmap on neuromorphic computing and engineering,” *Neuromorphic Computing and Engineering*, vol. 2, no. 2, p. 022501, 2022. 13

[9] F. Zenke and E. O. Neftci, “Brain-inspired learning on neuromorphic substrates,” *Proceedings of the IEEE*, vol. 109, no. 5, pp. 935–950, 2021. 1

[10] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars, “A continual learning survey: Defying forgetting in classification tasks,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 7, pp. 3366–3385, 2021. 1, 3, 25, 26, 34

[11] D. Hassabis, D. Kumaran, C. Summerfield, and M. Botvinick, “Neuroscience-inspired artificial intelligence,” *Neuron*, vol. 95, no. 2, pp. 245–258, 2017. 2

[12] T. Poggio, D. Hassabis, and D. S. I. S. Geoffrey Hinton, Pietro Perona, “Panel discussion at cbmm10.” Presented at CBMM10 - A Symposium on Intelligence: Brains, Minds, and Machines, MIT, Cambridge, MA, August 2023. 2

[13] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015. 3

[14] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pp. 234–241, Springer, 2015. 3

[15] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6299–6308, 2017. 3

[16] W. Peebles and S. Xie, "Scalable diffusion models with transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023. 3

[17] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in cognitive sciences*, vol. 3, no. 4, pp. 128–135, 1999. 3, 25, 26

[18] R. Kemker, M. McClure, A. Abitino, T. Hayes, and C. Kanan, "Measuring catastrophic forgetting in neural networks," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018. 3, 25

[19] O. Sporns, G. Tononi, and G. Edelman, "Connectivity and complexity: the relationship between neuroanatomy and brain dynamics," *Neural Networks*, vol. 13, no. 8, pp. 909–922, 2000. 3, 82

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016. 4, 11, 37, 40, 43, 48, 54, 89, 96, 98

[21] C. Lee, S. S. Sarwar, P. Panda, G. Srinivasan, and K. Roy, "Enabling spike-based backpropagation for training deep neural network architectures," *Frontiers in neuroscience*, vol. 14, p. 119, 2020. 4, 15, 18, 19, 24, 39, 43, 57

[22] W. Fang, Z. Yu, Y. Chen, T. Huang, T. Masquelier, and Y. Tian, "Deep residual learning in spiking neural networks," *Advances in Neural Information Processing Systems*, vol. 34, 2021. 4, 13, 15, 16, 18, 24, 39, 40, 43, 57

[23] A. Vicente-Sola, D. L. Manna, P. Kirkland, G. Di Caterina, and T. Bihl, "Keys to accurate feature extraction using residual spiking neural networks," *Neuromorphic Computing and Engineering*, vol. 2, no. 4, p. 044001, 2022. 4, 15, 22, 97

[24] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015. 7, 16

[25] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989. 7

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012. 7

[27] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, pp. 211–252, 2015. 7

[28] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, “Support vector machines,” *IEEE Intelligent Systems and their applications*, vol. 13, no. 4, pp. 18–28, 1998. 7

[29] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *nature*, vol. 323, no. 6088, pp. 533–536, 1986. 7

[30] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel, “Handwritten digit recognition with a back-propagation network,” *Advances in neural information processing systems*, vol. 2, 1989. 8, 9

[31] A. Diamond, “Executive functions,” *Annual review of psychology*, vol. 64, pp. 135–168, 2013. 9

[32] N. Cowan, “What are the differences between long-term, short-term, and working memory?,” *Progress in brain research*, vol. 169, pp. 323–338, 2008. 9

[33] R. M. Schmidt, “Recurrent neural networks (rnns): A gentle introduction and overview,” *arXiv preprint arXiv:1912.05911*, 2019. 9

[34] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997. 9

[35] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017. 10, 12, 37, 97

[36] S. Takase and S. Kiyono, “Lessons on parameter sharing across layers in transformers,” *arXiv preprint arXiv:2104.06022*, 2021. 10

[37] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, “Informer: Beyond efficient transformer for long sequence time-series forecasting,” in *Proceedings of AAAI*, 2021.

[38] C. Wei, H. Fan, S. Xie, C.-Y. Wu, A. Yuille, and C. Feichtenhofer, “Masked feature prediction for self-supervised visual pre-training,” *arXiv preprint arXiv:2112.09133*, 2021. 10

[39] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2015. 11

[40] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *International conference on machine learning*, pp. 448–456, PMLR, 2015. 11, 12, 86, 98

[41] A. Brock, S. De, and S. L. Smith, “Characterizing signal propagation to close the performance gap in unnormalized resnets,” *arXiv preprint arXiv:2101.08692*, 2021. 11, 86, 98

[42] S. Qiao, H. Wang, C. Liu, W. Shen, and A. Yuille, “Micro-batch training with batch-channel normalization and weight standardization,” *arXiv preprint arXiv:1903.10520*, 2019. 12, 86

[43] K. Fukushima, “Visual feature extraction by a multilayered network of analog threshold elements,” *IEEE Transactions on Systems Science and Cybernetics*, vol. 5, no. 4, pp. 322–333, 1969. 12

[44] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, 2018. 12

[45] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014. 12

[46] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning*, pp. 6105–6114, PMLR, 2019. 12, 20, 37, 97

[47] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2820–2828, 2019. 12

[48] C. Mead, "Analog vlsi and neural systems," 1989. 12

[49] G. Indiveri, *Neuromorphic Engineering*, pp. 715–725. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015. 13

[50] E. M. Izhikevich, "Which model to use for cortical spiking neurons?," *IEEE transactions on neural networks*, vol. 15, no. 5, pp. 1063–1070, 2004. 13

[51] M. Davies, A. Wild, G. Orchard, Y. Sandamirskaya, G. A. F. Guerra, P. Joshi, P. Plank, and S. R. Risbud, "Advancing neuromorphic computing with loihi: A survey of results and outlook," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 911–934, 2021. 13, 61

[52] T. J. Bihl, P. Farr, G. Di Caterina, P. Kirkland, A. Vicente Sola, D. Manna, J. Liu, and K. Combs, "Exploring spiking neural networks (snn) for low size, weight, and power (swap) benefits," *Hawaii International Conference on System Sciences 2024 (HICSS-57)*, 2023. 13, 104

[53] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski, *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014. 13

[54] L. F. Abbott, "Lapicque's introduction of the integrate-and-fire model neuron (1907)," *Brain research bulletin*, vol. 50, no. 5-6, pp. 303–304, 1999. 13, 37

[55] H. Zheng, Y. Wu, L. Deng, Y. Hu, and G. Li, "Going deeper with directly-trained larger spiking neural networks," in *AAAI*, 2021. 13, 15, 16, 17, 18, 24, 39, 43, 46, 57, 64, 65

[56] Z. Zhou, Y. Zhu, C. He, Y. Wang, S. YAN, Y. Tian, and L. Yuan, "Spikformer: When spiking neural network meets transformer," in *The Eleventh International Conference on Learning Representations*, 2023. 13, 15, 16, 17, 22, 24

[57] W. Potjans, A. Morrison, and M. Diesmann, "A spiking neural network model of an actor-critic learning agent," *Neural Computation*, vol. 21, pp. 301–339, 2009. 15

[58] Z. Bing, C. Meschede, F. Röhrbein, K. Huang, and A. Knoll, "A survey of robotics control based on learning-inspired spiking neural networks," *Frontiers in Neurorobotics*, vol. 12, 2018. 15

[59] D. Reid, A. J. Hussain, and H. Tawfik, "Financial time series prediction using spiking neural networks," *PloS one*, vol. 9, no. 8, p. e103656, 2014. 15

[60] R.-J. Zhu, Q. Zhao, G. Li, and J. K. Eshraghian, "Spikegpt: Generative pre-trained language model with spiking neural networks," *arXiv preprint arXiv:2302.13939*, 2023. 15, 16

[61] B. Han and K. Roy, "Deep spiking neural network: Energy efficiency through time based coding," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part X 16*, pp. 388–404, Springer, 2020. 15, 24, 37, 57

[62] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, "Going deeper in spiking neural networks: Vgg and residual architectures," *Frontiers in neuroscience*, vol. 13, p. 95, 2019. 24, 37, 57, 103

[63] Y. Kim and P. Panda, "Revisiting batch normalization for training low-latency deep spiking neural networks from scratch," *Frontiers in neuroscience*, p. 1638, 2020. 15, 17, 18, 24, 43, 46, 48, 57

[64] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020. 15, 24

[65] G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, and W. Maass, "Long short-term memory and learning-to-learn in networks of spiking neurons," *Advances in neural information processing systems*, vol. 31, 2018. 16

[66] A. M. Lamb, A. G. ALIAS PARTH GOYAL, Y. Zhang, S. Zhang, A. C. Courville, and Y. Bengio, "Professor forcing: A new algorithm for training recurrent networks," *Advances in neural information processing systems*, vol. 29, 2016. 16

[67] e. a. Garofolo, John S., "Timit acoustic-phonetic continuous speech corpus," *Linguistic Data Consortium, Philadelphia*, 1983. 16

[68] A. Lotfi Rezaabad and S. Vishwanath, "Long short-term memory spiking networks and their applications," in *International Conference on Neuromorphic Systems 2020*, pp. 1–9, 2020. 16

[69] E. Calabrese, G. Taverni, C. Awai Easthope, S. Skriabine, F. Corradi, L. Longinotti, K. Eng, and T. Delbruck, "Dhp19: Dynamic vision sensor 3d human pose dataset," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pp. 0–0, 2019. 16, 63

[70] A. Amir, B. Taba, D. J. Berg, T. Melano, J. L. McKinstry, C. di Nolfo, T. K. Nayak, A. Andreopoulos, G. Garreau, M. Mendoza, J. A. Kusnitz, M. V. DeBole, S. K. Esser, T. Delbrück, M. Flickner, and D. S. Modha, "A low power, fully event-based gesture recognition system," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7388–7397, 2017. 16, 61, 62, 63

[71] Y. Kim and P. Panda, "Optimizing deeper spiking neural networks for dynamic vision sensing," *Neural Networks*, vol. 144, pp. 686–698, 2021. 16

[72] Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi, "Spatio-temporal backpropagation for training high-performance spiking neural networks," *Frontiers in neuroscience*, vol. 12, p. 331, 2018. 17

[73] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 51–63, 2019. 17

[74] D. Huh and T. J. Sejnowski, "Gradient descent for spiking neural networks," in *Advances in Neural Information Processing Systems* (S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), vol. 31, Curran Associates, Inc., 2018. 17

[75] H. Mostafa and G. Cauwenberghs, "A learning framework for winner-take-all networks with stochastic synapses," *Neural computation*, vol. 30, no. 6, pp. 1542–1572, 2018. 17

[76] H. Mostafa, "Supervised learning based on temporal coding in spiking neural networks," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 7, pp. 3227–3235, 2017. 17

[77] J. Kaiser, H. Mostafa, and E. Neftci, "Synaptic plasticity dynamics for deep continuous local learning (decolle)," *Frontiers in Neuroscience*, vol. 14, p. 424, 2020. 17, 24, 64, 65

[78] Ł. Kuśmierz, T. Isomura, and T. Toyoizumi, "Learning with three factors: modulating hebbian plasticity with errors," *Current opinion in neurobiology*, vol. 46, pp. 170–177, 2017. 17

[79] W. Fang, Z. Yu, Y. Chen, T. Masquelier, T. Huang, and Y. Tian, "Incorporating learnable membrane time constant to enhance learning of spiking neural networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2661–2671, 2021. 17, 18, 24, 44, 46, 51, 54, 57, 62, 64, 65

[80] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990. 17

[81] S. Shrestha and G. Orchard, "Slayer: Spike layer error reassignment in time," in *NeurIPS*, 2018. 18, 62

[82] G. Bellec, F. Scherr, A. Subramoney, E. Hajek, D. Salaj, R. Legenstein, and W. Maass, "A solution to the learning dilemma for recurrent networks of spiking neurons," *Nature communications*, vol. 11, no. 1, p. 3625, 2020. 18

[83] T. C. Wunderlich and C. Pehle, "Event-based backpropagation can compute exact gradients for spiking neural networks," *Scientific Reports*, vol. 11, no. 1, p. 12829, 2021. 18

[84] S. Schmitt, J. Klähn, G. Bellec, A. Grübl, M. Guettler, A. Hartel, S. Hartmann, D. Husmann, K. Husmann, S. Jeltsch, *et al.*, "Neuromorphic hardware in the loop: Training a deep spiking network on the brainscales wafer-scale system," in *2017 international joint conference on neural networks (IJCNN)*, pp. 2227–2234, IEEE, 2017. 19

[85] E. Van Doremaele, X. Ji, J. Rivnay, and Y. Van De Burgt, "A retrainable neuromorphic biosensor for on-chip learning and classification," *Nature Electronics*, vol. 6, no. 10, pp. 765–770, 2023. 19

[86] L. Deng, Y. Wu, X. Hu, L. Liang, Y. Ding, G. Li, G. Zhao, P. Li, and Y. Xie, "Rethinking the performance comparison between SNNS and ANNS," *Neural Networks*, vol. 121, pp. 294–307, 2020. 19

[87] S. K. Esser, P. A. Merolla, J. V. Arthur, A. S. Cassidy, R. Appuswamy, A. Andreopoulos, D. J. Berg, J. L. McKinstry, T. Melano, D. R. Barch, *et al.*, "Convolutional networks for fast, energy-efficient neuromorphic computing," *Proceedings of the national academy of sciences*, vol. 113, no. 41, pp. 11441–11446, 2016. 19, 24

[88] A. Kugele, T. Pfeil, M. Pfeiffer, and E. Chicca, "Efficient processing of spatio-temporal data streams with spiking neural networks," *Frontiers in Neuroscience*, vol. 14, p. 439, 2020. 19, 61

[89] B. Rueckauer, I.-a. Lungu, Y. Hu, and M. Pfeiffer, "Conversion of Continuous-Valued Deep Networks to Efficient Event-Driven Networks for Image Classification," vol. 11, no. December, pp. 1–12, 2017. 20, 24

[90] A. Kugele, T. Pfeil, M. Pfeiffer, and E. Chicca, “Efficient Processing of Spatio-Temporal Data Streams With Spiking Neural Networks,” *Frontiers in Neuroscience*, vol. 14, no. May, pp. 1–13, 2020. 20, 24

[91] V. Fischer, J. Köhler, and T. Pfeil, “The streaming rollout of deep networks - Towards fully model-parallel execution,” *Advances in Neural Information Processing Systems*, vol. 2018-Decem, no. Nips, pp. 4039–4050, 2018. 20

[92] C. Stöckl and W. Maass, “Recognizing images with at most one spike per neuron,” *arXiv preprint arXiv:2001.01682*, 2019. 20, 24

[93] P. U. Diehl and M. Cook, “Unsupervised learning of digit recognition using spike-timing-dependent plasticity,” vol. 9, no. August, pp. 1–9, 2015. 21

[94] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, “Adversarial Autoencoders,” 2015. 21

[95] K. D. Carlson, M. Richert, N. Dutt, and J. L. Krichmar, “Biologically plausible models of homeostasis and stdp: stability and learning in spiking neural networks,” in *The 2013 international joint conference on neural networks (IJCNN)*, pp. 1–8, IEEE, 2013. 21

[96] Y. LeCun and C. Cortes, “MNIST handwritten digit database,” 2010. 21

[97] G. Orchard, A. Jayawant, G. K. Cohen, and N. Thakor, “Converting static image datasets to spiking neuromorphic datasets using saccades,” *Frontiers in neuroscience*, vol. 9, p. 437, 2015. 21, 63

[98] A. Krizhevsky, G. Hinton, *et al.*, “Learning multiple layers of features from tiny images,” 2009. 21, 83, 89

[99] H. Li, H. Liu, X. Ji, G. Li, and L. Shi, “Cifar10-dvs: an event-stream dataset for object classification,” *Frontiers in neuroscience*, vol. 11, p. 309, 2017. 21, 54, 63

[100] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009. 21

[101] X. Qiu, R.-J. Zhu, Y. Chou, Z. Wang, L.-j. Deng, and G. Li, “Gated attention coding for training high-performance and efficient spiking neural networks,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, pp. 601–610, 2024. 22, 24

[102] A. Byerly, T. Kalganova, and I. Dear, “A Branching and Merging Convolutional Network with Homogeneous Filter Capsules,” 2020. 24

[103] Y. Wu, L. Deng, G. Li, J. Zhu, Y. Xie, and L. Shi, “Direct Training for Spiking Neural Networks: Faster, Larger, Better,” 2019. 24

[104] A. Kolesnikov, L. Beyer, X. Zhai, and C. V. May, “General Visual Representation Learning,” 24

[105] Y. Kim and P. Panda, “Revisiting batch normalization for training low latency deep spiking neural networks from scratch,” no. Burkitt 2006, pp. 1–14, 2020. 24

[106] C. Lee, S. S. Sarwar, P. Panda, G. Srinivasan, and K. Roy, “Enabling Spike-Based Backpropagation for Training Deep Neural Network Architectures,” *Frontiers in Neuroscience*, vol. 14, no. February, pp. 1–22, 2020. 24

[107] H. Zheng, Y. Wu, L. Deng, Y. Hu, and G. Li, “Going Deeper With Directly-Trained Larger Spiking Neural Networks,” 2020. 24

[108] J. Wu, C. Xu, X. Han, D. Zhou, M. Zhang, H. Li, and K. C. Tan, “Progressive tandem learning for pattern recognition with deep spiking neural networks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 24, 57

[109] S. Deng and S. Gu, “Optimal conversion of conventional artificial neural networks to spiking neural networks,” in *International Conference on Learning Representations*, 2021. 24, 57

[110] A. Samadzadeh, F. Sadat, T. Far, A. Javadi, and A. Nickabadi, “Convolutional Spiking Neural Networks for Spatio-Temporal Feature Extraction,” 2020. 24

[111] H. Touvron, A. Vedaldi, M. Douze, and H. Jégou, “Fixing the train-test resolution discrepancy: Fixefficientnet,” *arXiv preprint arXiv:2003.08237*, 2020. 24

[112] M. Tan and Q. V. Le, “EfficientNet : Rethinking Model Scaling for Convolutional Neural Networks,” 2019. 24

[113] W. Fang, Z. Yu, T. Masquelier, Y. Chen, T. Huang, and Y. Tian, “Spike-based residual blocks,” *arXiv preprint arXiv:2102.04159*, 2021. 24

[114] M. Yao, J. Hu, Z. Zhou, L. Yuan, Y. Tian, B. Xu, and G. Li, “Spike-driven transformer,” *Advances in Neural Information Processing Systems*, vol. 36, 2024. 24

[115] E. Verwimp, S. Ben-David, M. Bethge, A. Cossu, A. Gepperth, T. L. Hayes, E. Hüllermeier, C. Kanan, D. Kudithipudi, C. H. Lampert, *et al.*, “Continual learning: Applications and the road forward,” *arXiv preprint arXiv:2311.11908*, 2023. 25, 26

[116] R. L. Davis and Y. Zhong, “The biology of forgetting—a perspective,” *Neuron*, vol. 95, no. 3, pp. 490–503, 2017. 25

[117] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, “Generalizing from a few examples: A survey on few-shot learning,” *ACM computing surveys (csur)*, vol. 53, no. 3, pp. 1–34, 2020. 25

[118] M. Mermillod, A. Bugaiska, and P. Bonin, “The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects,” *Frontiers in psychology*, vol. 4, p. 54654, 2013. 26

[119] M. Masana, X. Liu, B. Twardowski, M. Menta, A. D. Bagdanov, and J. van de Weijer, “Class-incremental learning: Survey and performance evaluation on image classification,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 5, pp. 5513–5533, 2023. 26, 28

[120] E. Belouadah, A. Popescu, and I. Kanellos, “A comprehensive study of class incremental learning algorithms for visual tasks,” *Neural Networks*, vol. 135, pp. 38–54, 2021. 26

[121] G. M. van de Ven and A. S. Tolias, “Three continual learning scenarios,” *NeurIPS Continual Learning Workshop*, vol. 9, 2018. 26

[122] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, “icarl: Incremental classifier and representation learning,” in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017. 28, 30, 84

[123] J. Zhang, J. Zhang, S. Ghosh, D. Li, S. Tasci, L. Heck, H. Zhang, and C.-C. J. Kuo, “Class-incremental learning via deep model consolidation,” in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 1131–1140, 2020. 28

[124] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017. 28, 83, 84, 89, 91, 97, 100

[125] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, "Memory aware synapses: Learning what (not) to forget," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 139–154, 2018. 28, 84

[126] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *International conference on machine learning*, pp. 3987–3995, PMLR, 2017. 28

[127] R. Pascanu and Y. Bengio, "Revisiting natural gradient for deep networks," *arXiv preprint arXiv:1301.3584*, 2013. 29

[128] R. Aljundi, K. Kelchtermans, and T. Tuytelaars, "Task-free continual learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11254–11263, 2019. 29

[129] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 12, pp. 2935–2947, 2017. 29, 83, 84, 89, 91, 97, 100

[130] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015. 29

[131] S. Hou, X. Pan, C. C. Loy, Z. Wang, and D. Lin, "Lifelong learning via progressive distillation and retrospection," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 437–452, 2018. 30

[132] S. Hou, X. Pan, C. C. Loy, Z. Wang, and D. Lin, "Learning a unified classifier incrementally via rebalancing," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 831–839, 2019.

[133] H. Cha, J. Lee, and J. Shin, "Co2l: Contrastive continual learning," in *Proceedings of the IEEE/CVF International conference on computer vision*, pp. 9516–9525, 2021. 30

[134] A. Chaudhry, M. Rohrbach, M. Elhoseiny, T. Ajanthan, P. K. Dokania, P. H. Torr, and M. Ranzato, "On tiny episodic memories in continual learning. arxiv," *Learning*, vol. 6, no. 7, 2019. 30

[135] L. Caccia, E. Belilovsky, M. Caccia, and J. Pineau, "Online learned continual compression with adaptive quantization modules," in *International conference on machine learning*, pp. 1240–1250, PMLR, 2020. 30

[136] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," *Advances in neural information processing systems*, vol. 30, 2017. 30

[137] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny, "Efficient lifelong learning with a-gem," *arXiv preprint arXiv:1812.00420*, 2018. 30

[138] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," *Advances in neural information processing systems*, vol. 30, 2017. 31

[139] A. Seff, A. Beatson, D. Suo, and H. Liu, "Continual learning in generative adversarial nets," *arXiv preprint arXiv:1705.08395*, 2017. 31

[140] L. Wang, K. Yang, C. Li, L. Hong, Z. Li, and J. Zhu, "Ordisco: Effective and efficient usage of incremental unlabeled data for semi-supervised continual learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5383–5392, 2021. 31

[141] C. He, R. Wang, S. Shan, and X. Chen, "Exemplar-supported generative reproduction for class incremental learning.," in *BMVC*, vol. 1, p. 2, 2018. 31

[142] F. Zhu, X.-Y. Zhang, C. Wang, F. Yin, and C.-L. Liu, "Prototype augmentation and self-supervision for incremental learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5871–5880, 2021. 31, 32, 84

[143] K. Zhu, W. Zhai, Y. Cao, J. Luo, and Z.-J. Zha, "Self-sustaining representation expansion for non-exemplar class-incremental learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9296–9305, 2022. 31, 32, 84

[144] F. Zhu, Z. Cheng, X.-Y. Zhang, and C.-l. Liu, "Class-incremental learning via dual augmentation," *Advances in Neural Information Processing Systems*, vol. 34, pp. 14306–14318, 2021. 31

[145] W. Shi and M. Ye, "Prototype reminiscence and augmented asymmetric knowledge aggregation for non-exemplar class-incremental learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1772–1781, 2023. 31

[146] G. Petit, A. Popescu, H. Schindler, D. Picard, and B. Delezoide, "Fetril: Feature translation for exemplar-free class-incremental learning," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 3911–3920, 2023. 32, 83, 84, 91, 97, 100

[147] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," *Advances in neural information processing systems*, vol. 27, 2014. 32, 89

[148] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," *ACM Computing Surveys*, vol. 55, no. 9, pp. 1–35, 2023. 32

[149] H. Liu, D. Tam, M. Muqeeth, J. Mohta, T. Huang, M. Bansal, and C. A. Raffel, "Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 1950–1965, 2022. 32

[150] Z. Wang, Z. Zhang, C.-Y. Lee, H. Zhang, R. Sun, X. Ren, G. Su, V. Perot, J. Dy, and T. Pfister, "Learning to prompt for continual learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 139–149, 2022. 33

[151] Z. Wang, Z. Zhang, S. Ebrahimi, R. Sun, H. Zhang, C.-Y. Lee, X. Ren, G. Su, V. Perot, J. Dy, *et al.*, "Dualprompt: Complementary prompting for rehearsal-free continual learning," in *European Conference on Computer Vision*, pp. 631–648, Springer, 2022. 33

[152] J. S. Smith, L. Karlinsky, V. Gutta, P. Cascante-Bonilla, D. Kim, A. Arbelle, R. Panda, R. Feris, and Z. Kira, "Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11909–11919, 2023. 33

[153] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," *arXiv preprint arXiv:1606.04671*, 2016. 33

[154] J. Yoon, E. Yang, J. Lee, and S. J. Hwang, "Lifelong learning with dynamically expandable networks," *arXiv preprint arXiv:1708.01547*, 2017.

[155] J. Hurtado, A. Raymond, and A. Soto, "Optimizing reusable knowledge for continual learning via metalearning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 14150–14162, 2021. 33

[156] J. Serra, D. Suris, M. Miron, and A. Karatzoglou, "Overcoming catastrophic forgetting with hard attention to the task," in *International conference on machine learning*, pp. 4548–4557, PMLR, 2018. 33, 83, 84, 85, 89, 91, 97, 100

[157] A. Mallya and S. Lazebnik, "Packnet: Adding multiple tasks to a single network by iterative pruning," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 7765–7773, 2018. 33, 34, 84

[158] M. Masana, T. Tuytelaars, and J. Van de Weijer, "Ternary feature masks: zero-forgetting for task-incremental learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3570–3579, 2021. 33, 34, 84, 85, 98

[159] H. Kang, R. J. L. Mina, S. R. H. Madjid, J. Yoon, M. Hasegawa-Johnson, S. J. Hwang, and C. D. Yoo, "Forget-free continual learning with winning subnetworks," in *Proceedings of the 39th International Conference on Machine Learning* (K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, eds.), vol. 162 of *Proceedings of Machine Learning Research*, pp. 10734–10750, PMLR, 17–23 Jul 2022. 33, 84

[160] G. Kim, B. Liu, and Z. Ke, "A multi-head model for continual learning via out-of-distribution replay," in *Conference on Lifelong Learning Agents*, pp. 548–563, PMLR, 2022. 34, 84

[161] G. Kim, C. Xiao, T. Konishi, and B. Liu, "Learnability and algorithm for continual learning," in *International Conference on Machine Learning*, pp. 16877–16896, PMLR, 2023. 34, 84

[162] M. Wortsman, V. Ramanujan, R. Liu, A. Kembhavi, M. Rastegari, J. Yosinski, and A. Farhadi, "Supermasks in superposition," in *Advances in Neural Information Processing Systems* (H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds.), vol. 33, pp. 15173–15184, Curran Associates, Inc., 2020. 35, 84

[163] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio, "An empirical investigation of catastrophic forgetting in gradient-based neural networks," *arXiv preprint arXiv:1312.6211*, 2013. 35

[164] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018. 37, 97

[165] G. Orchard, E. P. Frady, D. B. D. Rubin, S. Sanborn, S. B. Shrestha, F. T. Sommer, and M. Davies, "Efficient neuromorphic signal processing with loihi 2," in *2021 IEEE Workshop on Signal Processing Systems (SiPS)*, pp. 254–259, IEEE, 2021. 39, 42, 104

[166] T. Cooijmans, N. Ballas, C. Laurent, Ç. Gülçehre, and A. C. Courville, "Recurrent batch normalization," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017. 43

[167] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, pp. 8026–8037, 2019. 46

[168] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128× 128 120 db 15 μs latency asynchronous temporal contrast vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, 2008. 46, 61

[169] T. Delbrück, B. Linares-Barranco, E. Culurciello, and C. Posch, "Activity-driven, event-based vision sensors," *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pp. 2426–2429, 2010. 46, 61

[170] W. Fang, Y. Chen, J. Ding, D. Chen, Z. Yu, H. Zhou, and Y. Tian, "and other contributors. spikingjelly," 2020. 47

[171] S. Falkner, A. Klein, and F. Hutter, "Bohb: Robust and efficient hyperparameter optimization at scale," in *International Conference on Machine Learning*, pp. 1437–1446, PMLR, 2018. 47

[172] C. Stöckl and W. Maass, "Optimized spiking neurons can classify images with high accuracy through temporal coding with two spikes," *Nature Machine Intelligence*, vol. 3, no. 3, pp. 230–238, 2021. 57

[173] P. Kirkland, D. Manna, A. Vicente-Sola, and G. Di Caterina, "Unsupervised spiking instance segmentation on event data using stdp features," *IEEE Transactions on Computers*, 2022. 61

[174] J. Lee, T. Delbrück, M. Pfeiffer, P. K. J. Park, C.-W. Shin, H. Ryu, and B.-C. Kang, "Real-time gesture interface based on event-driven processing from stereo silicon retinas," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, pp. 2250–2263, 2014. 61

[175] Y. Bi and Y. Andreopoulos, "Pix2nvs: Parameterized conversion of pixel-domain video frames to neuromorphic vision streams," *2017 IEEE International Conference on Image Processing (ICIP)*, pp. 1990–1994, 2017. 61, 63

[176] D. Gehrig, M. Gehrig, J. Hidalgo-Carri'o, and D. Scaramuzza, "Video to events: Recycling video datasets for event cameras," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3583–3592, 2020. 61, 63

[177] E. Mueggler, H. Rebecq, G. Gallego, T. Delbrück, and D. Scaramuzza, "The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and slam," *The International Journal of Robotics Research*, vol. 36, pp. 142 – 149, 2017. 62

[178] H. Rebecq, D. Gehrig, and D. Scaramuzza, "Esim: an open event camera simulator," in *Proceedings of The 2nd Conference on Robot Learning* (A. Billard, A. Dragan, J. Peters, and J. Morimoto, eds.), vol. 87 of *Proceedings of Machine Learning Research*, pp. 969–982, PMLR, 29–31 Oct 2018. 62, 63

[179] D. L. Manna, A. Vicente-Sola, P. Kirkland, T. J. Bihl, and G. Di Caterina, "Simple and complex spiking neurons: perspectives and analysis in a simple stdp scenario," *Neuromorphic Computing and Engineering*, vol. 2, no. 4, p. 044009, 2022. 62

[180] Y. Xing, G. Di Caterina, and J. Soraghan, "A new spiking convolutional recurrent neural network (scrnn) with applications to event-based hand gesture recognition," *Frontiers in neuroscience*, vol. 14, p. 590164, 2020. 62

[181] A. S. Cassidy, P. Merolla, J. V. Arthur, S. K. Esser, B. Jackson, R. Alvarez-Icaza, P. Datta, J. Sawada, T. M. Wong, V. Feldman, A. Amir, D. B.-D. Rubin, F. Akopyan, E. McQuinn, W. P. Risk, and D. S. Modha, "Cognitive computing building block: A versatile and efficient digital neuron model for neurosynaptic cores," in *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–10, 2013. 62

[182] A. Z. Zhu, Z. Wang, K. Khant, and K. Daniilidis, "Eventgan: Leveraging large scale image datasets for event cameras," in *2021 IEEE International Conference on Computational Photography (ICCP)*, pp. 1–11, IEEE, 2021. 63

[183] D. Joubert, A. Marcireau, N. Ralph, A. Jolley, A. van Schaik, and G. Cohen, "Event camera simulator improvements via characterized parameters," *Frontiers in Neuroscience*, p. 910, 2021. 63

[184] A. Sironi, M. Brambilla, N. Bourdis, X. Lagorce, and R. Benosman, "Hats: Histograms of averaged time surfaces for robust event-based object classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1731–1740, 2018. 63

[185] Y. Bi, A. Chadha, A. Abbas, E. Bourtsoulatze, and Y. Andreopoulos, "Graph-based object classification for neuromorphic vision sensing," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 491–501, 2019. 63

[186] L. Berlincioni, L. Cultrera, C. Albisani, L. Cresti, A. Leonardo, S. Picchioni, F. Becattini, and A. Del Bimbo, "Neuromorphic event-based facial expression recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4109–4119, 2023. 63

[187] Q. Liu, D. Xing, H. Tang, D. Ma, and G. Pan, "Event-based action recognition using motion information and spiking neural networks," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21* (Z.-H. Zhou, ed.), pp. 1743–1749, International Joint Conferences on Artificial Intelligence Organization, 8 2021. Main Track. 63

[188] P. De Tournemire, D. Nitti, E. Perot, D. Migliore, and A. Sironi, "A large scale event-based detection dataset for automotive," *arXiv preprint arXiv:2001.08499*, 2020. 63

[189] E. Perot, P. De Tournemire, D. Nitti, J. Masci, and A. Sironi, "Learning to detect objects with a 1 megapixel event camera," *Advances in Neural Information Processing Systems*, vol. 33, pp. 16639–16652, 2020. 63

[190] M. Gehrig, W. Aarents, D. Gehrig, and D. Scaramuzza, "Dsec: A stereo event camera dataset for driving scenarios," *IEEE Robotics and Automation Letters*, 2021. 63

[191] L. R. Iyer, Y. Chua, and H. Li, "Is neuromorphic mnist neuromorphic? analyzing the discriminative power of neuromorphic datasets in the time domain," *Frontiers in neuroscience*, vol. 15, p. 608567, 2021. 63

[192] A. Vicente-Sola, D. L. Manna, P. Kirkland, G. Di Caterina, and T. J. Bihl, "Keys to accurate feature extraction using residual spiking neural networks," *Neuromorphic Computing and Engineering*, 2022. 64, 65, 69

[193] S. S. Chowdhury, C. Lee, and K. Roy, "Towards understanding the effect of leak in spiking neural networks," *Neurocomputing*, vol. 464, pp. 83–94, 2021. 79

[194] B. Rueckauer, I.-A. Lungu, Y. Hu, M. Pfeiffer, and S.-C. Liu, "Conversion of continuous-valued deep networks to efficient event-driven networks for image classification," *Frontiers in Neuroscience*, vol. 11, 2017. 79

[195] R. Bisaz, A. Travaglia, and C. M. Alberini, "The neurobiological bases of memory formation: from physiological conditions to psychopathology," *Psychopathology*, vol. 47, no. 6, pp. 347–356, 2014. 81

[196] C. Ortega-de San Luis and T. J. Ryan, “Understanding the physical basis of memory: Molecular mechanisms of the engram,” *Journal of Biological Chemistry*, vol. 298, no. 5, 2022. 81

[197] Y. Le and X. Yang, “Tiny imagenet visual recognition challenge,” *CS 231N*, vol. 7, no. 7, p. 3, 2015. 83, 89

[198] H. Jung, J. Ju, M. Jung, and J. Kim, “Less-forgetting learning in deep neural networks,” *arXiv preprint arXiv:1607.00122*, 2016. 84

[199] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari, “End-to-end incremental learning,” in *Proceedings of the European conference on computer vision (ECCV)*, pp. 233–248, 2018. 84

[200] A. Mallya, D. Davis, and S. Lazebnik, “Piggyback: Adapting a single network to multiple tasks by learning to mask weights,” in *Proceedings of the European conference on computer vision (ECCV)*, pp. 67–82, 2018. 84

[201] H. Jin and E. Kim, “Helpful or harmful: Inter-task association in continual learning,” in *European Conference on Computer Vision*, pp. 519–535, Springer, 2022. 84

[202] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, pp. 818–833, Springer, 2014. 89

[203] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, “Decaf: A deep convolutional activation feature for generic visual recognition,” in *International conference on machine learning*, pp. 647–655, PMLR, 2014. 89

[204] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, “Striving for simplicity: The all convolutional net,” *arXiv preprint arXiv:1412.6806*, 2014. 89, 96, 97

[205] A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. Torr, “Riemannian walk for incremental learning: Understanding forgetting and intransigence,” in *Proceedings of the European conference on computer vision (ECCV)*, pp. 532–547, 2018. 90

[206] M. Masana, X. Liu, B. Twardowski, M. Menta, A. D. Bagdanov, and J. van de Weijer, “Class-incremental learning: survey and performance evaluation,” *arXiv preprint arXiv:2010.15277*, 2020. 90

[207] D. Kleyko, D. A. Rachkovskij, E. Osipov, and A. Rahimi, “A survey on hyperdimensional computing aka vector symbolic architectures, part i: Models and data transformations,” *ACM Computing Surveys*, vol. 55, no. 6, pp. 1–40, 2022. 95

[208] A. Brock, S. De, S. L. Smith, and K. Simonyan, “High-performance large-scale image recognition without normalization,” in *International Conference on Machine Learning*, pp. 1059–1071, PMLR, 2021. 98

[209] B. Zhao, X. Xiao, G. Gan, B. Zhang, and S.-T. Xia, “Maintaining discrimination and fairness in class incremental learning,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 13208–13217, 2020. 100

[210] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” *arXiv preprint arXiv:1711.05101*, 2017. 103

[211] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, *et al.*, “Loihi: A neuromorphic manycore processor with on-chip learning,” *Ieee Micro*, vol. 38, no. 1, pp. 82–99, 2018. 104