2023 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, SEPT. 17–20, 2023, ROME, ITALY

MEMORY REPLAY FOR CONTINUAL LEARNING WITH SPIKING NEURAL NETWORKS

Anonymous

Anonymous

ABSTRACT

Two of the most impressive features of biological neural networks are their high energy efficiency and their ability to continuously adapt to varying inputs. On the contrary, the amount of power required to train top-performing deep learning models rises as they become more complex. This is the main reason for the increasing research interest in spiking neural networks, which mimic the functioning of the human brain achieving similar performances to artificial neural networks, but with much lower energy costs. However, even this type of network is not provided with the ability to incrementally learn new tasks, with the main obstacle being catastrophic forgetting. This paper investigates memory replay as a strategy to mitigate catastrophic forgetting in spiking neural networks. Experiments are conducted on the MNIST-split dataset in both class-incremental learning and task-free continual learning scenarios.

***Index Terms*—** Spiking neural networks, continual learning, memory replay

1. INTRODUCTION

In recent years, deep learning models have managed to reach super-human performances in several domains. AlphaGo [1] and GT-Sophy [2] managed to outperform world champions in the respective games, while AlphaFold and its variants [3] reached impressive accuracies in protein structure prediction. However, these achievements come at the price of an increasing amount of computational power required to train such models, as the number of trainable parameters they contain becomes larger [4].

This issue is addressed by spiking neural networks (SNNs) [5], which are a special class of artificial neural networks. The basic computational units of such models are the spiking neurons, which communicate using discrete spike sequences. Similarly to biological neurons, the output is computed by combining the input discrete spikes and it can be either a spike or 0, depending on whether a certain threshold is reached. Including this type of temporal dynamics makes these networks completely event- and data-driven, and parameter updates are only needed when a spike is produced. These characteristics make SNNs much more computationally efficient than artificial neural networks and more suitable for real-time applications.

Despite these advantages, SNNs are also subject to the phenomenon of catastrophic forgetting [6] when trained sequentially on different tasks. This represents a major obstacle to continual learning, which is currently an important branch of research in AI. For this reason, several works propose approaches for mitigating catastrophic forgetting in this type of network. Most of the existing methods are based on parameter regularization [7, 8, 9, 10] and on the optimization process [11], while [12] uses off-line reactivation to consolidate memories from previous tasks. To the best of the author's knowledge, there is no prior work using memory replay to tackle catastrophic forgetting. Despite their high memory requirements, replay-based approaches have proved to significantly attenuate forgetting in artificial neural networks [13, 14, 15, 16, 17]. Consequently, it is interesting to test whether they are effective also in the case of SNNs.

In this work, we implement a simple form of memory replay to investigate whether it might also be a good strategy in SNNs. We conduct experiments on the MNIST-split dataset, considering both class-incremental learning and task-free continual learning scenarios. In summary, the specific contributions of this work are the following:

- We test the effectiveness of memory replay for addressing the problem of catastrophic forgetting in SNNs;
- We compare the improvement in performance obtained through memory replay in two challenging continual learning scenarios, namely class-incremental learning and task-free continual learning, using different memory sizes.

2. RELATED WORK

Computer vision is currently one of the main fields of application of SNNs [18]. Even if in most cases this type of network is applied to relatively small datasets, such as MNIST [19], SNNs have proved to be promising in addressing even more complex tasks, such as ImageNet [20, 21].

Anonymous

979-8-3503-2411-2/23/$31.00 ©2023 IEEE

Despite the good performances achieved by SNNs, these networks also suffer from catastrophic forgetting [6] when trained sequentially on multiple tasks. [8] introduce modified synaptic intelligence by adding a regularization term, called cost per synapse, that penalizes the update of the parameters considered important for previous tasks. The regularization term only affects the classification part of the network, which is non-spiking. The spiking feature extraction part, instead, is pre-trained using STDP [22], a learning rule that changes the strength of synaptic connections based on the temporal correlations of spikes between the connected neurons. Similarly, [9] develop a network consisting of two parts: a supervised convolutional neural network, and an unsupervised STDP classifier. On the contrary, [10] propose controlled forgetting networks completely trained with STDP, inspired by the dopamine signals in the mammalian brain that heterogeneously modulate synaptic plasticity. Instead of using STDP, [7] train a convolutional SNN using backpropagation and propose to adapt the thresholds based on the spiking activity of preceding layers.

A different approach is considered by [11], who presents an optimizer, called GRAPES (Group Responsibility for Adjusting the Propagation of Error Signals), that incorporates principles from biology, including synaptic integration, heterosynaptic competition [23], and synaptic scaling [24].

Finally, [12] train a multi-layer SNN with reinforcement learning adopting a completely different strategy, inspired by the mechanism of memory consolidation that takes place during sleep. The basic idea is to alternate new task training and offline reactivation to force the network synaptic state space to remain close to the previously learned manifold while converging towards the intersection of the manifolds representing old and new tasks.

However, to the best of our knowledge, no replay-based method has been tested on SNNs. The basic idea of memory replay is to try to approximate or recover the distributions of old tasks by storing some kind of information about them. Depending on what is stored, we can identify different types of memory replay. In experience replay [13, 14], the memory buffer is filled in with raw samples from old tasks. To avoid storing large amounts of data, resulting in high memory requirements and low scalability, another possibility is to use generative replay [15, 16], in which samples for old tasks are generated through a generative model. However, even these approaches have several drawbacks, as training generative models is a non-trivial task and such models might also be subject to catastrophic forgetting themselves. A third option is to use feature replay [17], which consists in storing feature-level distributions.

As a starting point in the study of the effectiveness of memory replay in the case of SNNs, we focus on experience replay, and we study the impact of the memory size on the overall performance on all seen tasks.

3. METHODS

3.1. Spiking neural networks

Spiking neural networks (SNNs) [5] represent the third generation of neural network models, having as main computational units spiking neurons. One of the most commonly used neuronal models is the leaky integrate-and-fire (LIF) model [25, 26]. In this type of neuron, the membrane is described by a resistance R and a capacitance C, and a spike is generated at time t_f if the membrane potential reaches a certain threshold u_{thr}. After t_f, the membrane potential is reset to the resting-state potential, u_{rest}, and for $t > t_f$ the dynamics are described by the following equation:

$$\tau \frac{du(t)}{dt} = u(t) - u_{rest} + Ri(t) \tag{1}$$

where $i(t)$ and $u(t)$ are the current and voltage across the membrane, respectively, and $\tau = RC$ is the membrane time constant.

As explained by [27], the discrete-time approximation of the solution to Equation 1 for a constant current input is:

$$U[t] = \beta U[t-1] + (1-\beta) I_{in}[t] \tag{2}$$

where $\beta = e^{\frac{-1}{\tau}}$ is the decay rate of $U[t]$.

The coefficient $(1-\beta)$ can be replaced by a learnable weight W and we replace $WX[t] = I_{in}[t]$, in such a way that $X[t]$ can be considered as an input voltage (spike), that is scaled by W (synaptic conductance) to generate a current injection to the neuron. This results in:

$$U[t+1] = \beta U[t] + WX[t+1] \tag{3}$$

At this point, we need to introduce the reset mechanism, as when the neuron fires, the membrane potential needs to be reset. This can be modeled as:

$$U[t+1] = \beta U[t] + WX[t+1] - S[t] U_{thr} \tag{4}$$

As previously explained, the neuron produces an output spike if $U[t] > U_{thr}$, while its output is 0 otherwise. This behavior can be expressed with the Heaviside step function:

$$S[t] = \Theta(U[t] - U_{thr}) \tag{5}$$

In order to train an SNN made by LIF neurons, we would need to take the derivative $\frac{\partial S}{\partial U}$. Such derivative corresponds to the Dirac Delta function, which is 0 everywhere except at the threshold $U_{thr} = 0$, where it tends to infinity. As a consequence, the gradient will be 0 most of the time and the network will not be able to learn. This problem is called the dead neuron problem. One common approach to solve this issue is to use a surrogate gradient. This means that the Heaviside function is kept unaltered during the forward pass, while it is smoothed during the backward pass to make it differentiable.

Some common smoothing functions are the arctangent, the sigmoid, and the fast sigmoid [28] functions. For brevity reasons, we just report the formulas for the fast sigmoid function, which is the one we use in our experiments and has the following form:

$$S \approx \frac{U}{1 + k\,|U|} \qquad \frac{\partial S}{\partial U} = \frac{1}{(1 + k\,|U|)^2} \tag{6}$$

Finally, to compute the gradient for all timesteps, the backpropagation through time algorithm [29, 30] is used.

3.2. Memory replay

In this paper, we implement a simple form of experience replay, which consists in storing a small number of raw samples from the previous tasks that are then replayed after training on a new task. Our purpose is to try to approximate the distributions of previously seen tasks in order to prevent a drastic drop in performance on those tasks.

In our method, the samples are stored in our memory buffer directly in batches. Specifically, while we iterate over the batches of the training dataset of task i during the training phase, each batch will have a probability of 33% to be stored in our replay buffer until we reach the maximum allowed memory size (N). This avoids looping over the whole dataset after the training of task i to choose the samples to store in memory and it also randomizes the selection process, instead of simply taking the first or the last N samples in the training dataset.

Therefore, while training our network on task i, we store samples to be replayed for task i. Then, before training on a new task $i + 1$, we loop over the replay buffer, revisiting samples from task 0 to task $i - 1$.

A crucial aspect in experience replay, and more generally in memory replay, is the selection of the appropriate number of samples to not have overfitting on the old tasks. For this reason, we make experiments with different sizes for the memory buffer to select the most suitable one.

4. EXPERIMENTS

4.1. Experimental setup

All the code was developed in PyTorch [31], using SNNTorch library [27]. All experiments are conducted on the MNIST dataset [19], which is split up into five distinct binary classification tasks, in such a way that task 1 contains digits 0 and 1, task 2 contains digits 2 and 3, and so on. 10% of the training dataset has been used for validation in the training phase.

The continual learning scenarios we consider are the most challenging ones, namely class-incremental learning and task-free continual learning. The former is characterized by the availability of task indices at training time only. The latter, instead, is characterized by the complete absence of task indices at both training and testing times.

For each of the two scenarios, we developed a different architecture, but the convolutional part is the same for both models. In particular, we use two convolutional layers of 12 and 64 5×5 filters, each followed by *Leaky* neurons from SNNTorch library with β parameter set to 0.99 and having as surrogate function a fast sigmoid with slope set to 25. Then, for the class-incremental learning scenario, we develop a dynamic multi-head architecture. Every time we introduce a new task, a new linear layer with output dimension 2 is added, followed by *Leaky* neurons. At training time, we return just the output of the head added for the task we are training on, in order to not interfere with the previously trained heads. At validation and testing times, instead, we do not provide the task index, so our model will return the concatenation of the outputs of all the heads and we will perform the classification over the total number of seen classes. For the task-free continual learning scenario, we simply use a linear layer with output dimension 10, followed by *Leaky* neurons.

In order to perform the training, we use the cross entropy spike count loss implemented in the SNNTorch library. This function first accumulates the spikes at each timestep and then applies the cross entropy loss function, thus encouraging the correct class to fire at all time steps while suppressing incorrect classes from firing. As optimizer, we used Adam with learning rates $1e-3$ and $1e-4$ for the class-incremental and task-free continual learning scenario, respectively. The batch size is set to 16. All experiments are run 5 times to compare different seeds.

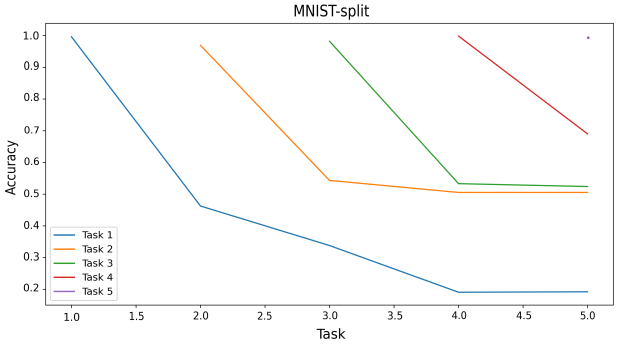
4.2. Class-incremental learning results

In table 1, we show the results obtained in the class incremental learning scenario. As a baseline, we report the accuracy results obtained using naive finetuning, which consists in training the network sequentially on the different tasks without any accouterments. This type of training leads to a final average accuracy on all tasks equal to 43.9%. Then, we report the results obtained by storing 10, 20, 30, 40, 50, and 100 batches of data for each task. It can be seen that without imposing any kind of regularization, storing a larger number of samples per task is not always beneficial. In particular, the highest final accuracy of 51% is reached by storing 40 batches per task. Instead, by storing 100 batches per task the performance decreases by 1.7%. Overall, using experience replay we are able to reduce the accuracy drop by at most 7.1%, which is a significant improvement considering that it was achieved without using any form of regularization. The results obtained with a joint training on all tasks are reported as an upper bound.

Figure 1 shows the evolution of the accuracy on each task as the number of tasks grows. The performance on the last

Method	Task1	Task2	Task3	Task4	Task5
Naive	99.8 ± 0.1	98.7 ± 0.4	70.8 ± 12	63.7 ± 17	43.9 ± 22
MR 10	99.8 ± 0.2	98.8 ± 0.1	73.2 ± 4.7	58.2 ± 6.6	47.9 ± 6.6
MR 20	99.8 ± 0.1	98.9 ± 0.3	73.1 ± 3.9	56.2 ± 2.7	46.9 ± 1.9
MR 30	99.8 ± 0.0	98.3 ± 0.8	74.8 ± 4.7	60.1 ± 5.6	50.0 ± 5.3
MR 40	99.8 ± 0.0	98.8 ± 0.3	75.0 ± 3.1	62.5 ± 5.7	51.0 ± 4.9
MR 50	99.8 ± 0.0	98.1 ± 1.1	79.5 ± 9.6	59.2 ± 6.7	46.7 ± 4.3
MR 100	99.8 ± 0.0	98.7 ± 0.6	73.7 ± 1.9	60.3 ± 4.5	49.3 ± 4.9
Joint					90.1

Table 1. Performance on MNIST-split in the class incremental learning scenario. Column *Task i* reports the average accuracy over all seen tasks up to the i-th task, after training the i-th task.



(a) Result 1

Fig. 1. Example of the evolution of the accuracy on each task in the class-incremental learning scenario.

trained task is always significantly high, close to 100%. After training the network on task i, the accuracy on task $i-1$ drops, reaching values around 50%, for $i = 2, 3, 4$. Instead, after training on task 5, the accuracy on task 4 remains slightly higher, at around 70%.

4.3. Task-free continual learning

In table 2, we report the results obtained for the task-free continual learning scenario. Again, we use as a baseline the naive finetuning, which achieves a final average accuracy over all the tasks of 21.9%. In this scenario, the reduction in accuracy drop registered is more notable than in the class-incremental learning scenario, obtaining 36% of accuracy by storing 100 batches per task, which corresponds to an improvement by 14.1% with respect to naive finetuning. In this case, using a larger memory is more helpful in preventing forgetting. This may be due to the higher intrinsic complexity of this scenario. Nevertheless, even with half the memory (50 batches per task), we are able to improve the accuracy by more than 10%.

5. CONCLUSIONS

In recent years, AI research has achieved successful results in a variety of applications thanks to the development of large models with millions of trainable parameters. However, the great computational capabilities of such models come at the price of an increase in the amount of power required to train them. This issue is partially overcome by spiking neural networks, specifically designed to be energy-efficient by taking inspiration from biological brains. Despite this, even such networks do not have the ability to continuously learn new tasks sequentially, as they are subject to catastrophic forgetting.

In this work, we study the effect of replaying experiences from old tasks in order to mitigate interference from the newly trained ones. We show that memory replay allows the reduction of the drop in performance caused by the overwriting of old tasks. In the class-incremental learning scenario, the highest accuracy is obtained by storing 40 batches per task, and increasing the size of the memory buffer does not bring further improvements. The task-free continual learning scenario, instead, benefits from storing a larger number of samples per task. This difference is related to the highest difficulty intrinsic in this scenario, as task ids are not provided during training or testing.

Future works might try to combine memory replay with other types of methods, such as regularization-based approaches, to try to further improve the average performance on all tasks. By combining different methods, it might be sufficient to store a smaller number of samples per task to be replayed to achieve comparable or even better performances. Another interesting line of research would be the adaptation of explainability methods to SNNs in order to have an informed criterion to choose the examples to store in memory. In this way, we could choose the smallest number of samples that contain the most influential information about each task. Several works in the literature propose methods for mitigating catastrophic forgetting in artificial neural networks that exploit explainable AI [32, 33, 34], and they manage to achieve comparable and, in some cases, higher performances than other state-of-the-art approaches.

Method	Task1	Task2	Task3	Task4	Task5
Naive	99.6 ± 0.0	49.1 ± 1.0	34.7 ± 1.6	25.6 ± 0.4	21.9 ± 1.6
MR 10	99.6 ± 0.2	50.4 ± 1.1	34.7 ± 1.4	28.6 ± 1.3	24.5 ± 2.7
MR 20	99.6 ± 0.1	50.6 ± 1.2	37.3 ± 2.8	30.0 ± 2.9	27.1 ± 2.1
MR 30	99.6 ± 0.4	54.8 ± 3.7	43.7 ± 4.4	36.0 ± 2.3	28.1 ± 1.5
MR 40	99.6 ± 0.9	55.3 ± 4.2	46.4 ± 6.3	36.0 ± 3.9	28.3 ± 5.7
MR 50	99.6 ± 0.0	60.8 ± 4.8	50.9 ± 8.7	39.6 ± 3.8	33.1 ± 5.6
MR 100	99.6 ± 0.1	63.3 ± 3.0	52.7 ± 2.6	45.7 ± 4.1	36.0 ± 9.1
Joint					90.1

Table 2. Performance on MNIST-split in the task-free continual learning scenario. Column *Task i* reports the average accuracy over all seen tasks up to the i-th task, after training the i-th task.

6. REFERENCES

[1] David Silver, Aja Huang, Christopher Maddison, Arthur Guez, Laurent Sifre, George Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–489, 01 2016.

[2] Peter R Wurman, Samuel Barrett, Kenta Kawamoto, James MacGlashan, Kaushik Subramanian, Thomas J Walsh, Roberto Capobianco, Alisa Devlic, Franziska Eckert, Florian Fuchs, et al., "Outracing champion gran turismo drivers with deep reinforcement learning," *Nature*, vol. 602, no. 7896, pp. 223–228, 2022.

[3] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon Kohl, Andrew Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, and Demis Hassabis, "Highly accurate protein structure prediction with alphafold," *Nature*, vol. 596, pp. 1–11, 08 2021.

[4] Neil C. Thompson, Kristjan Greenewald, Keeheon Lee, and Gabriel F. Manso, "The computational limits of deep learning," 2022.

[5] Wolfgang Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

[6] Michael McCloskey and Neal J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," *Psychology of Learning and Motivation - Advances in Research and Theory*, vol. 24, no. C, pp. 109–165, Jan. 1989.

[7] Ilyass Hammouamri, Timothée Masquelier, and Dennis G. Wilson, "Mitigating catastrophic forgetting in spiking neural networks through threshold modulation," 2022.

[8] Ruthvik Vaila, John Chiasson, and Vishal Saxena, "Continuous learning in a single-incremental-task scenario with spike features," *International Conference on Neuromorphic Systems 2020*, 2020.

[9] Irene Muñoz-Martín, Stefano Bianchi, Giacomo Pedretti, Octavian Melnic, Stefano Ambrogio, and Daniele Ielmini, "Unsupervised learning to overcome catastrophic forgetting in neural networks," *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, vol. 5, pp. 58–66, 2019.

[10] Jason M. Allred and Kaushik Roy, "Controlled forgetting: Targeted stimulation and dopaminergic plasticity modulation for unsupervised lifelong learning in spiking neural networks," *Frontiers in Neuroscience*, vol. 14, 2019.

[11] Giorgia Dellaferrera, Stanisław Woźniak, G. Indiveri, Angeliki Pantazi, and Evangelos Eleftheriou, "Introducing principles of synaptic integration in the optimization of deep neural networks," *Nature Communications*, vol. 13, 2022.

[12] Ryan Golden, Jean Erik Delanois, Pavel Sanda, and Maxim Bazhenov, "Sleep prevents catastrophic forgetting in spiking neural networks by forming a joint synaptic weight representation," *PLOS Computational Biology*, vol. 18, 2019.

[13] David Lopez-Paz and Marc'Aurelio Ranzato, "Gradient episodic memory for continual learning," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2017, NIPS'17, p. 6470–6479, Curran Associates Inc.

[14] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, , and Gerald Tesauro, "Learning to learn without forgetting by maximizing transfer

and minimizing interference," in *International Conference on Learning Representations*, 2019.

[15] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim, "Continual learning with deep generative replay," in *NIPS*, 2017.

[16] Chenshen Wu, Luis Herranz, Xialei Liu, Yaxing Wang, Joost van de Weijer, and Bogdan Raducanu, "Memory replay gans: Learning to generate new categories without forgetting," in *Neural Information Processing Systems*, 2018.

[17] Marco Toldo and Mete Ozay, "Bring evanescent representations to life in lifelong class incremental learning," *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16711–16720, 2022.

[18] Kashu Yamazaki, Viet-Khoa Vo-Ho, Darshan Bulsara, and Ngan Le, "Spiking neural networks and their applications: A review," *Brain Sciences*, vol. 12, no. 7, 2022.

[19] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[20] Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu, "Conversion of continuous-valued deep networks to efficient event-driven networks for image classification," *Frontiers in neuroscience*, vol. 11, pp. 682, 2017.

[21] Abhronil Sengupta, Yuting Ye, Robert Wang, Chiao Liu, and Kaushik Roy, "Going deeper in spiking neural networks: Vgg and residual architectures," *Frontiers in neuroscience*, vol. 13, pp. 95, 2019.

[22] Sen Song, Kenneth D. Miller, and L. F. Abbott, "Competitive hebbian learning through spike-timing-dependent synaptic plasticity," *Nature Neuroscience*, vol. 3, pp. 919–926, 2000.

[23] Craig H. Bailey, Maurizio Giustetto, Yan-You Huang, Robert D. Hawkins, and Eric R. Kandel, "Is heterosynaptic modulation essential for stabilizing hebbian plasiticity and memory," *Nature Reviews Neuroscience*, vol. 1, pp. 11–20, 2000.

[24] Gina G. Turrigiano, Kenneth Raj Leslie, Niraj S. Desai, Lana C. Rutherford, and Sacha B. Nelson, "Activity-dependent scaling of quantal amplitude in neocortical neurons," *Nature*, vol. 391, pp. 892–896, 1998.

[25] Louis Lapicque, "Recherches quantitatives sur l'excitation electrique des nerfs traitee comme une polarization," *Journal de physiologie et de pathologie générale*, vol. 9, pp. 620–635, 1907.

[26] Larry 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.

[27] Jason Kamran Eshraghian, Max Ward, Emre O. Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Bennamoun, Doo Seok Jeong, and Wei D. Lu, "Training spiking neural networks using lessons from deep learning," *ArXiv*, vol. abs/2109.12894, 2021.

[28] Friedemann Zenke and Tim Vogels, "The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks," *Neural Computation*, vol. 33, pp. 1–27, 01 2021.

[29] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[30] Paul J. Werbos, "Backpropagation through time: What it does and how to do it," *Proc. IEEE*, vol. 78, pp. 1550–1560, 1990.

[31] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala, *PyTorch: An Imperative Style, High-Performance Deep Learning Library*, Curran Associates Inc., Red Hook, NY, USA, 2019.

[32] Sami Ede, Serop Baghdadlian, Leander Weber, An Thai Nguyen, Dario Zanca, Wojciech Samek, and Sebastian Lapuschkin, "Explain to not forget: Defending against catastrophic forgetting with xai," in *International Cross-Domain Conference on Machine Learning and Knowledge Extraction*, 2022.

[33] Sayna Ebrahimi, Suzanne Petryk, Akash Gokul, William Gan, Joseph E. Gonzalez, Marcus Rohrbach, and trevor darrell, "Remembering for the right reasons: Explanations reduce catastrophic forgetting," in *International Conference on Learning Representations*, 2021.

[34] Gobinda Saha and Kaushik Roy, "Saliency guided experience packing for replay in continual learning," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, January 2023, pp. 5273–5283.