DOI: 10.34749/3061-1466.2026.51

Linearized Bregman Iterations for Sparse Spiking Neural Networks

Daniel Windhager
Silicon Austria Labs
Linz, Austria
daniel.windhager@silicon-austria.com

Bernhard A. Moser*
Software Competence Center Hagenberg
Hagenberg, Austria
bernhard.moser@scch.at

Michael Lunglmayr
Institute of Signal Processing
Johannes Kepler University
Linz, Austria
michael.lunglmayr@jku.at

Abstract

Spiking Neural Networks (SNNs) offer an energy efficient alternative to conventional Artificial Neural Networks (ANNs) but typically still require a large number of parameters. This work evaluates *Linearized Bregman Iterations* (LBI) as an optimizer for training SNNs, utilizing the algorithm's ability to enforce sparsity through iterative minimization of the Bregman distance and proximal soft thresholding updates. To improve convergence and generalization, we employ the *AdaBreg* optimizer, a momentum and bias corrected Bregman variant of Adam. Experiments on three established neuromorphic benchmarks, i.e. the Spiking Heidelberg Digits (SHD), the Spiking Speech Commands (SSC), and the Permuted Sequential MNIST (PSMNIST) datasets, show that LBI based optimization reduces the number of active parameters by about 50% while maintaining accuracy comparable to models trained with the Adam optimizer, demonstrating the potential of convex sparsity inducing methods for efficient neuromorphic learning.

1 Introduction

Sparsity in neural networks is an important and ongoing research field Hoefler et al. [2021]. In most neural network architectures sparsity refers to the weight matrices of the model being sparsely populated. This has the effect of sparsifying connections of the network (i.e. a connection with weight 0 is disconnected). For classical GPU implementations, however, having sparse weights can sometimes actually lead to increased computational overhead due to memory organization problems and unbalanced workloads Zaharia et al. [2020]. In contrast, for edge AI implementations where network structures are directly mapped to hardware, e.g. as in Windhager et al. [2025], the benefits are considerably higher as the sparsity of weights can be utilized and exploited more easily.

Training machine learning to produce sparse weight matrices is in essence a sparse optimization problem. In practice, sparsity is often achieved through heuristic approaches such as pruning Hoefler et al. [2021], even though there exist mathematically founded algorithms that can provably converge to sparse optimal solutions. A number of these algorithms is based on so-called linearized Bregman iterations, which have also been proposed for sparse estimation Osher et al. [2005], Yin et al. [2008] using concepts similar to Douglas–Rachford splitting Combettes and Eckstein [1992]. Their efficiency

*double affiliation with Institute of Signal Processing, Johannes Kepler University Linz

The Third Austrian Symposium on AI, Robotics, and Vision (AIROV26).

This work is licensed under CC BY 4.0, excluding materials attributed to other sources or explicitly excluded.

and stability for sparse estimation have been demonstrated repeatedly Hu and Chklovskii [2014], Gebhard et al. [2018], and more recent work has shown their suitability for training sparse deep neural networks, in many cases outperforming heuristic solutions Bungert et al. [2022].

In this work, we investigate how linearized-Bregman-based sparse learning performs for spiking neural networks. Specifically, we evaluate both feedforward and recurrent SNN architectures on established neuromorphic benchmarks and analyze how the regularization parameter λ influences sparsity and model accuracy.

2 Linearized Bregman Iterations

Sparse optimization problems often include a regularization term $J(\boldsymbol{\theta})$ based on the ℓ_1-norm to promote sparsity. However, the non-smoothness of the ℓ_1-norm causes standard gradient-based optimization to fail in regions where the gradient is undefined. A key property that enables a well-behaved optimization framework in such cases is the *convexity* of the regularization function J. For convex but possibly non-smooth functions, the concept of a gradient is replaced by a *sub-differential*, which generalizes differentiation to non-smooth settings. Sub-differentials, however, can not directly be used in steepest-decent like algorithms as they require a defined gradient at each step.

To handle such cases, Bregman iterations iteratively minimize the *Bregman distance* Bregman [1967]

$$D_J(x, y) = J(x) - J(y) - \langle \nabla J(y), x - y \rangle,$$

rather than minimizing the composite cost function directly. In this formulation, $\nabla J(y)$ represents an element of the *sub-differential* of J at point y, denoted $\partial J(y)$. For convex J, the sub-differential $\partial J(y)$ is a non-empty, convex set that captures all possible slopes of local supporting hyperplanes to J at y. For example, when $J(x) = |x|$, the sub-differential at $x = 0$ is the interval $[-1, 1]$. This interpretation allows the Bregman distance to generalize classical gradient-based methods to convex but non-differentiable regularizers, enabling optimization for the ℓ_1-norm and similar sparsity-promoting terms.

Because the resulting subproblems rarely admit closed-form solutions for sparse regularizers Yin et al. [2008], the *Linearized Bregman Iteration* (LBI) method provides an efficient linearized approximation. LBI introduces auxiliary "shadow" variables $\mathbf{v}$ corresponding to the model parameters $\boldsymbol{\theta}$. At iteration t, the updates can be expressed as

$$\mathbf{v}^{(t+1)} = \mathbf{v}^{(t)} + \mu \nabla L(\boldsymbol{\theta}^{(t)}, B), \qquad \boldsymbol{\theta}^{(t+1)} = \text{prox}_J(\mathbf{v}^{(t+1)}),$$

where μ denotes the step size and prox_J represents the proximal operator associated with the convex regularization term J.

For the commonly used sparse regularizer $J(\boldsymbol{\theta}) = \lambda \|\boldsymbol{\theta}\|_1$, the proximal operator corresponds to the elementwise *soft-thresholding* function,

$$\text{prox}_{\lambda\|\boldsymbol{\theta}\|_1}(\boldsymbol{\theta}) = \left[\text{prox}_{\lambda\|\boldsymbol{\theta}\|_1}(\theta_i)\right]_i \tag{1}$$

$$\text{prox}_{\lambda\|\boldsymbol{\theta}\|_1}(\theta_i) = \text{sign}(\theta_i)\ \max(0, |\theta_i| - \lambda), \tag{2}$$

which suppresses small weight values and drives many parameters exactly to zero, thereby yielding sparse network representations.

This mechanism makes LBI particularly well suited for training models such as Spiking Neural Networks, where convex sparse regularization aligns well with the need for energy-efficient, low-parameter inference on neuromorphic hardware.

2.1 AdaBreg Optimization

While classical Linearized Bregman Iterations provide an effective framework for promoting sparsity, their convergence can be slow when applied to large-scale neural network training. To improve adaptation and generalization, we adopt the AdaBreg optimizer introduced by Bungert et al. [2022], which extends the Bregman iteration concept by incorporating adaptive moment estimation in analogy to the Adam optimizer Kingma and Ba [2017].

AdaBreg inherits the shadow-variable formulation of the linearized Bregman framework while maintaining separate exponential moving averages of the first and second moments of the gradient.

Let $\mathbf{m}_t$ and $\mathbf{s}_t$ denote the biased estimates of the mean and variance of the stochastic gradient $\nabla L(\boldsymbol{\theta}_t, B)$ at iteration t. For $t = 0$, both $\mathbf{m}_t$ and $\mathbf{s}_t$ may be set to $\mathbf{0}$ and 0 respectively. The update rules can then be summarized as

$$\begin{aligned}
\mathbf{m}_{t+1} &= \beta_1 \mathbf{m}_t + (1-\beta_1)\nabla L(\boldsymbol{\theta}_t, B), \\
\mathbf{s}_{t+1} &= \beta_2 \mathbf{s}_t + (1-\beta_2)\big(\nabla L(\boldsymbol{\theta}_t, B)\big)^2, \\
\mathbf{v}_{t+1} &= \mathbf{v}_t + \mu \frac{\widehat{\mathbf{m}}_{t+1}}{\sqrt{\widehat{\mathbf{s}}_{t+1}} + \epsilon}, \\
\boldsymbol{\theta}_{t+1} &= \operatorname{prox}_{\lambda\|\boldsymbol{\theta}\|_1}(\mathbf{v}_{t+1}),
\end{aligned}$$

where μ denotes the learning rate, $\widehat{\mathbf{m}}_{t+1}$ and $\widehat{\mathbf{s}}_{t+1}$ represent bias-corrected moment estimates, and ϵ is a small numerical constant for stability.

In practice, the optimizer can be seamlessly integrated into existing deep learning frameworks such as PyTorch by substituting the Adam optimizer with its AdaBreg counterpart, requiring no structural changes to the network implementation.

3 Results

3.1 Setup and configurations

All of the results presented in this work were obtained by training the networks using the AdaBreg algorithm introduced by Bungert et al. [2022], which is a Bregman version of the Adam algorithm Kingma and Ba [2017] that includes momentum and a bias correction term. AdaBreg combines the adaptive moment estimation of Adam with the sparsity inducing properties of Linearized Bregman Iterations, enabling direct integration of convex regularization into the optimization process. The reason for choosing AdaBreg over the standard linearized Bregman iteration based algorithm with momentum (LinBreg) is the better performance and generalization capability as reported by the original authors. Unless otherwise stated, the same sets of hyperparameters and initialization conditions were used across all experiments for comparability.

For the datasets, three of the most common datasets among the neuromorphic community were chosen, namely Spiking Heidelberg Digits (SHD), Spiking Speech Commands (SSC) and the Permuted Sequential MNIST (PSMNIST) dataset, along with the neuron network models presented by Queant et al. [2025], who at the time of writing hold the record for the top performing SNN on the SSC dataset. These datasets jointly cover a broad range of temporal complexities, from short event based auditory signals (SHD) to long sequential patterns (PSMNIST), providing a balanced testbed for evaluating sparsity effects across different task domains.

Table 1: Number of neurons and layers of the networks used for evaluations of Linearized Bregman iterations on the SHD, SSC and PSMNIST datasets.

Dataset	Inputs	Hidden Layer 1	Hidden Layer 2	Hidden Layer 3	Outputs
SHD	140[§]	256	256[†]	-	20
SSC	140[§]	256[†]	256[†]	256[†]	35[‡]
PSMNIST	1	64[†]	212[†]	212[†]	10[‡]

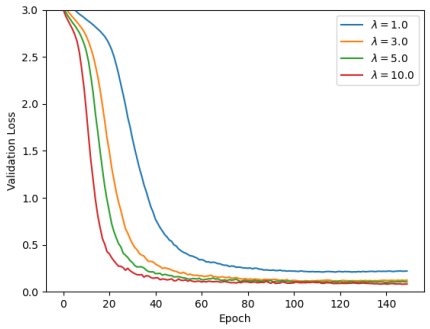
[§] Inputs are reduced from the original 700 inputs, by binning with a factor of 5.
[†] Recurrent layer with axonal delays in the recurrent path.
[‡] Outputs from last linear layer are used directly, without LIF activation.

The networks chosen for the datasets consist of either three or four layers with different feature sizes and configurations. All networks use the Leaky Integrate and Fire (LIF) neuron model. For the SHD dataset, the first and last layers of the network are simple feedforward SNN layers without any delay, while the middle layer is a recurrent layer with learned axonal delays in the recurrent path. The networks for the SSC and PSMNIST datasets each consist of three recurrent SNN layers with learned axonal delays in their recurrent paths, followed by a final linear layer without LIF activation. A simplified overview of the networks can be seen in Tab. 1, while further architectural details, including delay learning mechanisms can be found in Queant et al. [2025].

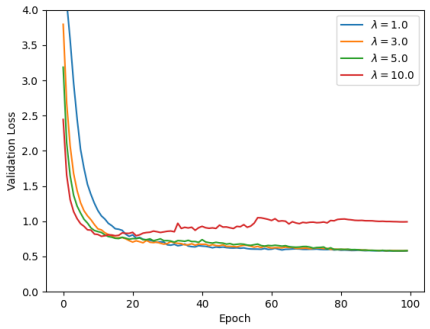
3.2 Performance with learning rate schedulers

The loss on the validation sets can be seen in Fig. 1, with the different colored curves indicating various values for the parameter λ, which controls the sparsity of the solution. These results indicate that larger λ values also introduce a beneficial regularization effect, leading to faster initial convergence and smoother loss trajectories at the early stages of training. This behavior is consistent across datasets and highlights the dual role of λ as both a sparsity and regularization parameter.

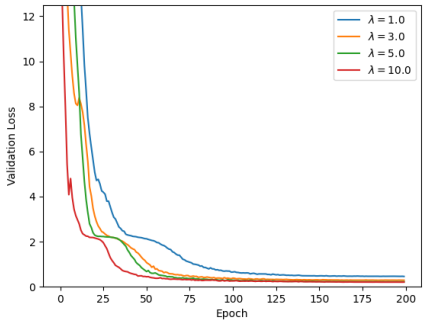
LBI seems to be more sensitive to higher learning rates, especially when used in conjunction with learning rate schedulers, which might be caused by the inherent stagnation phases that occur during training with Linearized Bregman iterations. In fact, choosing a high enough learning rate, paired with large values of the sparsity controlling parameter λ, causes the training to diverge after a few epochs. The onset of this divergent behaviour can be seen in Fig. 1b for $\lambda = 10$, which is due to λ being slightly too high for the chosen learning rate.



(a) Loss curve for SHD dataset, when trained with the OneCycleLR scheduler from PyTorch for 150 epochs, with an initial learning rate of $5 \cdot 10^{-3}$.



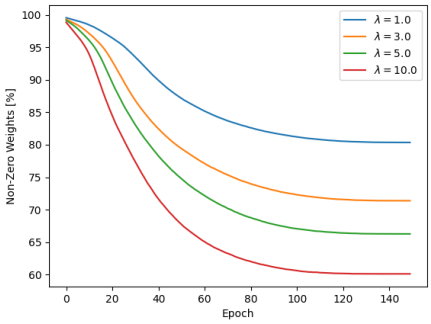
(b) Loss curve for SSC dataset, when trained with the OneCycleLR scheduler from PyTorch for 100 epochs, with an initial learning rate of $1 \cdot 10^{-3}$. Onset of divergent behaviour can be seen for $\lambda = 10$.



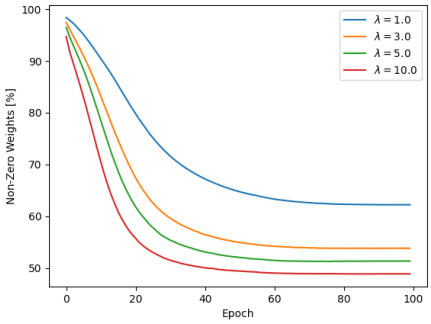
(c) Loss curve for PSMNIST dataset, when trained with the OneCycleLR scheduler from PyTorch for 200 epochs, with an initial learning rate of $1 \cdot 10^{-3}$.

Figure 1: Loss curves for the training on SHD, SSC and PSMNIST dataset with different values for λ. All curves were averaged over three separate training runs with different seeds.

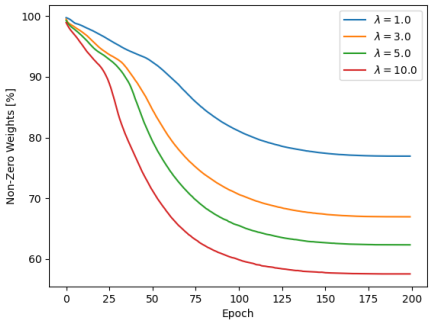
The goal of Linearized Bregman training is to produce highly sparse weight matrices, i.e. containing a large fraction of zero entries. As can be seen in Fig. 2, the number of non-zero parameters across the entire network does indeed decrease monotonically as training progresses. Much like the loss curves, the sparsity level increases rapidly during early training epochs before reaching a plateau, a behavior characteristic of Bregman iterations which preferentially eliminate less important features in the initial optimization phases.



(a) Progression of number of non-zero weights for the neural network trained on the SHD dataset.



(b) Progression of number of non-zero weights for the neural network trained on the SSC dataset.



(c) Progression of number of non-zero weights for the neural network trained on the PSMNIST dataset.

Figure 2: Number of non-zero values in networks for SHD, SSC and PSMNIST datasets during the training process plotted as a function of current epoch for different λ values. Results represent the mean across three independent training runs.

Since the parameter λ clearly influences both the best achieved accuracy and the achieved sparsity of the network, a natural question is the optimal selection of λ. Fig. 3 shows the best validation accuracy achieved across multiple training runs for all three datasets, plotted as a function of the chosen λ value. These results indicate that the choice of λ must be made depending on the learning rate and the chosen dataset. For SHD and SSC a slightly higher value of λ is beneficial, while it results in worse performance for the PSMNIST dataset.

3.3 Performance without learning rate schedulers

Since learning rate schedulers impact the training process and can even cause training divergence when coupled with a high enough learning rate, the previous experiments were repeated without any learning rate scheduling during the training. The resulting loss curves can be seen in Fig. 4.

From the subfigures in Fig. 4 it is apparent that the achieved accuracy is largely unaffected by the presence or absence of learning rate schedulers, further substantiating the hypothesis that the initial divergent behaviour was due to a too high learning rate. Furthermore, the number of non-zero weights in the network, i.e., the sparsity level, also remained largely unaffected by the presence or absence of schedulers. These sparsity curves are therefore omitted here for conciseness, as they closely mirror those shown in Fig. 2.

The achieved best validation accuracy across all datasets without learning rate scheduling, plotted as a function of the chosen λ value, can be seen in Fig. 5. These results demonstrate that without

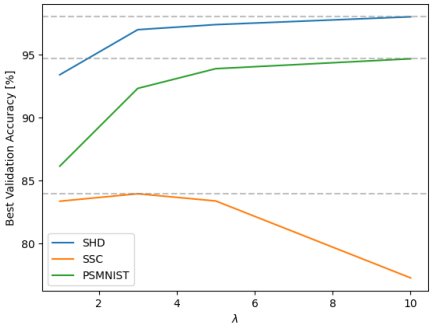
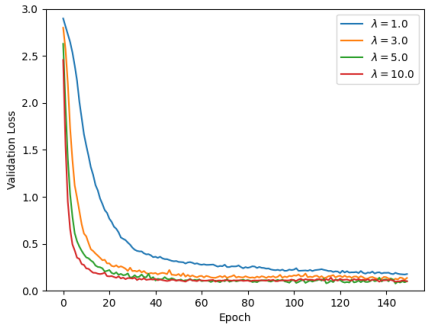
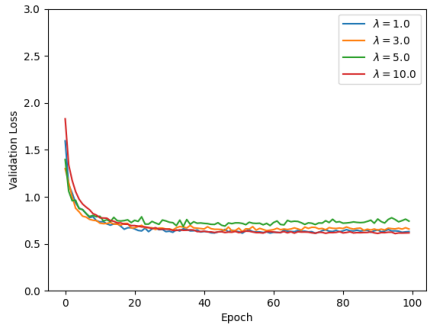


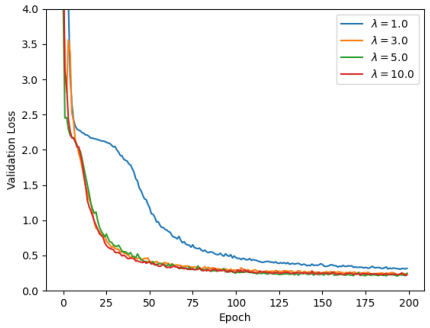
Figure 3: Peak validation accuracy across SHD, SSC, and PSMNIST datasets as a function of regularization parameter λ, averaged over multiple training runs. The optimal λ is slightly higher for SHD and PSMNIST, while a lower value of λ achieves the best results for SSC.



(a) Loss curve for SHD dataset, when trained without a learning rate scheduler for 150 epochs, with an initial learning rate of $2 \cdot 10^{-4}$.



(b) Loss curve for SSC dataset, when trained without a learning rate scheduler for 100 epochs, with an initial learning rate of $5 \cdot 10^{-4}$.



(c) Loss curve for PSMNIST dataset, when trained without a learning rate scheduler for 200 epochs, with an initial learning rate of $1 \cdot 10^{-4}$.

Figure 4: Loss curves for the training on SHD, SSC and PSMNIST dataset with different values for λ. All curves represent means over three independent training runs with different random seeds.

learning rate scheduling, a higher value of λ may be chosen sometimes (see results for PSMNIST in Fig. 3), thus potentially increasing regularization and performance on unseen data, although the effect seems comparatively small when comparing the performance on the test sets.

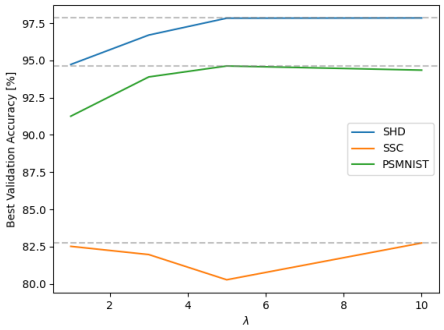


Figure 5: Peak validation accuracy without learning rate scheduling across SHD, SSC, and PSMNIST datasets versus regularization parameter λ, averaged over multiple training runs.

4 Performance compared to training with Adam

The baseline models from Queant et al. [2025] were trained using the well-known Adam optimizer Kingma and Ba [2017]. A direct comparison of test set performance between their results and ours is provided in Tab. 2. This comparison reveals that Linearized Bregman iterations (AdaBreg) achieve accuracies within 0.5–1.5% of the Adam baseline across all three datasets, despite limited hyperparameter tuning. When combined with the observed $\approx 50\%$ reduction in active parameters (cf. Fig. 2), this performance gap appears acceptable for sparsity-constrained neuromorphic applications.

Table 2: Test set accuracy comparison. Baseline results from Queant et al. [2025] (Adam optimizer) versus AdaBreg results with and without learning rate scheduling across SHD, SSC, and PSMNIST datasets.

Dataset	SHD	SSC	PSMNIST
Queant et al. [2025]	93.39%	82.58%	96.21%
Ours	92.98%	81.86%	95.59%
Ours (no LR scheduling)	92.28%	80.67%	95.11%

5 Conclusion

This work demonstrates the practical viability of Linearized Bregman Iterations (LBI) as an optimizer for Spiking Neural Networks (SNNs). Across three established neuromorphic benchmarks (SHD, SSC, PSMNIST), LBI-based training (AdaBreg) achieves competitive accuracy within 0.5–1.5% of Adam baselines while reducing the number of active parameters by approximately 50%.

Further performance gains appear achievable through systematic hyperparameter optimization. Notably, LBI integrates seamlessly into existing PyTorch workflows—requiring only a one-line optimizer replacement (Adam $\rightarrow$ AdaBreg)—dramatically lowering the adoption barrier for sparsity-aware SNN training.

These findings highlight a critical hardware-software co-design opportunity: while sparse SNN training is now readily accessible, neuromorphic hardware must evolve to fully exploit this parameter efficiency for energy-constrained edge deployments.

Acknowledgements

This work was supported by (1) the 'University SAL Labs' initiative of Silicon Austria Labs (SAL) and its Austrian partner universities for applied fundamental research for electronic based systems, and (2) the COMET Programme via SCCH funded by the Austrianministries BMIMI, BMWET, and the State of UpperAustria, (3) the COMET-K2 "Center for Symbiotic Mechatronics" of the Linz Center of Mechatronics (LCM), funded by the Austrian federal government and the federal state of Upper Austria.

The research reported in this paper has also been partly funded by the European Union's Horizon 2020 research and innovation program within the framework of Chips Joint Undertaking (Grant No. 101112268). This work has been supported by Silicon Austria Labs (SAL) owned by the Republic of Austria, the Styrian Business Promotion Agency (SFG), the federal state of Carinthia, the Upper Austrian Research (UAR), and the Austrian Association for the Electric and Electronics Industry (FEEI).

References

L. M. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7(3):200–217, 1967.

L. Bungert, T. Roith, D. Tenbrinck, and M. Burger. A bregman learning framework for sparse neural networks. *Journal of Machine Learning Research*, 23(192):1–43, 2022. URL http://jmlr.org/papers/v23/21-0545.html.

P. L. Combettes and J. Eckstein. On the Douglas–Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55(1-3):293–318, 1992.

A. Gebhard, M. Lunglmayr, and M. Huemer. Investigations on sparse system identification with $$l_0$$-lms, zero-attracting lms and linearized bregman iterations. In R. Moreno-Díaz, F. Pichler, and A. Quesada-Arencibia, editors, *Computer Aided Systems Theory – EUROCAST 2017*, pages 161–169, Cham, 2018. Springer International Publishing. ISBN 978-3-319-74727-9.

T. Hoefler, C.-J. Ng, T. Yoon, P. Yu, M. Low, P. Lee, H. de Kruijf, and M. Zaharia. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Communications of the ACM*, 64(12):82–90, 2021. doi: 10.1145/3546258.3546499. URL https://dl.acm.org/doi/abs/10.5555/3546258.3546499.

T. Hu and D. B. Chklovskii. Sparse lms via online linearized bregman iteration. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7213–7217, 2014. doi: 10.1109/ICASSP.2014.6855000.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin. An iterative regularization method for total variation-based image restoration. *Multiscale Modeling and Simulation*, 4(2):460–489, 2005.

A. Queant, U. Rançon, B. R. Cottereau, and T. Masquelier. Delrec: learning delays in recurrent spiking neural networks, 2025. URL https://arxiv.org/abs/2509.24852.

D. Windhager, L. Ratschbacher, B. Moser, and M. Lunglmayr. Mineuron: Minimal neuron realization for fast fpga snn inference using logic optimization. In *Proceedings of the IEEE International Conference on Image Processing (ICIP) 2025*, Sept. 2025.

W. Yin, S. Osher, D. Goldfarb, and J. Darbon. Bregman iterative algorithms for ℓ_1-minimization with applications to compressed sensing. *SIAM Journal on Imaging Sciences*, 1(1):143–168, 2008. doi: 10.1137/070703983.

M. Zaharia et al. Sparse gpu kernels for deep learning. *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2020. URL https://people.eecs.berkeley.edu/~matei/papers/2020/sc_sparse_gpu.pdf.