SPARSE AND WIDE LINEAR RNNS ARE AT THE EFFICIENCY-PERFORMANCE PARETO FRONT

Alessandro Pierro[1,2,*,+] **Steven Abreu**[1,3,*] **Jonathan Timcheck**[1] **Philipp Stratmann**[1]

Sumit Bam Shrestha[1]

[1]Neuromorphic Computing Lab, Intel Corporation, USA
[2]Institute of Informatics, LMU Munich, Germany
[3]Bernoulli Institute & CogniGron, University of Groningen, Netherlands
[*]Equal contribution
[+]Corresponding author: alessandro.pierro@intel.com

ABSTRACT

Linear recurrent neural networks enable powerful long-range sequence modeling with constant memory usage and time-per-token during inference. These architectures hold promise for streaming applications at the edge, but deployment in resource-constrained environments requires hardware-aware optimizations to minimize latency and energy consumption. In this paper, we investigate the effectiveness of unstructured sparsity–both in weights and activations–at reducing the computational demand of linear RNNs, as well as its combination with quantization. We find that highly sparse linear RNNs consistently achieve better efficiency-performance trade-offs than dense baselines, with $2\times$ less compute and 36% less memory at iso-accuracy, and quantizing a sparse-and-wide network leads to lower performance degradation. When quantized to fixed-point arithmetic and deployed on the Intel Loihi 2 neuromorphic chip, sparse models demonstrate $42\times$ lower latency and $149\times$ lower energy consumption compared to an iso-accuracy dense model on an edge GPU, providing hardware validation to the theoretical gains of unstructured sparsity.

1 INTRODUCTION

Linear Recurrent Neural Networks (LRNNs) have emerged as powerful primitives for efficient sequence modeling with constant memory and linear runtime (Smith et al., 2023; Gu et al., 2021), making them ideal for low-latency, edge applications like audio denoising and keyword spotting (Timcheck et al., 2023; Warden, 2018). While these models are efficient by design, further gains are possible through model compression–which remains under-explored for LRNNs.

In this work, we explore the potential of unstructured sparsity–in weights and activations–and fixed-point quantization for the compression of LRNNs. Theoretical studies have shown that wider sparse layers outperform dense layers with the same parameter count (Golubeva et al., 2021; Chang et al., 2021), with unstructured sparsity (often $\geq 90\%$) representing an upper bound on potential gains (Liu & Wang, 2023; Mishra et al., 2021; Han et al., 2015). While GPUs cannot fully exploit unstructured sparsity, we use neuromorphic hardware to realize the efficency gains from highly sparse neural networks. Specifically, we explore three research questions. **(1)** *Can we train LRNNs with high weight and activation sparsity while retaining task accuracy?*, **(2)** *Do highly sparse LRNNs outperform dense baselines across inference compute budgets?*, **(3)** *What is the combined effect of sparsity and fixed-point quantization on accuracy for LRNNs?*. We provide a positive answer to Question **(1)**, and present positive evidence for Questions **(2)** and **(3)**.

2 BACKGROUND

Linear Recurrent Neural Networks Recurrent neural networks (RNNs) are a class of neural networks designed for processing sequential data by maintaining hidden states that capture temporal dependencies. Linear RNNs (LRNNs) distinguish themselves through their linear dynamics, which

enables parallelization over the sequence length and, thus, efficient training. It has been shown that the network's recurrent weight matrix can be diagonalized in the complex domain without loss of generality (Orvieto et al., 2024; Gu et al., 2022) We use this diagonal formulation of linear RNNs, with the state $\mathbf{x}_k \in \mathbb{C}^N$ and output $\mathbf{y}_k \in \mathbb{R}^M$ evolving as:

$$\mathbf{x}_k = \mathrm{diag}(\bar{\mathbf{A}}) \otimes \mathbf{x}_{k-1} + \bar{\mathbf{B}}^\top \mathbf{u}_k \tag{1}$$

$$\mathbf{y}_k = \mathrm{Re}\left[\bar{\mathbf{C}}^\top \mathbf{x}_k\right] + \mathrm{diag}(\bar{\mathbf{D}}) \otimes \mathbf{u}_k \tag{2}$$

where $\otimes$ denotes the Hadamard product, $\mathbf{u}_k \in \mathbb{R}^M$ is the input sequence, $\mathrm{diag}(\bar{\mathbf{A}}) \in \mathbb{C}^N$ are the diagonal recurrent weights, $\bar{\mathbf{B}}^T \in \mathbb{C}^{M \times N}$ are the input weights, $\bar{\mathbf{C}}^T \in \mathbb{C}^{N \times M}$ are the output weights, and $\mathrm{diag}(\bar{\mathbf{D}}) \in \mathbb{R}^M$ are the residual weights. We follow the S5 model (Smith et al., 2023) for the initialization and parameterization of the model. See Appendix A.2 for more details.

Neuromorphic Computing with Intel Loihi 2 Loihi 2 is the second-generation of Intel's neuromorphic research processor (Orchard et al., 2021) and implements a spiking neural network as illustrated in Figure 1. The network is processed by massively parallel compute units, with 120 such *neuro-cores* per chip. The neuro-cores compute and communicate asynchronously, but a global algorithmic time step is maintained through a barrier synchronization process. The neuro-cores are co-located with memory and can thus efficiently update local states, simulating up to $8\,192$ stateful neurons per core. Each neuron can be programmed by the user to realize a variety of temporal dynamics through assembly code. Input from and output to external hosts and sensors is provided with up to $160\,\mathrm{M}$ 32 bit integer $\mathrm{messages/s}$ (Shrestha et al., 2024b). Loihi 2 can scale to real-world workloads of various sizes with up to $1\,\mathrm{B}$ neurons and $128\,\mathrm{B}$ synapses, using fully-digital stacked systems shown in Figure 1. Importantly, Loihi 2 offers efficient low-precision arithmetic and its event-driven architecture enables acceleration of sparse weight matrices and sparse activation vectors.

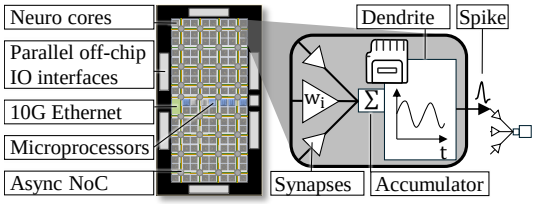

Figure 1: Loihi 2 implements a mesh of neuro-cores through an asynchronous network-on-chip. On a neuro-core, each neuron receives $24\,\mathrm{bit}$ spike messages from other neurons via weighted synapses, with $8\,\mathrm{bit}$ precision, which are summed up in dendritic accumulators. This input updates memory states that are local to the respective neuron. The neuron communicates with other neurons by sending spike messages.

Synaptic Pruning Following previous work (Mishra et al., 2021), we initialize the parameters from the pre-trained dense models. We adopt iterative magnitude pruning (IMP) which increases sparsity progressively during training and achieves better task performance than one-shot approaches, especially at high sparsity levels (Zhu & Gupta, 2018; Lee et al., 2023). More details on our implementation are given in Appendix A.2.

Activity Sparsification Sparsifying layer activations provide another means for reducing the compute and on-chip memory requirements during inference. On sparse and event-driven accelerators, such as Loihi 2, sparse pre-activations directly translate into Multiply-and-Accumulate (MACs) savings. In contrast, GPU architectures struggle to leverage dynamic sparse activation patterns and have demonstrated gains with more structured activation patterns, and only in memory-bound regimes as in auto-regressive generation with large models. We use *ReLU-fication* (Mirzadeh et al., 2024) for activation sparsity since this is a fully element-wise operation that doesn't require synchronization across channels, which simplifies the implementation in compute-memory integrated platforms such as Loihi 2. Other promising techniques include top-k (Key et al., 2024), sigma-delta coding (Shrestha et al., 2024a; O'Connor & Welling, 2016), and sparse mixture-of-experts (Fedus et al., 2022; He, 2024). We detail our ReLU-fication process in Appendix A.2.

Quantization and Fixed-Point Computation Reducing the numerical precision of weights and activations through quantization is an essential way to compress neural networks, directly leading to reduced memory footprint and faster inference (Gholami et al., 2021). Without constraints during training, post-training quantization (PTQ) has been shown to under-perform on both nonlinear and linear RNNs (Wu et al., 2016; Abreu et al., 2024). In contrast, quantization-aware training (QAT)

simulates quantization in the forward pass and has shown promising results on LRNNs (Meyer et al., 2024; Abreu et al., 2024). To enable hardware acceleration, we use static quantization (Gholami et al., 2021) with fixed-point arithmetic (Wu et al., 2020). For more details, see Appendix A.2.

3 Experimental Results

We evaluated our approach on the Intel Neuromorphic Deep Noise Suppression Challenge (Timcheck et al., 2023), which aims to enhance the clarity of human speech recorded on a single microphone in a noisy environment. Clean human speech and noise samples are mixed to produce noisy human speech with a ground truth clean human speech goal. The denoising quality is evaluated with the scale-invariant signal-to-noise ratio **SI-SNR**. See Appendix A.2 for more details.

Pareto Front of Performance and Efficiency We studied the performance-efficiency Pareto front of dense and sparse models across inference compute budgets. Starting from the S5 architecture (Smith et al., 2023), we trained a family of dense models by linearly scaling the model width, both the model dimension and size of the S5 hidden state, while keeping the depth fixed to three layers. Similarly, we trained a family of sparse models, pruned and ReLU-fied according to our methodology discussed above, with 90% of weights pruned by the end of training (further details on the model dimensions are provided in Appendix A.2). The results, reported in Figure 2, compare de-noising performance (SI-SNR) and computational efficiency as measured by effective MACs (see Appendix A.1).

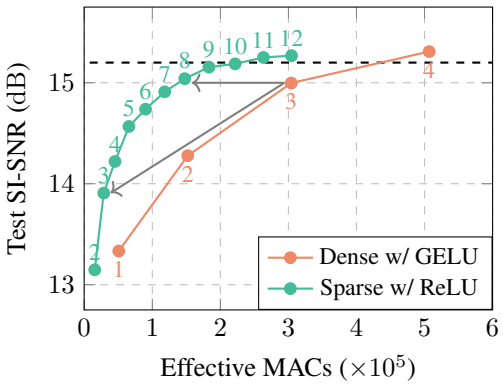

Figure 2: Pareto fronts for S5 network audio denoising quality (SI-SNR) as a function of effective compute on the Intel N-DNS test set. S5 networks with weight and activation sparsity (green) exhibit a large domain of Pareto optimality versus dense S5 networks (orange). Number annotations enumerate increasing model width, from 500 k to 4 M parameters. Dashed horizontal like marks SI-SNR of Spiking-FullSubNet XL, the previous state-of-the-art model.

The results show that sparsification significantly degrades performance when applied to underparametrized dense models (e.g., sparsifying dense-3 reduces SI-SNR by 7.3%). However, task performance is recovered with increased model width and the accuracy of dense models is matched by wider sparse ones, with fewer MACs and lower memory requirements. This gives empirical support to theoretical work on the capacity of sparse-and-wide neural networks (Golubeva et al., 2021). For example, sparse-8 model requires **2× lower compute** and **36% lower memory** than the dense-3 model, **while achieving the same level of accuracy**. Overall, *sparse models constitute the Pareto front of task performance and computational efficiency across compute budgets*. In terms of absolute task performance, we find that the S5 architecture provides state-of-the-art results on audio denoising. When compared to Spiking-FullSubNet-XL (Hao et al., 2024), the Track 1 winner of the Intel N-DNS Challenge with 15.2 dB SI-SNR, our sparse-11 S5 model requires **3.2× lower compute** and **5.37× lower memory iso-accuracy**.

Interaction of Weight and Activation Sparsity An interesting question is what is the interaction between the two types of sparsity, in weights and activations. Figure 3a) reports the pre-activation sparsity for different layers across the model depth for two ReLU-fied models of the same size (model variant 6), with and without synaptic sparsity. We observe that the synaptic-sparse model exhibits lower activation sparsity across the board, a finding that is consistent across model sizes. In addition, activation sparsity significantly decreases with model depth, both for dense and sparse models. These phenomena, previously observed in other models (Mukherji et al., 2024), suggest that, during training, the model compensates the reduced information flow caused by pruning with increased levels of activation. Additional research on more advanced activation functions would allow for the optimal

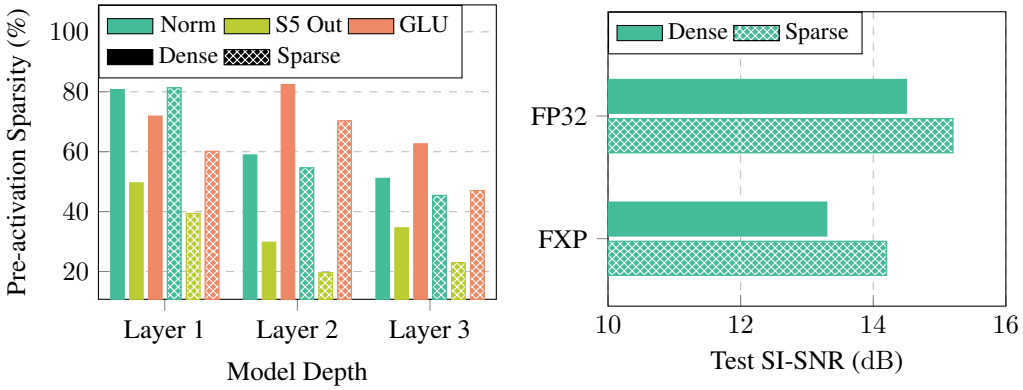

Figure 3: **a)** Activation sparsity of ReLU blocks across model depth for a dense-weight model and a sparse-weight model. The sparse-weight model exhibits significantly lower activation sparsity across layers. **b)** Impact of quantization intervention on Test SI-SNR for weight-sparse (sparse-8) and a weight-dense (dense-3) model, with similar initial accuracy. The wider but sparse model is more resilient to quantization, showing a -6.6% drop in SI=SNR, compared to -8.3% of the dense one.

allocation of MACs, especially those that provide explicit control over sparsity without cross-channel synchronization (e.g., approximate top-k (Key et al., 2024)).

Interaction of Quantization and Weight Sparsity Figure 3b) shows the effect of fixed-point quantization on Test SI-SNR for a dense (dense-3) and a sparse one (sparse-8), with similar initial task performance. We found that the wide-sparse model is more resilient to quantization than the narrow-dense model, showing a -6.6% drop in SI-SNR, compared to -8.3% of the dense one. This is in line with previous findings showing that wider models are more robust to quantization (Harma et al., 2025). This is promising as wide-sparse models outperform narrow-dense models, and can be further compressed with quantization more easily.

Hardware Implementation To measure the empirical efficiency benefits afforded by the sparse S5 model on neuromorphic hardware, we profile inference on Loihi 2 using the fixed-point S5 model, in particular, configuration sparse-8 from Figure 2. We also compared to conventional hardware, profiling the dense-3 model on Jetson Orin Nano[1]. On the Jetson, we optimize our implementation to fit the real-time requirement of audio denoising. Our sparse model on Loihi provides a $42\times$ latency improvement, $149\times$ energy reduction and $4\times$ throughput improvement.

Table 1: Results of a dense model on the Jetson Orin Nano compared against a sparse model on the Loihi 2. See text for explanation.

	Latency	**Energy**	**Throughput**
Loihi 2	76 µs	13 µJ	13 178
Orin Nano	3 224 µs	1 936 µJ	3 103

4 Discussion

Our work demonstrates that sparse event-driven accelerators, such as neuromorphic processors, can provide state-of-the-art accuracy on high-frequency signal processing tasks, with orders of magnitude gains in latency and energy efficiency. This possibility opens up several research directions to further materialize these gains in real-world applications. In particular, future work should investigate how the efficiency-performance Pareto front scales up to larger models and more complex tasks, such as language and multimodal modeling. In this setting, the scalability of multi-chip neuromorphic processors (Kudithipudi et al., 2025) and high-frequency execution could power the growing need for large-scale inference compute (Snell et al., 2024).

[1]Our W8A16 fixed-point model in JAX does not provide a speedup over the FP32 model on the Jetson Orin Nano, therefore we profile the FP32 model.

REFERENCES

Steven Abreu, Jens E Pedersen, Kade M Heckel, and Alessandro Pierro. Q-S5: Towards quantized state space models. *International Conference on Machine Learning Workshops*, 2024.

Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432, 2013. URL http://arxiv.org/abs/1308.3432.

Xiangyu Chang, Yingcong Li, Samet Oymak, and Christos Thrampoulidis. Provable Benefits of Overparameterization in Model Compression: From Double Descent to Pruning Neural Networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(8):6974–6983, May 2021. ISSN 2374-3468. doi: 10.1609/aaai.v35i8.16859. URL https://ojs.aaai.org/index.php/AAAI/article/view/16859. Number: 8.

Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the Lottery: Making All Tickets Winners. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 2943–2952. PMLR, November 2020. URL https://proceedings.mlr.press/v119/evci20a.html. ISSN: 2640-3498.

William Fedus, Barret Zoph, and Noam Shazeer. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity, June 2022. URL http://arxiv.org/abs/2101.03961. arXiv:2101.03961 [cs].

Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. A Survey of Quantization Methods for Efficient Neural Network Inference, June 2021. URL http://arxiv.org/abs/2103.13630. arXiv:2103.13630 [cs].

Anna Golubeva, Guy Gur-Ari, and Behnam Neyshabur. Are wider nets better given the same number of parameters? October 2021. URL https://openreview.net/forum?id=_zx8Oka09eF.

Albert Gu, Isys Johnson, Karan Goel, Khaled Saab, Tri Dao, Atri Rudra, and Christopher Ré. Combining recurrent, convolutional, and continuous-time models with linear state space layers. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 572–585, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/05546b0e38ab9175cd905eebcc6ebb76-Abstract.html.

Albert Gu, Karan Goel, Ankit Gupta, and Christopher Ré. On the parameterization and initialization of diagonal state space models. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/e9a32fade47b906de908431991440f7c-Abstract-Conference.html.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

Xiang Hao, Chenxiang Ma, Qu Yang, Kay Chen Tan, and Jibin Wu. When audio denoising meets spiking neural network. In *2024 IEEE Conference on Artificial Intelligence (CAI)*, pp. 1524–1527, 2024. doi: 10.1109/CAI59869.2024.00275.

Simla Burcu Harma, Ayan Chakraborty, Elizaveta Kostenok, Danila Mishin, Dongho Ha, Babak Falsafi, Martin Jaggi, Ming Liu, Yunho Oh, Suvinay Subramanian, and Amir Yazdanbakhsh. Effective Interplay between Sparsity and Quantization: From Theory to Practice, January 2025. URL http://arxiv.org/abs/2405.20935. arXiv:2405.20935 [cs].

Xu Owen He. Mixture of A Million Experts, July 2024. URL http://arxiv.org/abs/2407.04153. arXiv:2407.04153 [cs].

Oscar Key, Luka Ribar, Alberto Cattaneo, Luke Hudlass-Galley, and Douglas Orr. Approximate top-*k* for increased parallelism. *CoRR*, abs/2412.04358, 2024. doi: 10.48550/ARXIV.2412.04358. URL https://doi.org/10.48550/arXiv.2412.04358.

Dhireesha Kudithipudi, Catherine Schuman, Craig M. Vineyard, Tej Pandit, Cory Merkel, Rajkumar Kubendran, James B. Aimone, Garrick Orchard, Christian Mayr, Ryad Benosman, Joe Hays, Cliff Young, Chiara Bartolozzi, Amitava Majumdar, Suma George Cardwell, Melika Payvand, Sonia Buckley, Shruti Kulkarni, Hector A. Gonzalez, Gert Cauwenberghs, Chetan Singh Thakur, Anand Subramoney, and Steve Furber. Neuromorphic computing at scale. *Nature*, 637(8047): 801–812, January 2025. ISSN 1476-4687. doi: 10.1038/s41586-024-08253-8. URL https://www.nature.com/articles/s41586-024-08253-8.

Joo Hyung Lee, Wonpyo Park, Nicole Mitchell, Jonathan Pilault, Johan S. Obando-Ceron, Han-Byul Kim, Namhoon Lee, Elias Frantar, Yun Long, Amir Yazdanbakhsh, Shivani Agrawal, Suvinay Subramanian, Xin Wang, Sheng-Chun Kao, Xingyao Zhang, Trevor Gale, Aart Bik, Woohyun Han, Milen Ferev, Zhonglin Han, Hong-Seok Kim, Yann N. Dauphin, Karolina Dziugaite, Pablo Samuel Castro, and Utku Evci. Jaxpruner: A concise library for sparsity research. *CoRR*, abs/2304.14082, 2023. doi: 10.48550/ARXIV.2304.14082. URL https://doi.org/10.48550/arXiv.2304.14082.

Shiwei Liu and Zhangyang Wang. Ten lessons we have learned in the new "sparseland": A short handbook for sparse neural network researchers. *CoRR*, abs/2302.02596, 2023. doi: 10.48550/ARXIV.2302.02596. URL https://doi.org/10.48550/arXiv.2302.02596.

Svea Marie Meyer, Philipp Weidel, Philipp Plank, Leobardo Campos-Macias, Sumit Bam Shrestha, Philipp Stratmann, and Mathis Richter. A diagonal structured state space model on loihi 2 for efficient streaming sequence processing. *arXiv preprint arXiv:2409.15022*, 2024.

Seyed Iman Mirzadeh, Keivan Alizadeh-Vahid, Sachin Mehta, Carlo C del Mundo, Oncel Tuzel, Golnoosh Samei, Mohammad Rastegari, and Mehrdad Farajtabar. ReLU strikes back: Exploiting activation sparsity in large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=osoWxY8q2E.

Asit Mishra, Jorge Albericio Latorre, Jeff Pool, Darko Stosic, Dusan Stosic, Ganesh Venkatesh, Chong Yu, and Paulius Micikevicius. Accelerating sparse deep neural networks. *arXiv preprint arXiv:2104.08378*, 2021.

Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H. Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature Communications*, 9(1):2383, June 2018. ISSN 2041-1723. doi: 10.1038/s41467-018-04316-3. URL https://doi.org/10.1038/s41467-018-04316-3.

Rishav Mukherji, Mark Schöne, Khaleelulla Khan Nazeer, Christian Mayr, David Kappel, and Anand Subramoney. Weight sparsity complements activity sparsity in neuromorphic language models. *arXiv preprint arXiv:2405.00433*, 2024.

Peter O'Connor and Max Welling. Sigma delta quantized networks. *arXiv preprint arXiv:1611.02024*, 2016.

Garrick Orchard, Edward Paxon Frady, Daniel Ben Dayan Rubin, Sophia Sanborn, Sumit Bam Shrestha, Friedrich T. Sommer, and Mike Davies. Efficient neuromorphic signal processing with loihi 2. In *IEEE Workshop on Signal Processing Systems, SiPS 2021, Coimbra, Portugal, October 19-21, 2021*, pp. 254–259. IEEE, 2021. doi: 10.1109/SIPS52927.2021.00053. URL https://doi.org/10.1109/SiPS52927.2021.00053.

Antonio Orvieto, Soham De, Caglar Gulcehre, Razvan Pascanu, and Samuel L. Smith. Universality of linear recurrences followed by non-linear projections: Finite-width guarantees and benefits of complex eigenvalues. In *ICML*, 2024. URL https://openreview.net/forum?id=47ahBl70xb.

Sumit Bam Shrestha, Jonathan Timcheck, Paxon Frady, Leobardo Campos-Macias, and Mike Davies. Efficient video and audio processing with loihi 2. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 13481–13485. IEEE, 2024a.

Sumit Bam Shrestha, Jonathan Timcheck, Paxon Frady, Leobardo Campos-Macias, and Mike Davies. Efficient Video and Audio Processing with Loihi 2. In *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 13481–13485, April 2024b. doi: 10.1109/ICASSP48485.2024.10448003. URL https://ieeexplore.ieee.org/abstract/document/10448003.

Jimmy T. H. Smith, Andrew Warrington, and Scott W. Linderman. Simplified state space layers for sequence modeling. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/pdf?id=Ai8Hw3AXqks.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *CoRR*, abs/2408.03314, 2024. doi: 10.48550/ARXIV.2408.03314. URL https://doi.org/10.48550/arXiv.2408.03314.

Jonathan Timcheck, Sumit Bam Shrestha, Daniel Ben Dayan Rubin, Adam Kupryjanow, Garrick Orchard, Lukasz Pindor, Timothy Shea, and Mike Davies. The intel neuromorphic dns challenge. *Neuromorphic Computing and Engineering*, 3(3):034005, aug 2023. doi: 10.1088/2634-4386/ace737. URL https://dx.doi.org/10.1088/2634-4386/ace737.

Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *CoRR*, abs/1804.03209, 2018. URL http://arxiv.org/abs/1804.03209.

Hao Wu, Patrick Judd, Xiaojie Zhang, Mikhail Isaev, and Paulius Micikevicius. Integer Quantization for Deep Learning Inference: Principles and Empirical Evaluation, April 2020. URL http://arxiv.org/abs/2004.09602. arXiv:2004.09602 [cs, stat].

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation, October 2016. URL http://arxiv.org/abs/1609.08144. arXiv:1609.08144 [cs].

Michael Zhu and Suyog Gupta. To prune, or not to prune: Exploring the efficacy of pruning for model compression. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=Sy1iIDkPM.

A APPENDIX

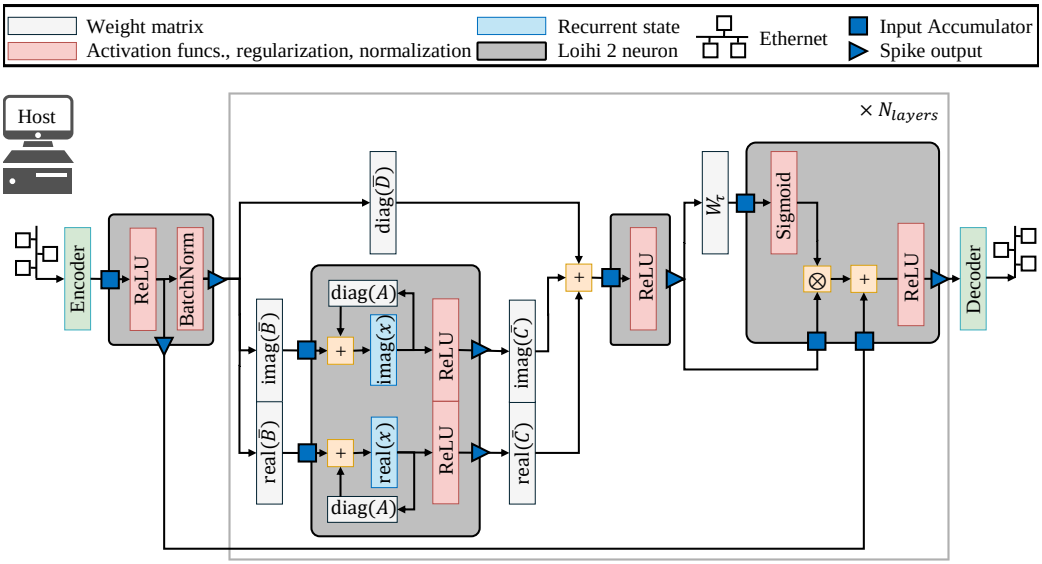

Figure 4: Diagram of S5 as implemented on Loihi 2. To leverage the neuromorphic hardware architecture, several adjustments are made in comparison to the original S5 model: First, complex numbers are split into real and complex components for processing. Second, ReLUs are introduced to increase activation sparsity. Third, multiple element-wise operations are fused into single neuromorphic neurons. Symbols are shown as defined in Section 2.

A.1 EFFECTIVE MACS COMPUTATION FOR S5 ARCHITECTURE

In this section, we detail the computation of effective multiply-accumulate operations (MACs) for different components of the S5 architecture. The total MAC count provides an estimate of the computational cost associated with each stage of the model. Below, we outline the individual contributions from key components of the architecture. The effective MACs for all model sizes–sparse and dense–in Figure Figure 2 are calculated based on the formulas below, summed over the entire network structure.

Notation:

- N_{input}: Input dimension
- N_{model}: Model dimension for activations outside of the linear RNN.
- N_{ssm}: Dimension of the linear RNN's hidden state.
- N_{output}: Output dimension (equal to the number of classes for classification)
- d_x^{wgt}: Density of weights for x
- d_x^{act}: Density of activations for x

where the density d is calculated from the sparsity s as $d = 1 - s$.

Breakdown of MAC Calculation per Component:

- **Encoder:** The MACs for the encoder depend on the input dimension, model size, and scale linearly with activation and weight densities:

$$N_{\text{input}} N_{\text{model}} d_{\text{encoder}}^{\text{wgt}} d_{\text{input}}^{\text{act}} \tag{3}$$

- **Batch Normalization (BatchNorm):** A lightweight operation, requiring only element-wise scaling, leading to:

$$N_{\text{model}} \tag{4}$$

- **S5 Hidden Layer:** The hidden state update for the S5 model involves both matrix multiplications and element-wise operations:

$$2N_{\text{model}}N_{\text{ssm}}d_B^{\text{wgt}}d_{\text{pre\_ssm}}^{\text{act}} + 4N_{\text{ssm}} \tag{5}$$

- **SSM Output Layer:** Computes the output transformation of the linear RNN:

$$2N_{\text{ssm}}N_{\text{model}}d_C^{\text{wgt}}d_{\text{hidden}}^{\text{act}} + N_{\text{model}}d_{\text{pre\_ssm}}^{\text{act}} \tag{6}$$

- **Gated Linear Unit (GLU):** The computation for the GLU involves matrix multiplications for the dense weight matrix, followed by an element-wise multiplication:

$$N_{\text{model}}^2 d_{\text{GLU}}^{\text{wgt}}d_{\text{pre\_GLU}}^{\text{act}} + N_{\text{model}} \tag{7}$$

- **Classification Head:** The final linear projection for classification:

$$N_{\text{model}}N_{\text{output}}d_{\text{head}}^{\text{wgt}}d_{\text{pre\_hread}}^{\text{act}} \tag{8}$$

- **Regression Head:** The regression head follows the same computation as the classification head:

$$N_{\text{model}}N_{\text{output}}d_{\text{head}}^{\text{wgt}}d_{\text{pre\_hread}}^{\text{act}} \tag{9}$$

Numerical operations such as the inverse square-root, sigmoid function, and others, are ignored from our MAC calculations, as is commonly done when calculating the MACs or floating point operations (FLOPs) of machine learning models (Evci et al., 2020).

A.2 Experimental Details

Model architecture Our linear RNN is based on the S5 architecture (Smith et al., 2023), as described in section 2. We use the following dimensions for our base model with width scaling $k = 1$ (*i.e.* configuration 4 in Figure 2). We use three layers, the recurrent state vector is $\mathbf{x}_t \in \mathbb{R}^{256}$, we use a model dimension of 192. Both input and output have dimension 257. The width scaling factors k_i scale the model and recurrent state dimension linearly. In Figure 2, we report results for a k-family of sparse and densely trained networks where $k_{\text{sparse}} \in [0.5, 3.0],\ k_{\text{dense}} \in [0.25, 1.0]$.

Training recipe We trained all models for 50 epochs with the Adam optimizer. The parameters of the SSM block were updated with initial learning rate 0.002, while the rest of the architecture used initial learning rate 0.008 and weight decay 0.04. All learning rates used cosine annealing and no warmup epochs. The dropout was set to 0.1.

Iterative pruning We adopt iterative magnitude pruning (IMP) for weight sparsity. We train for E epochs with T update steps in total. Sparsity starts at $S_i = 0$ at $t_i = 0$ and is increased following a degree-3 polynomial schedule (Zhu & Gupta, 2018) and updated three times per epoch as:

$$S_t = S_f - (S_f - S_i) \cdot \left(1 - \frac{t - t_i}{t_f - t_i}\right)^3$$

with $t_f = 0.75T$. Given the total sparsity S_t and weights $W_t^\ell \in \mathbb{R}^{N^\ell \times M^\ell}$ at time t and position ℓ in the network, we scale the sparsity s_t^ℓ for each weight according to the Erdös-Renyi-Kernel (ERK) strategy (Evci et al., 2020; Mocanu et al., 2018) to compute the mask M_t^ℓ:

$$\begin{aligned} s_t^\ell &= s_t \cdot \frac{N^\ell + M^\ell}{N^\ell \cdot M^\ell} \\ M_t^\ell &= \mathbb{1}\left(|W_t^\ell| \geq \tau_t^\ell\right) \\ \tau_t^\ell &= \min\left[\text{TopK}\left(|W_t^\ell|, s_t^\ell N^\ell M^\ell\right)\right] \end{aligned}$$

where τ_t^ℓ is the calculated threshold for W_t^ℓ to reach sparsity s_t^ℓ and $\text{TopK}(W, k)$ gives the top-k values from W. In the forward pass, weights are masked as $\bar{W} = M \odot W$, while the backward pass applies straight-through estimation (Bengio et al., 2013) enabling gradient updates also for masked weights.

ReLU-fication . Following previous work on transformer models (Mirzadeh et al., 2024), we start from the original dense model with GELU non-linearity, as shown in **??**, and apply two modifications. First, we replace the GELU activation with a ReLU, sparsifying pre-activations of the linear layer in the GLU block. Second, we insert additional ReLU activations after the residual add in the GLU block and to the real component of the S5 hidden layer, further increasing the pre-activation sparsity of linear operators. Both model surgeries are applied to the pre-trained model at the beginning of the iterative pruning procedure, enabling accuracy recovery from both weight and activation pruning without extra training budget.

Quantization We denote the tensor to be quantized with $\mathbf{x}$ and the number of bits to use with n, such that the quantized tensor $\bar{\mathbf{x}}_n$ is defined as:

$$\bar{\mathbf{x}}_n = \left\lfloor \frac{\mathbf{x}}{\Delta_x} + z_x \right\rceil = \lfloor s_x \mathbf{x} + z_x \rceil \tag{10}$$

where $\lfloor \cdot \rceil$ indicates rounding to the nearest integer, s_x is the scale for the given tensor, z_x is the zero point, and Δ_x is the corresponding step size. $s_x = (2^{n-1} - 1)(\max |\mathbf{x}|)^{-1}$ and $z_x = \mathbf{0}$, *i.e.*, we use symmetric quantization based on the absolute maximum. Following prior work on quantizing linear RNNs (Abreu et al., 2024), we choose $8\,\mathrm{bit}$ for all weights, except the diagonal recurrent $\mathrm{diag}(\bar{A})$ weights which is stored with $16\,\mathrm{bit}$. All activations are quantized to $16\,\mathrm{bit}$. We denote this quantization recipe with W8A16. For the linear RNNs that are deployed to the Loihi 2 chip, we combine QAT with sparse training.

N-DNS Denoising Task The denoising quality is evaluated with the scale-invariant signal-to-noise ratio

$$\text{SI-SNR} = 10 \log_{10} \frac{\left\| s_{\text{target}} \right\|^2}{\left\| e_{\text{noise}} \right\|^2}, \tag{11}$$

which provides a volume-agnostic measure of audio cleanliness relative to the ground truth signal. To train our models, we used the default Intel N-DNS Challenge training and validation sets, each consisting of $60\,000$ noisy audio samples of $30\,\mathrm{s}$ each, and a test set with $12\,000$ samples. We encoded and decoded each audio sample using the Short-Time Fourier Transform (STFT) and Inverse Short-Time Fourier Transformer (iSTFT) following the N-DNS baseline solution, NsSDNet (Shrestha et al., 2024a), with a $32\,\mathrm{ms}$ window length and a $8\,\mathrm{ms}$ hop length. This resulted in a nominal real-time audio processing latency of $32\,\mathrm{ms}$, which allows ample time ($8\,\mathrm{ms}$) for denosing network inference, as $40\,\mathrm{ms}$ is the standard for an acceptable latency as recognized in the Microsoft N-DNS Challenge.

A.3 Additional Results

A.3.1 Loihi execution mode

Loihi 2's asynchronous architecture allows to trade off between throughput and latency, as illustrated in Figure 5a. For optimal throughput, new input is provided every time step and forwarded through the neuronal layers in a pipelined mode. For optimal latency, new input is injected only once the previous input has been processed by, or fallen through, the network as fast as possible. The pipelined and fall-through mode can be balanced by changing the rate of new input, to match the throughput of a given input stream while minimizing its processing latency.

As audio denoising is typically deployed in realtime in an online fashion where one STFT input frame in processed at a time, fall-through mode is appropriate, as one desires a corresponding output STFT frame immediately.

We see that Loihi 2 processes a single STFT frame $35\times$ faster and with $1200\times$ less energy than the Jetson Orin Nano (Token-by-token; Loihi 2 Fall-Through and Jetson Orin Nano Recurrent 1-step (b=1) in Table 2).

Table 2: Power and performance results*. The Loihi 2 is running a sparse and quantized S5 model, while the Jetson Orin Nano is running a smaller dense S5 model that reaches similar test performance. All measurements are averaged over 8 random samples from the test set, each containing 3 750 time steps. Gray highlights denote violation of real-time constraints for the audio denoising task. Best real-time results are underlined.

	Mode	Latency (↓)	Energy (↓)	Throughput (↑)
Token-by-token				
Intel Loihi 2†	Fall-Through	76 µs	13 µJ/tok	13 178 tok/s
Jetson Orin Nano‡	Recurrent 1-step ($b = 1$)	2 688 µs	15 724 µJ/tok	372 tok/s
Jetson Orin Nano‡	Recurrent 10-step ($b = 1$)	3 224 µs	1 936 µJ/tok	3 103 tok/s
Jetson Orin Nano‡	Recurrent 100-step ($b = 1$)	10 653 µs	626 µJ/tok	9 516 tok/s
Jetson Orin Nano‡	Recurrent scan ($b = 1$)	236 717 µs	404 µJ/tok	15 845 tok/s
Sample-by-sample				
Intel Loihi 2†	Pipeline	60.58 ms	185.80 mJ/sam	16.58 sam/s
Jetson Orin Nano‡	Scan ($b = 1$)	233.48 ms	1 512.60 mJ/sam	4.28 sam/s
Jetson Orin Nano‡	Scan ($b = b_{\max}$)	226.53 ms	5.89 mJ/sam	1 130.09 sam/s

† Loihi 2 workloads were characterized on an Oheo Gulch system with N3C1-revision Loihi 2 chips running NxCore 2.5.8 and NxKernel 0.2.0 with on-chip IO unthrottled sequencing of inputs. Researchers interested to run S5 on Loihi 2 can gain access to the software and systems by joining *Intel's Neuromorphic Research Community*. ‡ Jetson workloads were characterized on an NVIDIA Jetson Orin Nano 8GB running Jetpack 6.2, CUDA 12.4, JAX 0.4.32, using the MAXN SUPER power mode; energy values are computed based on the TOT power as reported by jtop 4.3.0. The batch size $b_{\max} = 256$ was chosen to be the largest that fits into memory. *Performance results are based on testing as of January 2025 and may not reflect all publicly available security updates; results may vary.

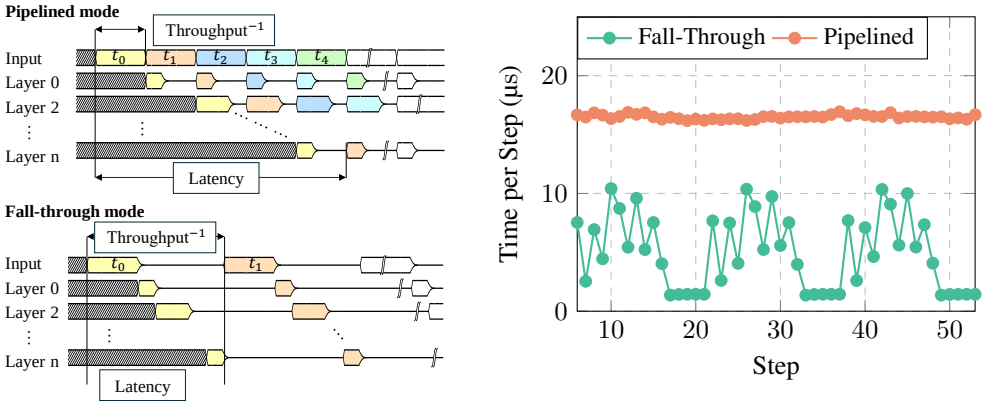

Figure 5: (a) Loihi 2 offers two processing modes that optimize either throughput or latency. In the *pipelined mode*, a new data point is inserted in each time step, to use all processing cores and maximize the throughput–at the expense of latency because equal time bins $t_0 = t_1 = \ldots$ are enforced. In the *fall-through mode*, a new data points is only provided once the last data point has been fully processed with minimum latency. Only a single neuronal layer is active at any step as data travels through the network. The time per step is thus minimized as traffic is reduced and potentially more complex neuronal layers are not updated. (b) Comparison of execution mode and time per step.