Activity Sparsity Complements Weight Sparsity for Efficient RNN Inference

Rishav Mukherji[1*] **Mark Schöne**[2] **Khaleelulla Khan Nazeer**[2]
Christian Mayr[2] **Anand Subramoney**[3]
[1] Birla Institute of Technology and Science, Pilani – Goa Campus, India
[2] TUD Dresden University of Technology, Germany [3] Royal Holloway, University of London, UK
rishavm16@gmail.com mark.schoene@tu-dresden.de

Abstract

Artificial neural networks open up unprecedented machine learning capabilities at the cost of ever growing computational requirements. Sparsifying the parameters, often achieved through weight pruning, has been identified as a powerful technique to compress the number of model parameters and reduce the computational operations of neural networks. Less well studied are sparse activations for computational efficiency, while omnipresent in both biological neural networks and deep learning systems. Moreover, the interaction between sparse activations and weight pruning is not fully understood. In this work, we demonstrate that activity sparsity can compose multiplicatively with parameter sparsity in a recurrent neural network model based on the GRU that is designed to be activity sparse. We achieve up to $20\times$ reduction of computation while maintaining perplexities below 60 on the Penn Treebank language modeling task. This magnitude of reduction has not been achieved previously with solely sparsely connected LSTMs, and the language modeling performance of our model has not been achieved previously with any sparsely activated recurrent neural networks or spiking neural networks. Neuromorphic computing devices are especially good at taking advantage of the dynamic activity sparsity, and our results provide strong evidence that making deep learning models activity sparse and porting them to neuromorphic devices can be a viable strategy that does not compromise on task performance. Our results also drive further convergence of methods from deep learning and neuromorphic computing for efficient machine learning.

1 Introduction

As the available compute per energy unit grows, artificial neural networks (ANNs) become increasingly popular for applications ranging from cloud services to mobile and edge systems. While task performance is crucial for all applications, including on low-power environments such as mobiles, the energy consumption of the system is critical to allow deployment in such environments. Many tasks have additional latency requirements for safety reasons or to enhance the user experience. Hence, enhancing ANNs inference efficiency is vital for deep learning application deployment.

The power and latency of deployed neural networks depends on the number of memory accesses and the number of arithmetic and logic operations conducted by the system. While on-chip SRAM access energy costs are comparable to arithmetic operations, DRAM access is orders of magnitude more energy and latency intensive [20]. Hence, the key performance indicators for accelerating neural network inference, energy and latency, are dominated by reading weights from DRAM. This issue is intensified for the case of batch size 1 inference, a common setting for mobile applications, since the cost of fetching weights cannot be spread across multiple samples.

[*] Work carried out at TUD Dresden University of Technology

First Workshop on Machine Learning with New Compute Paradigms at NeurIPS 2023 (MLNPCP 2023).

Compressing neural networks to reduce the energy and latency cost is an active area of research. The most relevant techniques include (1.) sparse and low-rank weight matrices [19], and (2.) fixed-point integer quantization to commonly 8-bit and below [48, 31]. Sparse and low-rank matrices reduce the number of weights that have to be fetched from memory, while quantization reduces the number of bits transferred per weight. Less popular is the topic of sparse neuron activations. Sparse activations theoretically limit the weights that have to be fetched from memory to the columns/rows associated with the non-zero activations, a large potential efficiency gain. Although, sparse activations have been observed in deep feed-forward networks [23, 21, 25], many hardware accelerators, with a few exceptions [38, 52, 53], do not leverage this type of sparsity due to their dynamic nature. The ability to handle dynamic activation sparsity is a core feature of neuromorphic hardware such as [26, 28]. Most neuromorphic systems operate in an event-driven manner, and co-locate memory and computation to reduce the energy and latencies [6].

In this work, we focus on activity sparse recurrent neural networks (RNNs) based on the recently published event-based GRU (EGRU) model [41]. With the recent resurgence in RNNs with architectures that are able to get close to or beat transformers in language modeling task performance [37, 8], we expect our analysis to be relevant in a broader and more practical context as well.

Our contributions to this workshop are as follows:

- Using an efficient recurrent architecture (EGRU) designed for novel neural network accelerators, we show how activity sparsity can be tuned using weight decay
- We show that the number of connections of small-scale language models can be compressed with minimal loss in perplexity, which previously has only been shown on subpar baseline models for the Penn Treebank and WikiText-2 datasets [54].
- We demonstrate that the reduction factors from activity sparsity and weight sparsity compose multiplicatively to yield a significant reduction of required memory accesses and arithmetic operations of up to $20\times$ without compromising on perplexity.

2 Related Work

Pruning RNNs. An extensive review of weight pruning techniques can be found in [19]. Pruning has been applied to a range of recurrent architectures including Elman RNNs, LSTMs, and GRUs [34, 17, 54, 1]. On speech recognition benchmarks compression rates of up to $90\,\%$ can be achieved with the LSTM model without loss in performance [17, 54, 1]. Dai et al. [7] reported an increase of sparsity by expanding the linear transformations of LSTM gates to be multi layer neural networks. The best pruned LSTM model for language modeling on the Penn Treebank dataset in the literature is reported by [54]. They achieve their best results, a perplexity of 77.5 (where lower is better), at a weight sparsity of $80\,\%$. We generally find that pruning has not been applied to more recent LSTM based language models such as the AWD-LSTM [33], which achieves a perplexity of 57.3.

Activity sparsity for RNNs. The ReLU activation function sparsifies RNN activations at the cost of unstable training dynamics. Talathi and Vartak [43], Li et al. [24] addressed this by training diagonal RNNs with ReLU activations. They traded non-linear recurrent operations for feed-forward operations, and used more parameters than required with shallow but fully connected RNNs. Delta Networks [35] operate on differences between hidden states at consecutive time-steps. This operator is shown to be equivalent to standard RNNs, while the deltas can be thresholded to yield a sparse approximation. Spiking neural networks (SNNs) based on the biologically plausible leaky-integrate-and-fire neuron hold the promise of efficient inference on neuromorphic hardware but struggle to achieve state-of-the-art performance on machine learning benchmarks [13, 44, 42, 49]. Aiming to bridge the efficiency promise of SNNs with the task performance of ANNs, [47] and Subramoney et al. [41] combined deep neural network architectures with discontinuous step functions and state resets to mimic the behaviour of SNNs, while achieving better task performance. Zhu et al. [55] followed a similar approach and applied SNNs in the context of larger-scale language models.

Combining activity sparsity and weight sparsity. Hunter et al. [21] introduced a structured sparse algorithm for top-k winner-takes-it-all activation sparsity for feed-forward networks. Gao et al. [12] show significant efficiency gains of joint activity and weight sparsity with an FPGA-based LSTM accelerator on a speech recognition benchmark. While weight pruning is a popular research topic in the context of SNNs [3, 40, 36, 4, 22, 51], most works focus on image classification.

Language modeling with RNNs. Small scale language modeling datasets such as Penn Treebank [27] or WikiText-2 [32] drove progress of LSTM based language models before Vaswani et al. [45] enabled large-scale language modeling. The best raw LSTM language models on the Penn Treebank dataset are reported in [33] and [29]. Recently, a set of RNNs based on linear recurrences demonstrated strong results on large-scale language modeling tasks [15, 11, 39]. In this work, we work in the small data regime, and comparison with the newer architectures will be done in future work.

3 Efficient Recurrent Neural Networks for Neuromorphic Accelerators

With our method, we aim for two goals: First, match the performance metrics targeted by the task. Second, respect the power and compute budget of the hardware system. We exploit lessons from deep learning as well as neuromorphic computing to significantly reduce the communication between processing elements and memory.

3.1 Event-based Gated Recurrent Unit

Long Short-Term Memory (LSTM) networks [18] and Gated Recurrent Units (GRU) [5] allow long range learning by gating input and hidden state variables for better gradient propagation. The Event-based GRU (EGRU) [41] combines gating mechanisms with spiking mechanisms inspired by biological neuron models. Therefore, it distinguishes between a local cell state $\mathbf{c} = (c_1,...,c_n)$ and a communicated cell state $\mathbf{y} = (y_1,...,y_n)$, where n is the hidden dimension. In each time-step, the communicated state $\mathbf{y}$ is a sparsification of the local state $\mathbf{c}$ via the pointwise heaviside thesholding function

$$y_i^{\langle t \rangle} = c_i^{\langle t \rangle} H\Big(c_i^{\langle t \rangle} - \vartheta_i\Big), \quad H(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}, \tag{1}$$

where $\boldsymbol{\vartheta} = (\vartheta_1,...,\vartheta_n)$ is a potentially trainable threshold parameter. This sparse state $\mathbf{y}$ is then passed to an update gate $\mathbf{u}$ and a reset gate $\mathbf{r}$, similar to the GRU model

$$\mathbf{u}^{\langle t \rangle} = \sigma\Big(\mathbf{W}_u \Big[\mathbf{x}^{\langle t \rangle}, \mathbf{y}^{\langle t-1 \rangle}\Big] + \mathbf{b}_u\Big), \quad \mathbf{r}^{\langle t \rangle} = \sigma\Big(\mathbf{W}_r \Big[\mathbf{x}^{\langle t \rangle}, \mathbf{y}^{\langle t-1 \rangle}\Big] + \mathbf{b}_r\Big). \tag{2}$$

The gates compute a proposed state $\mathbf{z}$ and the updated local state $\mathbf{c}$ as

$$\mathbf{z}^{\langle t \rangle} = g\Big(\mathbf{W}_z \Big[\mathbf{x}^{\langle t \rangle}, \mathbf{r}^{\langle t \rangle} \odot \mathbf{y}^{\langle t-1 \rangle}\Big] + \mathbf{b}_z\Big), \quad \mathbf{c}^{\langle t \rangle} = \mathbf{u}^{\langle t \rangle} \odot \mathbf{z}^{\langle t \rangle} + (1 - \mathbf{u}^{\langle t \rangle}) \odot \mathbf{c}^{\langle t-1 \rangle} - \mathbf{s}^{\langle t \rangle}. \tag{3}$$

Note that this is almost the update of the GRU, but with an additional reset term $\mathbf{s}^{\langle t \rangle} = \boldsymbol{\vartheta} H\big(\mathbf{c}^{\langle t \rangle} - \boldsymbol{\vartheta}\big)$. This term is motivated by the reset term commonly used in SNNs to improve activity sparsity (see [9] for a review). Another common strategy is to attach surrogate gradients to the non-differentiable Heaviside function $\frac{\mathrm{d}H}{\mathrm{d}c} = \lambda \max(1 - |c|/\epsilon)$ similar to [2] to allow differentiation of the event-based system.

3.2 Sparsely Connected Networks

Event-based systems such as EGRU improve efficiency by reducing the activity on each neuron-to-neuron channel. An orthogonal method to reduce the communication of a system is removing neuron-to-neuron channels entirely, i.e. pruning weights of the neural network (see sec. 2). The most popular heuristic for weight removal is weight magnitude pruning [16]. In weight magnitude pruning, we choose a set of target weight tensors. We, then, systematically identify the weights with the smallest magnitudes from the target tensors and remove a specified percentage by setting it to zero. Following the recommendations in [19], we investigated various pruning routines. A two-step approach, wherein we first train the RNN model to convergence followed by sparsification through iterative pruning, produces the best results for our goals of inference performance and sparsity.

The specific pruning methodology we implement is a global unstructured weight magnitude pruning technique. At each step, we carry out weight magnitude pruning on all the weight tensors that constitute the RNN model. The weights to be pruned are selected globally from all the tensors except for the embedding vectors. By selecting the weights globally, we enable the layers that play a larger role in the forward pass to retain a commensurate proportion of its weights. Our rationale for pruning the RNN weights and not the embeddings is based on the perception that RNN weights are more representative across different tasks rather than just language modeling. After each pruning iteration, we allow the model to fine-tune for a few epochs before advancing to the subsequent pruning step. This iterative procedure is repeated until a pre-defined target sparsity level is achieved. For further details on the pruning experimental setup, refer Appendix B.

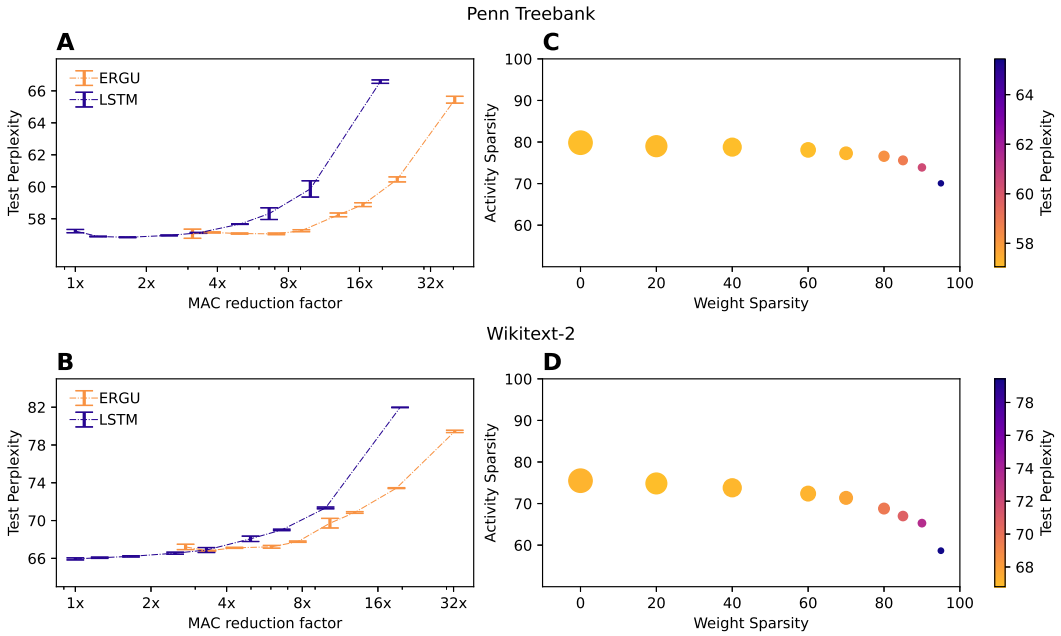

Figure 1: Influence of weight sparsity and activity sparsity on the Penn Treebank and WikiText-2 datasets. **A and B:** Test perplexity versus reduction of MAC operations through weight sparsity (LSTM) and combined activity and weight sparsity (EGRU). We plot the mean test perplexity and corresponding standard deviation over 15 seeds. **C and D:** Activity sparsity vs weight sparsity trade-off for EGRU. The marker size is proportional to the number of MAC operations while the colour represents task performance in terms of test perplexity.

3.3 Efficiency of Sparse Activations and Sparse Connectivity

The efficiency gains of sparse activations and sparse weights complement each other in a multiplicative way. This yields significantly more efficient systems compared to ones that have each of these sparsities separately. Consider the linear transformation $\mathbf{Wa}$. Let $\lambda_\mathrm{a} = \mathbb{E}[\mathbf{a} \neq 0]$ denote the fraction of active neurons in each time step. We then call $\sigma_\mathrm{a} = 1 - \lambda_\mathrm{a}$ the activation sparsity. Likewise, we call σ_w the weight sparsity of $\mathbf{W}$, and denote the fraction of non-zero connections λ_w. The transformation $\mathbf{Wa}$ requires memory access and arithmetics for the non-zero weights $\mathbf{W}_{ij}$ for each non-zero $\mathbf{a}_j$. Hence, we need to load λ_a columns of $\mathbf{W}$ that each have a fraction of λ_w non-zeros. Effectively the fraction of remaining operations compared to a dense vector matrix multiplication is $\lambda_\mathrm{a} \cdot \lambda_\mathrm{w}$.

4 Results

Our main goal for this work is to show that activity sparsity and weight sparsity can be combined for optimal inference efficiency on novel accelerators. According to previous work (see sec. 2), RNNs for language modeling in the computationally feasible domain are more sensitive to the removal of weights compared to RNNs for speech recognition. We, therefore, choose language modeling as the more challenging task to support our main claim and conducted the evaluation on the Penn Treebank [27] and the WikiText-2 [32] word-level language modeling datasets. The task performance metric for both datasets is perplexity (i.e. exponentiated cross-entropy, where lower is better). Since computational efficiency depends on hardware properties, there is no universal metric to quantify efficiency. We choose multiply accumulate (MAC) operations as our metric, which poses a finegrained measure of theoretically required operations on digital hardware.

All models trained for this work follow the architecture presented in [33]. Hence, an embedding look-up table for the word embeddings is followed by three layers of stacked RNNs without skip connections and a linear decoder, whose weights are tied to the embedding layer. DropConnect is applied to the recurrent weights [46]. In contrast to [33], we significantly simplify the optimization procedure by using AdamW instead of their proposed averaged SGD schedule. AdamW speeds up

Model (Weight sparsity)	LSTM		EGRU	
	MAC	Test PPL	MAC	Test PPL
Merity et al. [33]	20.2M	57.3	-	-
Ours (0%)	20.2M	57.1	6.4M	56.6
Ours (20%)	16.2M	56.9	5.3M	57.1
Ours (40%)	12.1M	56.8	4.1M	56.9
Ours (60%)	8.1M	56.9	2.8M	57.0
Ours (70%)	6.1M	57.1	2.2M	57.1
Ours (80%)	4.1M	57.6	1.6M	58.0
Ours (85%)	3.1M	57.7	1.2M	58.7
Ours (90%)	2.0M	58.3	0.9M	60.2
Ours (95%)	1.0M	66.5	0.5M	65.2

(a) Penn Treebank

Model (Weight sparsity)	LSTM		EGRU	
	MAC	Test PPL	MAC	Test PPL
Merity et al. [33]	20.2M	65.8	-	-
Ours (0%)	20.2M	65.7	7.4M	66.6
Ours (20%)	16.2M	65.9	6.0M	66.7
Ours (40%)	12.1M	66.1	4.7M	67.0
Ours (60%)	8.1M	66.4	3.3M	67.1
Ours (70%)	6.1M	66.3	2.6M	67.6
Ours (80%)	4.1M	68.0	2.0M	69.4
Ours (85%)	3.1M	68.9	1.6M	70.7
Ours (90%)	2.0M	71.2	1.1M	73.3
Ours (95%)	1.0M	81.9	0.6M	79.3

(b) WikiText-2

Table 1: Summarized results for the Penn Treebank and WikiText-2 datasets. We record the effective number of MAC operations in the RNN, expressed in millions. Lower value indicates greater efficiency.

the convergence of both Merity et al. [33]'s LSTM and Subramoney et al. [41]'s EGRU by a factor of 3-4. While it would be natural to choose GRU as a baseline for comparison with EGRU, GRU models did not match the LSTM performance in our experiments. This is consistent with the literature that does not report GRU results close to the LSTM baseline. Since overfitting is a major problem on Penn Treebank and WikiText-2, we speculate that the sparse activations of EGRU may impose additional regularization of the model compared to GRU.

Since this work focuses on the interaction of activity sparsity and weight sparsity, we do not use strategies such as mixture-of-softmaxes [50], neural cache [14], or mogrifier gates [30] that can further improve the performance of the LSTM based methods. We note that these methods could be applied on top of all the models presented in this work.

4.1 Weight Sparsity

We evaluate our weight pruning method across sparsity levels ranging from 20% to 95%. Training and subsequent pruning is carried out as per the methodology described in sec. 3.2. The results are presented in detail in tab. 1, and visualized in fig. 1. Weight magnitude pruning effectively compresses our models up to $85\,\%$ with marginal loss in task performance on Penn Treebank. In fact, lightly pruned models, with weight sparsity levels up to $60\,\%$, actually exhibit an improvement in performance. On WikiText-2, slightly lower compression rates are required to maintain task performance. While a similar trend was observed by Zhu and Gupta [54], the results presented in tab. 1 surpass their models significantly by 20 perplexity points.

Keeping the MAC cost constant, if we compare a densely connected EGRU with sparse activity to a sparsely connected LSTM with dense activity, we find that they achieve comparable perplexities. However, combining both weight and activity sparsity yields better results than using either one in isolation.

4.2 Activity Sparsity

Utilizing $85\,\%$ sparse weights and sparse activations, our method reduces the number of MAC operations by a theoretical factor of nearly 20 compared to the dense baseline set by Merity et al. [33] on Penn Treebank. This reduction in computational complexity is accompanied by a minimal perplexity gain of less than 2 points. Fig. 1C and fig. 1D show the trade-off between activity sparsity and weight sparsity for the EGRU model. It is discernible that there is minimal drop in activity sparsity as we increase weight sparsity until we reach high levels of weight sparsity; with the trade-off being slightly higher on WikiText-2. This observation provides further evidence that these two kinds of sparsities are mostly orthogonal and can be combined for greater efficiency in RNNs.

In our hyperparameter search, we observe that weight decay strongly influences both the task performance as well as the activity sparsity. The effect of weight decay on the amount of sparse activations is particularly interesting as it provides a means to trade off sparsity for task performance. We systematically study the influence of weight decay on the EGRU model by training a set of models with different degrees of weight decay applied to the weights and biases separately. Fig. 2 summarizes our findings. The task performance experiences an optimum around a weight decay of

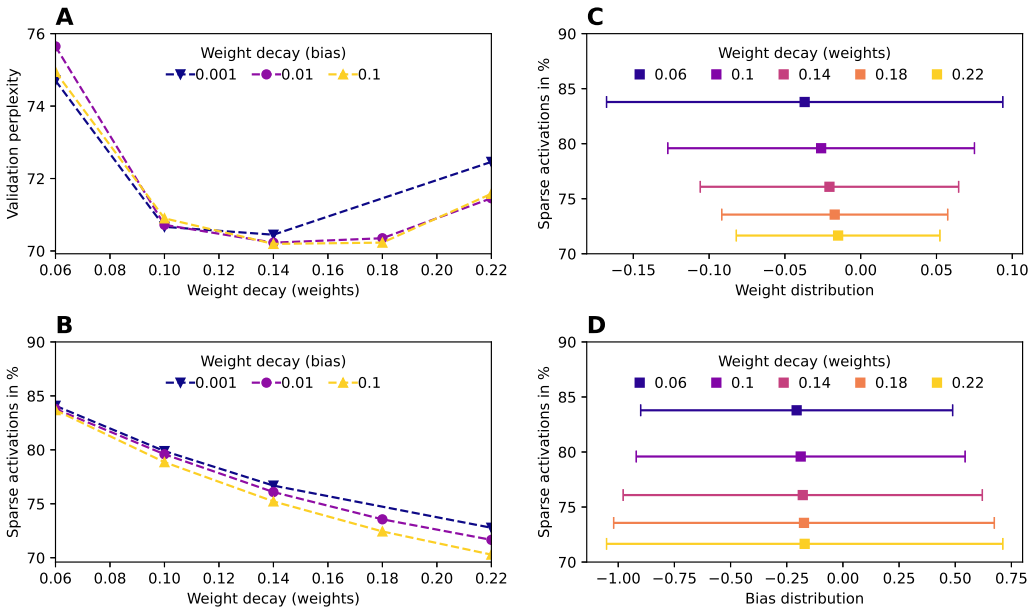

Figure 2: We show the effect of weight decay regularization on the performance and activity sparsity of the EGRU. Therefore, we consider separate degrees of weight decay for the weights and bias, separately. All models are trained on WikiText-2. **A:** Validation perplexity trade-off with weight decay. **B:** Weight decay on both the weights and the bias reduces the amount of sparse activations. **C** and **D**: Effect of weight decay on the distribution of weights and biases for fixed weight decay on the bias of 0.01.

0.14. Weight decay applied to the weights also has a strong influence on the activity of our model, while weight decay applied to the bias has a limited effect. Investigating the distribution of weights and biases, we find that both weights and bias have a tendency to be negative, which drives the cell states below their threshold and promotes sparse activations. With stronger weight decay the distribution of weights concentrates closer around zero. Assuming statistical independence between weights $\mathbf{W}$ and activations $\mathbf{a} = \left[\mathbf{x}^{\langle t \rangle}, \mathbf{y}^{\langle t-1 \rangle}\right]$, the expectation of the preactivations is given by $\mathbb{E}[\mathbf{W}\mathbf{a}+\mathbf{b}] = \mathbb{E}[\mathbf{W}]\mathbb{E}[\mathbf{a}]+\mathbb{E}[\mathbf{b}]$. Hence, negative mean weights tend to drive weights below thresholds, which increases the probability of preactivations passing the threshold $\boldsymbol{\vartheta}$ and reduces sparsity.

5 Discussion

This work shows that activity sparsity and connectivity sparsity complement each other for efficient recurrent neural network inference. While the spiking neural network literature shows promising pruning results for image classification[40, 36, 22, 51], SNNs do not yet deliver competitive baselines even for small-scale sequence modelling problems such as language modeling on Penn Treebank unlike more general event-based RNNs such as the EGRU [41]. To the best of our knowledge this work is the first to show the multiplicative efficiency gain of activation sparsity and connectivity sparsity in the challenging domain of language modeling.

The EGRU model is just one example that shows the potential gains for both fields, i.e. improved benchmark performance for SNNs and improved efficiency for ANNs. In the spirit of this workshop, our results suggest the need for more convergence of methods between deep learning and neuromorphic computing. Such efforts will require joint commitment from accelerator designers and algorithm developers to explore models beyond the deep learning mainstream dominated by GPUs. Unstructured weight sparsity alone does not necessarily justify the design of specific accelerators. However, joint connectivity and activity sparsity could deliver the required reduction in operations that drive the efficiency of irregular accelerators beyond highly regular accelerators.

Acknowledgements

Rishav Mukherji was funded by the DAAD WISE scholarship for the duration of the work. Mark Schöne is fully funded by the Bosch Research Foundation. Khaleel Khan is funded by the German Federal Ministry of Education and Research (BMBF), funding reference 16ME0729K, joint project "EVENTS". The authors gratefully acknowledge the GWK support for funding this project by providing computing time through the Center for Information Services and HPC (ZIH) at TU Dresden.

References

[1] Guillaume Bellec, David Kappel, Wolfgang Maass, and Robert Legenstein. Deep rewiring: Training very sparse deep networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=BJ_wN01C-.

[2] Guillaume Bellec, Darjan Salaj, Anand Subramoney, Robert Legenstein, and Wolfgang Maass. Long short-term memory and Learning-to-learn in networks of spiking neurons. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 787–797. Curran Associates, Inc., 2018.

[3] Ruizhi Chen, Hong Ma, Shaolin Xie, Peng Guo, Pin Li, and Donglin Wang. Fast and efficient deep sparse multi-strength spiking neural networks with dynamic pruning. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2018. doi: 10.1109/IJCNN.2018.8489339.

[4] Yanqi Chen, Zhaofei Yu, Wei Fang, Tiejun Huang, and Yonghong Tian. Pruning of deep spiking neural networks through gradient rewiring. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 1713–1721. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/236. URL https://doi.org/10.24963/ijcai.2021/236. Main Track.

[5] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734. ACL, 2014. doi: 10.3115/v1/d14-1179. URL https://doi.org/10.3115/v1/d14-1179.

[6] Dennis V Christensen, Regina Dittmann, Bernabe Linares-Barranco, Abu Sebastian, Manuel Le Gallo, Andrea Redaelli, Stefan Slesazeck, Thomas Mikolajick, Sabina Spiga, Stephan Menzel, Ilia Valov, Gianluca Milano, Carlo Ricciardi, Shi-Jun Liang, Feng Miao, Mario Lanza, Tyler J Quill, Scott T Keene, Alberto Salleo, Julie Grollier, Danijela Marković, Alice Mizrahi, Peng Yao, J Joshua Yang, Giacomo Indiveri, John Paul Strachan, Suman Datta, Elisa Vianello, Alexandre Valentian, Johannes Feldmann, Xuan Li, Wolfram H P Pernice, Harish Bhaskaran, Steve Furber, Emre Neftci, Franz Scherr, Wolfgang Maass, Srikanth Ramaswamy, Jonathan Tapson, Priyadarshini Panda, Youngeun Kim, Gouhei Tanaka, Simon Thorpe, Chiara Bartolozzi, Thomas A Cleland, Christoph Posch, ShihChii Liu, Gabriella Panuccio, Mufti Mahmud, Arnab Neelim Mazumder, Morteza Hosseini, Tinoosh Mohsenin, Elisa Donati, Silvia Tolu, Roberto Galeazzi, Martin Ejsing Christensen, Sune Holm, Daniele Ielmini, and N Pryds. 2022 roadmap on neuromorphic computing and engineering. *Neuromorphic Computing and Engineering*, 2(2):022501, may 2022. doi: 10.1088/2634-4386/ac4a83. URL https://dx.doi.org/10.1088/2634-4386/ac4a83.

[7] Xiaoliang Dai, Hongxu Yin, and Niraj K. Jha. Grow and prune compact, fast, and accurate lstms. *IEEE Transactions on Computers*, 69(3):441–452, 2020. doi: 10.1109/TC.2019.2954495.

[8] Tri Dao, Daniel Y. Fu, Khaled K. Saab, Armin W. Thomas, Atri Rudra, and Christopher Ré. Hungry Hungry Hippos: Towards Language Modeling with State Space Models, December 2022. URL http://arxiv.org/abs/2212.14052.

[9] Jason K. Eshraghian, Max Ward, Emre O. Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, and Wei D. Lu. Training spiking neural networks using lessons from deep learning. *Proceedings of the IEEE*, 111(9):1016–1054, 2023. doi: 10.1109/JPROC.2023.3308088.

[10] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJl-b3RcF7.

[11] Daniel Y Fu, Tri Dao, Khaled Kamal Saab, Armin W Thomas, Atri Rudra, and Christopher Re. Hungry hungry hippos: Towards language modeling with state space models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=COZDy0WYGg.

[12] Chang Gao, Tobi Delbruck, and Shih-Chii Liu. Spartus: A 9.4 top/s fpga-based lstm accelerator exploiting spatio-temporal sparsity. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15, 2022. doi: 10.1109/TNNLS.2022.3180209.

[13] Samanwoy Ghosh-Dastidar and Hojjat Adeli. Spiking neural networks. *Int. J. Neural Syst.*, 19:295–308, 08 2009. doi: 10.1142/S0129065709002002.

[14] Edouard Grave, Armand Joulin, and Nicolas Usunier. Improving neural language models with a continuous cache. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=B184E5qee.

[15] Albert Gu, Karan Goel, and Christopher Re. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=uYLFoz1vlAC.

[16] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/ae0eb3eed39d2bcef4622b2499a05fe6-Paper.pdf.

[17] Song Han, Junlong Kang, Huizi Mao, Yiming Hu, Xin Li, Yubin Li, Dongliang Xie, Hong Luo, Song Yao, Yu Wang, Huazhong Yang, and William (Bill) J. Dally. Ese: Efficient speech recognition engine with sparse lstm on fpga. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '17, page 75–84, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450343541. doi: 10.1145/3020078.3021745. URL https://doi.org/10.1145/3020078.3021745.

[18] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[19] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *J. Mach. Learn. Res.*, 22(1), jan 2021. ISSN 1532-4435. URL http://jmlr.org/papers/v22/21-0366.html.

[20] Mark Horowitz. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 10–14, 2014. doi: 10.1109/ISSCC.2014.6757323.

[21] Kevin Hunter, Lawrence Spracklen, and Subutai Ahmad. Two sparsities are better than one: unlocking the performance benefits of sparse–sparse networks. *Neuromorphic Computing and Engineering*, 2(3):034004, jul 2022. doi: 10.1088/2634-4386/ac7c8a. URL https://dx.doi.org/10.1088/2634-4386/ac7c8a.

[22] Youngeun Kim, Yuhang Li, Hyoungseob Park, Yeshwanth Venkatesha, Ruokai Yin, and Priyadarshini Panda. Exploring lottery ticket hypothesis in spiking neural networks. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision – ECCV 2022*, pages 102–120, Cham, 2022. Springer Nature Switzerland. ISBN 978-3-031-19775-8.

[23] Mark Kurtz, Justin Kopinsky, Rati Gelashvili, Alexander Matveev, John Carr, Michael Goin, William Leiserson, Sage Moore, Nir Shavit, and Dan Alistarh. Inducing and exploiting activation sparsity for fast inference on deep neural networks. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5533–5543. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/kurtz20a.html.

[24] Shuai Li, Wanqing Li, Chris Cook, Ce Zhu, and Yanbo Gao. Independently recurrent neural network (indrnn): Building a longer and deeper rnn. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5457–5466, 2018. doi: 10.1109/CVPR.2018.00572.

[25] Zonglin Li, Chong You, Srinadh Bhojanapalli, Daliang Li, Ankit Singh Rawat, Sashank J. Reddi, Ke Ye, Felix Chern, Felix Yu, Ruiqi Guo, and Sanjiv Kumar. The lazy neuron phenomenon: On emergence of activation sparsity in transformers. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=TJ2nxciYCk-.

[26] Andrew Lines, Prasad Joshi, Ruokun Liu, Steve McCoy, Jonathan Tse, Yi-Hsin Weng, and Mike Davies. Loihi asynchronous neuromorphic research chip. In *2018 24th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, pages 32–33, 2018. doi: 10.1109/ASYNC.2018.00018.

[27] Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2):313–330, jun 1993. ISSN 0891-2017.

[28] Christian Mayr, Sebastian Hoeppner, and Steve Furber. Spinnaker 2: A 10 million core processor system for brain simulation and machine learning, 2019.

[29] Gábor Melis, Chris Dyer, and Phil Blunsom. On the state of the art of evaluation in neural language models. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=ByJHuTgA-.

[30] Gábor Melis, Tomáš Kočiský, and Phil Blunsom. Mogrifier lstm. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJe5P6EYvS.

[31] Gaurav Menghani. Efficient deep learning: A survey on making deep learning models smaller, faster, and better. *ACM Comput. Surv.*, 55(12), mar 2023. ISSN 0360-0300. doi: 10.1145/3578938. URL https://doi.org/10.1145/3578938.

[32] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Byj72udxe.

[33] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. Regularizing and optimizing LSTM language models. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SyyGPP0TZ.

[34] Sharan Narang, Greg Diamos, Shubho Sengupta, and Erich Elsen. Exploring sparsity in recurrent neural networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=BylSPv9gx.

[35] Daniel Neil, Jun Haeng Lee, Tobi Delbruck, and Shih-Chii Liu. Delta networks for optimized recurrent network computation. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2584–2593. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/neil17a.html.

[36] Thao N. N. Nguyen, Bharadwaj Veeravalli, and Xuanyao Fong. Connection pruning for deep spiking neural networks with on-chip learning. In *International Conference on Neuromorphic Systems 2021*, ICONS 2021, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450386913. doi: 10.1145/3477145.3477157. URL https://doi.org/10.1145/3477145.3477157.

[37] Antonio Orvieto, Samuel L. Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. Resurrecting Recurrent Neural Networks for Long Sequences, March 2023. URL http://arxiv.org/abs/2303.06349.

[38] Angshuman Parashar, Minsoo Rhu, Anurag Mukkara, Antonio Puglielli, Rangharajan Venkatesan, Brucek Khailany, Joel Emer, Stephen W. Keckler, and William J. Dally. Scnn: An accelerator for compressed-sparse convolutional neural networks. In *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, pages 27–40, 2017. doi: 10.1145/3079856.3080254.

[39] Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, Kranthi Kiran GV, Xuzheng He, Haowen Hou, Przemyslaw Kazienko, Jan Kocon, Jiaming Kong, Bartlomiej Koptyra, Hayden Lau, Krishna Sri Ipsit Mantri, Ferdinand Mom, Atsushi Saito, Xiangru Tang, Bolun Wang, Johan S. Wind, Stansilaw Wozniak, Ruichong Zhang, Zhenyuan Zhang, Qihang Zhao, Peng Zhou, Jian Zhu, and Rui-Jie Zhu. Rwkv: Reinventing rnns for the transformer era, 2023.

[40] Nitin Rathi, Priyadarshini Panda, and Kaushik Roy. Stdp-based pruning of connections and weight quantization in spiking neural networks for energy-efficient recognition. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 38(4):668–677, 2019. doi: 10.1109/TCAD.2018.2819366.

[41] Anand Subramoney, Khaleelulla Khan Nazeer, Mark Schöne, Christian Mayr, and David Kappel. Efficient recurrent architectures through activity sparsity and sparse back-propagation through time. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=lJdOlWg8td.

[42] Aboozar Taherkhani, Ammar Belatreche, Yuhua Li, Georgina Cosma, Liam P. Maguire, and T.M. McGinnity. A review of learning in biologically plausible spiking neural networks. *Neural Networks*, 122:253–272, 2020. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2019.09.036. URL https://www.sciencedirect.com/science/article/pii/S0893608019303181.

[43] Sachin S. Talathi and Aniket Vartak. Improving performance of recurrent neural network with relu nonlinearity. In *International Conference on Learning Representations: Workshop Track*, 2016.

[44] Amirhossein Tavanaei, Masoud Ghodrati, Saeed Reza Kheradpisheh, Timothée Masquelier, and Anthony Maida. Deep learning in spiking neural networks. *Neural Networks*, 111: 47–63, 2019. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2018.12.002. URL https://www.sciencedirect.com/science/article/pii/S0893608018303332.

[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[46] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1058–1066, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR. URL https://proceedings.mlr.press/v28/wan13.html.

[47] Stanisław Woźniak, Angeliki Pantazi, Thomas Bohnstingl, and Evangelos Eleftheriou. Deep learning incorporating biologically inspired neural dynamics and in-memory computing. *Nature Machine Intelligence*, 2(6):325–336, Jun 2020. ISSN 2522-5839. doi: 10.1038/s42256-020-0187-0. URL https://doi.org/10.1038/s42256-020-0187-0.

[48] Hao Wu, Patrick Judd, Xiaojie Zhang, Mikhail Isaev, and Paulius Micikevicius. Integer quantization for deep learning inference: Principles and empirical evaluation. *ArXiv*, abs/2004.09602, 2020.

[49] Kashu Yamazaki, Viet-Khoa Vo-Ho, Darshan Bulsara, and Ngan Le. Spiking neural networks and their applications: A review. *Brain Sciences*, 12(7), 2022. ISSN 2076-3425. doi: 10.3390/brainsci12070863. URL https://www.mdpi.com/2076-3425/12/7/863.

[50] Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. Breaking the softmax bottleneck: A high-rank RNN language model. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HkwZSG-CZ.

[51] Ruokai Yin, Youngeun Kim, Yuhang Li, Abhishek Moitra, Nitin Satpute, Anna Hambitzer, and Priyadarshini Panda. Workload-balanced pruning for sparse spiking neural networks, 2023.

[52] Zhe Yuan, Yongpan Liu, Jinshan Yue, Yixiong Yang, Jingyu Wang, Xiaoyu Feng, Jian Zhao, Xueqing Li, and Huazhong Yang. Sticker: An energy-efficient multi-sparsity compatible accelerator for convolutional neural networks in 65-nm cmos. *IEEE Journal of Solid-State Circuits*, 55(2):465–477, 2020. doi: 10.1109/JSSC.2019.2946771.

[53] Jie-Fang Zhang, Ching-En Lee, Chester Liu, Yakun Sophia Shao, Stephen W. Keckler, and Zhengya Zhang. Snap: An efficient sparse neural acceleration processor for unstructured sparse deep neural network inference. *IEEE Journal of Solid-State Circuits*, 56(2):636–647, 2021. doi: 10.1109/JSSC.2020.3043870.

[54] 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.

[55] Rui-Jie Zhu, Qihang Zhao, Guoqi Li, and Jason K. Eshraghian. Spikegpt: Generative pre-trained language model with spiking neural networks, 2023.

A Extended Results

Table 2: Penn Treebank results

Model	Weight Sparsity	MAC	Validation Perplexity		Test Perplexity	
			Min	Mean ± Std Dev	Min	Mean ± Std Dev
EGRU	0%	6.4M	60.81	61.12±0.26	56.61	57.06±0.29
	20%	5.3M	61.15	61.22±0.04	57.10	57.15±0.03
	40%	4.1M	61.01	61.13±0.05	57.00	57.07±0.03
	60%	2.8M	61.04	61.12±0.04	57.01	57.06±0.05
	70%	2.2M	61.22	61.27±0.04	57.08	57.25±0.06
	80%	1.6M	62.06	62.31±0.15	58.04	58.24±0.12
	85%	1.2M	62.76	63.01±0.17	58.74	58.88±0.12
	90%	0.9M	64.38	64.66±0.22	60.22	60.46±0.16
	95%	0.5M	69.64	70.03±0.31	65.22	65.44±0.21
LSTM	0%	20.2M	59.81	60.04±0.18	57.06	57.23±0.10
	20%	16.2M	59.61	59.64±0.02	56.87	56.89±0.02
	40%	12.1M	59.53	59.57±0.02	56.81	56.84±0.02
	60%	8.1M	59.58	59.61±0.02	56.93	56.96±0.03
	70%	6.1M	59.73	59.77±0.03	57.06	57.12±0.03
	80%	4.1M	60.28	60.36±0.04	57.63	57.67±0.02
	85%	3.1M	61.04	61.10±0.03	57.67	58.32±0.37
	90%	2.0M	62.88	62.96±0.04	58.30	59.87±0.51
	95%	1.0M	69.82	70.04±0.10	66.47	66.58±0.11

Table 3: WikiText-2 results

Model	Weight Sparsity	MAC	Validation Perplexity		Test Perplexity	
			Min	Mean ± Std Dev	Min	Mean ± Std Dev
EGRU	0%	7.4M	69.74	70.26±0.34	66.64	67.21±0.28
	20%	6.0M	69.77	69.88±0.06	66.71	66.80±0.04
	40%	4.7M	70.28	70.37±0.05	67.03	67.11±0.06
	60%	3.3M	70.47	70.57±0.05	67.12	67.22±0.15
	70%	2.6M	71.17	71.25±0.04	67.64	67.76±0.07
	80%	2.0M	73.00	73.08±0.05	69.40	69.72±0.51
	85%	1.6M	74.42	74.53±0.07	70.70	70.85±0.09
	90%	1.1M	77.18	77.37±0.09	73.33	73.44±0.05
	95%	0.6M	83.89	84.11±0.11	79.28	79.44±0.12
LSTM	0%	20.2M	68.68	68.85±0.12	65.66	65.94±0.11
	20%	16.2M	68.73	68.82±0.06	65.93	66.06±0.07
	40%	12.1M	68.73	68.88±0.10	66.11	66.21±0.07
	60%	8.1M	69.00	69.22±0.11	66.41	66.55±0.11
	70%	6.1M	69.52	69.78±0.14	66.32	66.88±0.26
	80%	4.1M	70.61	70.86±0.11	67.96	68.07±0.28
	85%	3.1M	71.94	72.11±0.08	68.89	69.01±0.07
	90%	2.0M	74.56	74.69±0.08	71.20	71.34±0.09
	95%	1.0M	86.02	86.06±0.03	81.91	81.97±0.02

B Pruning Methodology

To achieve the best results after pruning we tested out multiple pruning techniques. We tried out parallel training and sparsification methodologies such as the lottery ticket hypothesis mentioned in [10] and the sparsify during training method mentioned in [19]. However we achieved much better results by first training to convergence and then pruning. Hoefler et al. [19] suggests pruning to the target sparsity in one go however we experimented with an iterative pruning method as described in sec. 3.2

For the iterative pruning method, we vary the learning rate and number of pruning steps for each level of target sparsity we tried out . The final results are then obtained by repeating the best methodology for each target on 15 different seeds. For low sparsity, it is beneficial to carry out the pruning in one go whereas for higher sparsity we obtain better results on carrying out multiple steps of pruning and subsequent fine-tuning. The trends are visualized in fig. 3.

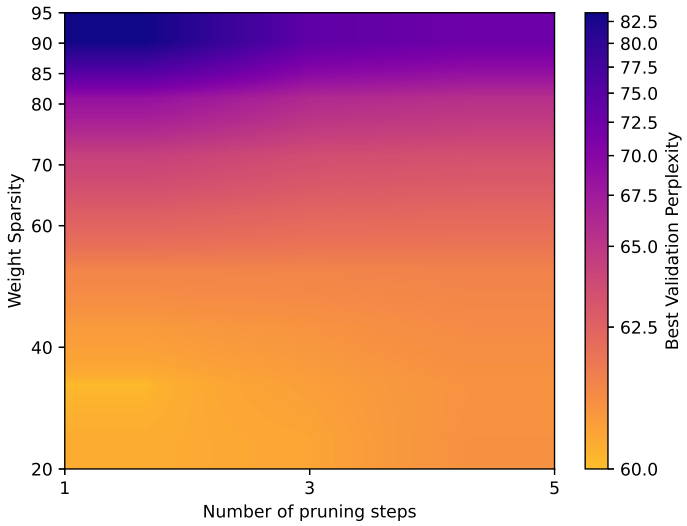

Figure 3: Influence of number of pruning steps on the results for different target weight sparsities. Results are displayed for EGRU experiments on Penn Treebank dataset. Other experimental setups follow similar trends

Similarly for learning rate, lower sparsity targets achieved better performance with a lower learning rate ($0.1\times$ baseline training learning rate) while higher sparsity models required larger learning rates (equal to the baseline training learning rate).

C Network Activity

Our investigation of activity sparsity in sec. 4.2 showed how the distribution of weights affects the activity of EGRU networks when weight decay is applied. We provide more detail on the activity spectrum of EGRU here. Fig. 4 shows the distribution of activity for the layers individually. The lower weight decay, the more the activities move towards zero. An anomaly is the output layer, which has significantly higher activity than the bottom layers. This observation could be driven by the learning objective of language modeling. At each point in time, the model has to output a candidate word embedding vector. This candidate vector is compared to all the (learned) word embedding vectors from the dictionary via dot-product. Since, we don't apply a decoder layer on top of the EGRU following Merity et al. [33], the candidate word embeddings have positive and zero entries only. Yet, the word embeddings are learned and perhaps take any value, especially not necessarily sparse values. The learning objective of matching the correct next word embedding with the output of the final EGRU layer might force a high activity in the final layer. Future work can consider more beneficial decoding strategies to keep activity low in the final layer as well.

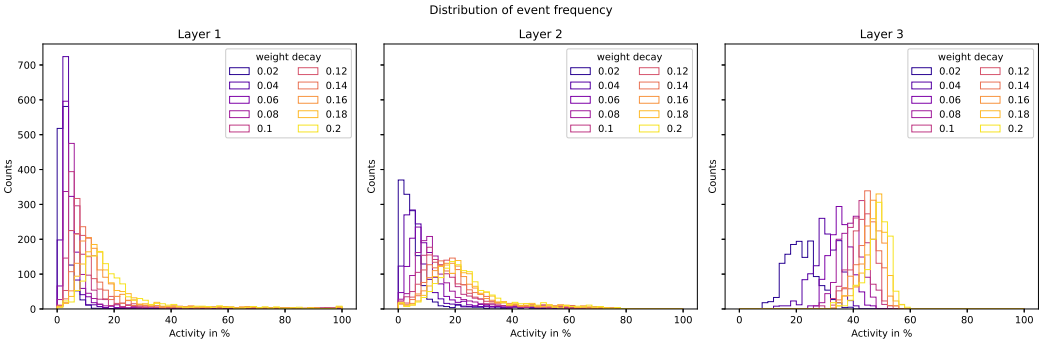

Figure 4: Histogram of activity of the EGRU neurons in each layer. For example, an activity of $20\,\%$ denotes that a neuron's output is non-zero $20\,\%$ of the time, hence saving operations $80\,\%$ of the time.

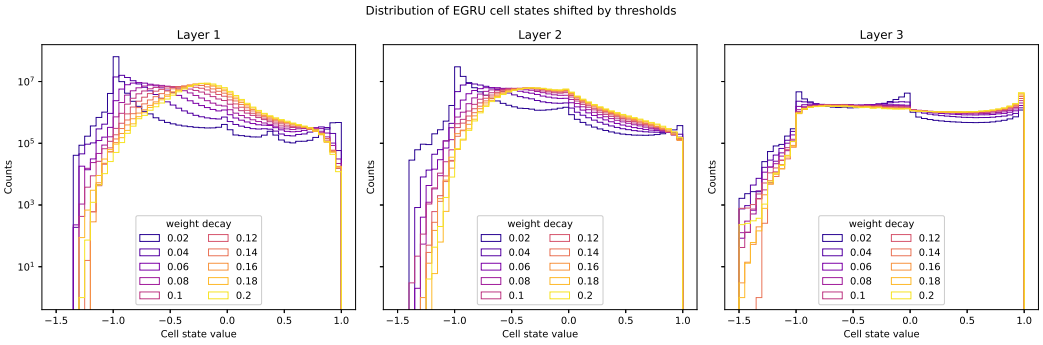

Figure 5: Histogram of cell state values shifted by the thresholds $\mathbf{c} - \boldsymbol{\vartheta}$ in each layer.

Fig. 5 visualizes the distribution of the normalized cell states $\mathbf{c} - \boldsymbol{\vartheta}$ of EGRU on the Penn Treebank validation set. We observe that higher weight decay motivates the cell states to operate closer to their thresholds. At the same time, cells with smaller weight decay operate a significant amount of time far away from the threshold in the negative regime. This makes the output signals $\mathbf{y}$ less sensitive to inputs, which increases the regularity of EGRU networks compared to GRU.

D Limitations

Our work is based on unstructured weight sparsity and irregular activation sparsity. Both sparsities are difficult to accelerate on contemporary hardware. Unstructured weight sparsity is not aligned with the regular memory access instructions of GPU programming. Yet, the sparsity pattern is known at compile time for inference applications. This simplifies the design of specialized accelerators. Our activation sparsity is irregular in the sense that it cannot be predicted ahead of time. Hence, the instructions depend on context, and the compiler can hardly optimize the system e.g. by fetch weights from memory in advance. Efficiently simulating our method requires an event-based programming paradigm, which only few accelerators such as SpiNNaker2 [28] or Loihi [26] support.

We find that EGRU requires larger word embeddings than LSTM for the language modeling task. This might be due to the sparsity of the feature map, which does not align with the dense embedding vectors. The larger word embeddings introduce additional MAC operations, which limits the effective reduction of MAC operations through activity sparsity to a factor of 3 (see tab. 1).