An efficient RNN Language Model using activity sparsity and sparse back-propagation through time

Mark Schöne[1]**, Khaleelulla Khan Nazeer**[1]**, Christian Mayr**[1]**, David Kappel**[2]**, Anand Subramoney**[2]
[1] Faculty of Electrical and Computer Engineering, Technische Universität Dresden, Dresden, Germany
[2] Institute for Neural Computation, Ruhr University Bochum, Germany
{khaleelulla.khan_nazeer,mark.schoene,christian.mayr}@tu-dresden.de
{anand.subramoney,david.kappel}@ini.rub.de

Abstract

Transformers have displaced recurrent neural networks (RNN) for language modelling due to their scalability on ubiquitous GPUs. However, resource constrained systems are confronted with the high computational cost and memory footprint of both training and inference with transformer language models. RNN language models are a potential alternative, but there are gaps to bridge in terms of capabilities. The sequential dependence of activations together with the memory and computational requirements arising from propagating the activations of all the neurons at every time step to every connected neuron make RNNs harder to train efficiently. We propose an architecture inspired by biological neuron dynamics, that makes the communication between RNN units sparse and discrete along the forward direction. We demonstrate our sparsity model with a gated recurrent unit (GRU). The recurrent units emit discrete events for communication triggered by a gating mechanism. Thus, no information needs to be communicated to other units in the absence of events. We show that this makes backpropagation through time (BPTT) and inference computationally sparse. With access to neuromorphic accelerators this unstructured sparsity can realize efficiency gains for energy and memory usage. Overall, we achieve efficiency without compromising task performance, demonstrating competitive performance compared to state-of-the-art recurrent network models in language modelling.

1 Introduction

Large scale models such as GPT-3 [4], switch transformers [6] and DALL-E [31] demonstrate that scaling up deep learning models to billions of parameters improves not just their performance but leads to entirely new forms of generalisation. Due to their computational cost and memory footprint, transformers are difficult to employ in resource constrained systems. Recurrent neural networks (RNNs) may provide a viable alternative in such low-resource environments, but still require further algorithmic and computational optimizations. While it is unknown if scaling up recurrent neural networks can lead to similar forms of generalisation, the limitations on scaling them up preclude studying this possibility. The dependence of each time step's computation on the previous time step's output prevents easy parallelisation of the model computation. Moreover, propagating the activations of all the units in each time step is computationally inefficient and leads to high memory requirements when training with backpropagation through time (BPTT).

While allowing extraordinary task performance, the biological brain's recurrent architecture is extremely energy efficient [20]. One of the brain's strategies to reach these high levels of efficiency is activity sparsity. In the brain, (asynchronous) event-based communication results from the properties of the specific physical and biological substrate on which the brain is built. Biologically realistic spiking neural networks and neuromorphic hardware aim to use these principles to build energy-efficient software and

The 2$^{\text{nd}}$ Workshop on Efficient Natural Language and Speech Processing (NeurIPS 2022 Workshop).

hardware models [32, 35]. However, despite progress in recent years, their task performance has been relatively limited for real-world tasks compared to recurrent architectures based on LSTM and GRU.

In this work, we propose an activity sparsity mechanism inspired by biological neuron models, to reduce the computation required by RNNs at each time step. Our method applies a Heaviside thresholding function with surrogate gradients to the hidden state at each time step to sparsify computation and communication between RNN units. Notably, the theoretically required compute scales with the number of active hidden units instead of the total number of hidden units or length of input sequence.

2 Related work

Activity sparsity in RNNs has been proposed previously in various forms [12, 25, 26], but only focusing on achieving it during inference. QRNNs [3], SRUs [16] and IndRNNs [17] target increasing the parallelism in a recurrent network without directly using activity sparsity. Unlike [6], our architecture uses a unit-local decision making process for the dynamic activity sparsity, specifically for recurrent architecture. The cost of computation is lower in our model compared to [26], and can be implemented to have parallel computation of intermediate updates between events, while also being activity sparse in its output.

Models based on sparse communication [37] for scalability have been proposed recently for feedforward networks as a dynamic form of parameter-sparsity [10]. But, parameter/model-sparsity and pruning [30] is, in general, orthogonal to and complementary with our method for activity sparsity, and can easily be combined for additional gains. While thresholding leads to discrete events in our model, the values communicated are real-valued unlike work on integer valued RNNs [34].

Recent work such as Gu et al. [8, 9] has shown that RNN-like models are state-of-the-art at long range dependency modelling. Our model does not attempt at solving the long-range dependency modelling problem, but rather aims to produce more efficient RNNs.

Biologically realistic spiking networks [18] are often implemented using event-based updates and have been scaled to huge sizes [14], albeit without any task-related performance evaluation. Models for deep learning with recurrent spiking networks [1, 33] mostly focus on modeling biologically realistic memory and learning mechanisms. Moreover, units in a spiking neural network implement dynamics based on biology and communicate solely through unitary events, while units in an EGRU send real-valued signals to other units, and have more general dynamics.

3 An activity sparsity mechanism for recurrent architectures

We introduce an activity sparsity mechanism for RNNs consisting of a rectifier Eq. (1) and a clearing mechanism Eq. (2). Assume an RNN with hidden state vector $\mathbf{c} = (c_1,...,c_n)$ and update equation $c^t = f(x^t, c^{t-1})$. Biological neuron models such as spiking neural networks [18] create sparsity by only communicating entries c_i of the state vector if they surpass a threshold. To design a similar mechanism for general RNNs, we define a gated state vector $\mathbf{y} = (y_1,...,y_n)$ as an element-wise gated function of the hidden state vector c as

$$y_i^t = c_i^t H\left(c_i^t - \vartheta_i\right), \tag{1}$$

where $H(\cdot)$ is the Heaviside step function. We attach surrogate gradients $\frac{\mathrm{d}H(c_i)}{\mathrm{d}c_i} = \lambda \max(0, 1 - |c_i|/\epsilon)$ to the non-differentiable Heaviside function similar to [1] to train the rectifier and the thresholds ϑ_i, where λ and ϵ are constant parameters. For a visualization of the surrogate see Fig. 1C. Our model achieves sparsity by passing the gated state vector y to the RNN update equation as an input

$$c_i^t = f(\mathbf{x}^t, \mathbf{y}^{t-1}, c_i^{t-1}) - y_i^{t-1} \quad \text{for} \quad i = 1,...,n. \tag{2}$$

Note that the update of the entry c_i depends on the full external state vector $\mathbf{y}$, but only on its local hidden state vector entry c_i. As visualized in Fig. 1D only the sparse gated state vector is communicated to other neurons. The $-y_i^{t-1}$ term in Eq. (2) forces the the cell state below the threshold if a signal was emitted in the previous timestep, i.e. $y_i^{t-1} \neq 0$. This term models the membrane potential reset of biological neuron models after emitting a spike. Fig. 1B illustrates the mechanism.

3.1 Event-based Gated Recurrent Unit

We apply the above activity sparsity mechanism to the Gated Recurrent Unit (GRU) [5] as a case study, and call our model Event-based Gated Recurrent Unit (EGRU). The GRU consists of internal gating

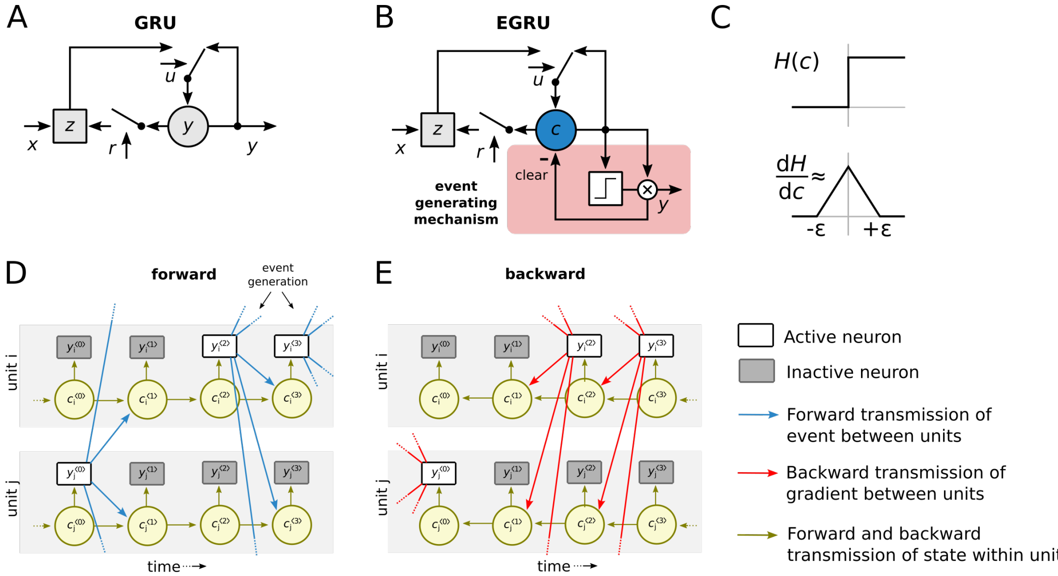

Figure 1: Illustration of our sparsity-generating mechanism. **A:** GRU unit adapted from [5]. **B:** EGRU unit with activity sparsity mechanism **C:** Heaviside function and surrogate gradient. **D:** Forward state dynamics for two EGRU units (i and j). **E:** Activity-sparse backward dynamics for two EGRU units (i and j).

variables for updates ($\mathbf{u}$) and a reset ($\mathbf{r}$), that determine the behavior of the internal state $\mathbf{c}$

$$\mathbf{u}^t = \sigma\big(\mathbf{W}_u\big[\mathbf{x}^t, \mathbf{y}^{t-1}\big] + \mathbf{b}_u\big), \quad \mathbf{r}^t = \sigma\big(\mathbf{W}_r\big[\mathbf{x}^t, \mathbf{y}^{t-1}\big] + \mathbf{b}_r\big), \tag{3}$$

The state variable $\mathbf{z}$ determines the interaction between external input $\mathbf{x}$ and the internal state. The dynamics EGRU at time step t, is given by the set of vector-valued update equations:

$$\mathbf{z}^t = g\big(\mathbf{W}_z\big[\mathbf{x}^t, \mathbf{r}^t \odot \mathbf{y}^{t-1}\big] + \mathbf{b}_z\big), \tag{4}$$

$$\mathbf{c}^t = \mathbf{u}^t \odot \mathbf{z}^t + (1 - \mathbf{u}^t) \odot \mathbf{c}^{t-1} - \mathbf{y}^{t-1} \qquad \mathbf{y}^t = \mathbf{c}^t H\big(\mathbf{c}^t - \vartheta\big). \tag{5}$$

where $\mathbf{W}_{u/r/z}$, $\mathbf{b}_{u/r/z}$ denote network weights and biases, $\odot$ denotes the element-wise (Hadamard) product, and $\sigma(\cdot)$ is the vectorized sigmoid function. The notation $\big[\mathbf{x}^t, \mathbf{y}^{t-1}\big]$ denotes vector concatenation. The function $g(\cdot)$ is an element-wise nonlinearity (typically the hyperbolic tangent function).

3.2 Computation and memory reduction due to sparsity

We refer to sparsity as the fraction of zero entries in a state vector (forward) or gradient vector (backward). During inference, an activity sparsity of α leads to a reduction of multiply-accumulate (MAC) operations by a factor of α. To arrive at the computational savings induced by activity sparsity, we have to consider the surrogate gradient defined above. The surrogate gradient is non-zero for values of states c_i between $\vartheta_i + \varepsilon$ and $\vartheta_i - \varepsilon$ as shown in the inset in Fig. 1C. For cell state values outside of this interval, the backpropagated gradients corresponding to these cells are 0, making the backward-pass sparse (see Fig.1E for an illustration). Hence, only a subset of the activations needs to be stored for later use, therefore reducing the memory usage and required MAC operations. This unstructured sparsity does not directly translate to efficiency gains on GPUs. However, recent neuromorphic devices allow to leverage dynamic and unstructured sparsity [11, 27] even in larger scale models.

4 Language Modeling Results

We evaluate our model on language modeling tasks based on the PennTreebank [19] dataset and the WikiText-2 dataset [22]. We exclusively focus on the RNN model itself in this work, and do not consider techniques such as transformer-based word embeddings, neural cache models [7] or dynamic evaluation [15]. A strong baseline for gate-based RNN architectures was established by [23]. Similarly, our models consist of three stacked RNN cells without skip connections. DropConnect [36] is applied to the hidden-to-hidden weights. The weights of the final softmax layer were tied to the embedding layer [13, 29].

Hyperparameters were tuned using a broad Bayesian search using Weights & Biases [2]. The surrogate gradient parameter ϵ and the initialization of the thresholds φ_i are hyperparameters of this model. We choose to re-parameterize thresholds with a sigmoid function to limit their domain to the interval $[0, 1]$. With τ_i drawn from a normal distribution $\tau_i \sim \mathcal{N}(\mu, \sqrt{2})$, the thresholds are initialized as $\varphi_i = 1/(1 + \exp(-\tau_i))$. The hyperparameters μ of the normal distribution and ϵ of the surrogate gradient are tuned jointly with the standard training parameters in the Bayesian search. All our models are optimized with Adam for 2000 epochs.

The results presented in Tab. 1 and Tab. 2 show that EGRU achieves competitive performance with AWD-LSTM [23]. At the same time, EGRU inherently exhibits activity sparsity that reduces the theoretically required computational operations. In our experiments, GRUs did not reach the performance of LSTM variants on this task, which, to the best of our knowledge, is consistent with recent RNN language modeling literature [21, 23].

We implement EGRU in Haste [24] and observe shorter wallclock times than PyTorch's [28] GRU implementation. Note that GPUs do not take advantage of unstructured sparsity. Further experimental details, and statistics over different runs can be found in the supplement sections A and table S1 respectively.

model	hidden dim	para-meters	effective MAC	validation	test	activity sparsity	backward sparsity
LSTM [21]	-	24M	-	61.8	59.6	-	-
AWD-LSTM [23]	1150	24M	24M	60.0	57.3	-	-
GRU	1350	24M	24M	68.5	66.3	-	-
EGRU	1350	31M	7.3M	61.3	58.7	83.0 %	47.4 %
EGRU	2000	54M	11.8M	60.9	58.8	85.8 %	39.8 %

Table 1: Model comparison on PennTreebank. Validation and test scores are given as perplexities, where lower is better. Effective MAC operations consider the layer-wise sparsity in the forward pass.

model	hidden dim	para-meters	effective MAC	validation	test	activity sparsity	backward sparsity
LSTM [21]	-	24M	-	69.3	65.9	-	-
AWD-LSTM [23]	1150	33M	32M	68.6	65.8	-	-
GRU	1350	43M	30.1M	76.8	73.0	-	-
EGRU	1350	48M	7.9M	72.2	69.0	80.3 %	49.6 %
EGRU	2000	71M	11.3M	71.4	68.6	83.9 %	46.7 %

Table 2: Model comparison on WikiText-2. Validation and test scores are given as perplexities, where lower is better. Effective MAC operations consider the layer-wise sparsity in the forward pass. Model parameters were optimized on Penn Treebank and transfered to WikiText-2.

5 Discussion

This work introduces a biologically inspired activity sparsity mechanism for recurrent neural networks. To the best of our knowledge, this is the first demonstration of such activity sparsity mechanisms that yields strong benchmark performance.

We stress that EGRU is a case study of this generally applicable activity sparsity mechanism. Activity sparsity leads to a significant reduction of required operations both during inference and backpropagation through time. The theoretical efficiency of this model can translate into gains in energy efficiency when implemented using event-based software primitives, and recent neuromorphic devices that allow leveraging of dynamic and unstructured sparsity [11, 27]. These same properties will also allow the model to scale to heterogenous compute resources. At the same time, they can achieve orders of magnitude higher energy efficiency in terms of operations per watt compared to GPUs. On neuromorphic devices with on-chip memory in the form of a crossbar array, the activity sparsity directly translates into energy efficiency. For larger models that need off-chip memory, activity sparsity needs to be combined with parameter-sparsity to reduce energy-intensive memory access operations.

In future work, we will jointly explore the benefits of activity sparsity and parameter-sparsity for energy-efficient hardware implementations.

6 Acknowledgements

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. We acknowledge the use of Fenix Infrastructure resources, which are partially funded from the European Union's Horizon 2020 research and innovation programme through the ICEI project under the grant agreement No. 800858. MS is fully funded by a grant of the Bosch Research Foundation. AS is funded by the Ministry of Culture and Science of the State of North Rhine-Westphalia, Germany. KKN is funded by the German Federal Ministry of Education and Research (BMBF) within the KI-ASIC project (16ES0996). AS would also like to thank Darjan Salaj and Franz Scherr for insightful early discussions.

References

[1] G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, and W. 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.

[2] L. Biewald. Experiment tracking with weights and biases, 2020. URL https://www.wandb.com/. Software available from wandb.com.

[3] J. Bradbury, S. Merity, C. Xiong, and R. Socher. Quasi-Recurrent Neural Networks. *arXiv:1611.01576 [cs]*, Nov. 2016.

[4] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language Models are Few-Shot Learners. *arXiv:2005.14165 [cs]*, July 2020.

[5] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[6] W. Fedus, B. Zoph, and N. Shazeer. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. *arXiv:2101.03961 [cs]*, Jan. 2021.

[7] E. Grave, A. Joulin, and N. Usunier. Improving neural language models with a continuous cache. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=B184E5qee.

[8] A. Gu, I. Johnson, K. Goel, K. Saab, T. Dao, A. Rudra, and C. Ré. Combining Recurrent, Convolutional, and Continuous-time Models with Linear State Space Layers. In *Advances in Neural Information Processing Systems*, volume 34, pages 572–585. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/hash/05546b0e38ab9175cd905eebcc6ebb76-Abstract.html.

[9] A. Gu, K. Goel, and C. Ré. Efficiently Modeling Long Sequences with Structured State Spaces, Aug. 2022. URL http://arxiv.org/abs/2111.00396.

[10] T. Hoefler, D. Alistarh, T. Ben-Nun, N. Dryden, and A. Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241):1–124, 2021.

[11] S. Höppner, Y. Yan, A. Dixius, S. Scholze, J. Partzsch, M. Stolba, F. Kelber, B. Vogginger, F. Neumärker, G. Ellguth, et al. The spinnaker 2 processing element architecture for hybrid digital neuromorphic computing. *arXiv preprint arXiv:2103.08392*, 2021.

[12] K. L. Hunter, L. Spracklen, and S. Ahmad. Two sparsities are better than one: Unlocking the performance benefits of sparse-sparse networks, 2021.

[13] H. Inan, K. Khosravi, and R. Socher. Tying word vectors and word classifiers: A loss framework for language modeling. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=r1aPbsFle.

[14] J. Jordan, T. Ippen, M. Helias, I. Kitayama, M. Sato, J. Igarashi, M. Diesmann, and S. Kunkel. Extremely Scalable Spiking Neuronal Network Simulation Code: From Laptops to Exascale Computers. *Frontiers in Neuroinformatics*, 12:2, 2018. ISSN 1662-5196.

[15] B. Krause, E. Kahembwe, I. Murray, and S. Renals. Dynamic evaluation of neural sequence models. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2766–2775. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/krause18a.html.

[16] T. Lei, Y. Zhang, S. I. Wang, H. Dai, and Y. Artzi. Simple Recurrent Units for Highly Parallelizable Recurrence. *arXiv:1709.02755 [cs]*, Sept. 2017.

[17] S. Li, W. Li, C. Cook, C. Zhu, and Y. Gao. Independently Recurrent Neural Network (IndRNN): Building A Longer and Deeper RNN. *arXiv:1803.04831 [cs]*, May 2018.

[18] W. Maass. Networks of spiking neurons: the third generation of neural network models. *Neural networks*, 10(9):1659–1671, 1997.

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

[20] C. Mead. How we created neuromorphic engineering. *Nature Electronics*, 3(7):434–435, 2020.

[21] G. Melis, C. Dyer, and P. Blunsom. On the state of the art of evaluation in neural language models. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=ByJHuTgA-.

[22] S. Merity, C. Xiong, J. Bradbury, and R. Socher. Pointer sentinel mixture models, 2016. URL https://arxiv.org/abs/1609.07843.

[23] S. Merity, N. S. Keskar, and R. Socher. Regularizing and Optimizing LSTM Language Models. *arXiv:1708.02182 [cs]*, Aug. 2017.

[24] S. Nanavati. Haste: a fast, simple, and open rnn library. https://github.com/lmnt-com/haste/, Jan 2020.

[25] D. Neil, M. Pfeiffer, and S.-C. Liu. Phased LSTM: Accelerating Recurrent Network Training for Long or Event-based Sequences. *arXiv:1610.09513 [cs]*, Oct. 2016.

[26] D. Neil, J. H. Lee, T. Delbruck, and S.-C. Liu. Delta Networks for Optimized Recurrent Network Computation. In *International Conference on Machine Learning*, pages 2584–2593. PMLR, July 2017.

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

[28] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[29] O. Press and L. Wolf. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163, Valencia, Spain, Apr. 2017. Association for Computational Linguistics. URL https://aclanthology.org/E17-2025.

[30] R. K. Ramakrishnan, E. Sari, and V. P. Nia. Differentiable Mask for Pruning Convolutional and Recurrent Networks, Apr. 2020. URL http://arxiv.org/abs/1909.04567.

[31] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever. Zero-Shot Text-to-Image Generation. *arXiv:2102.12092 [cs]*, Feb. 2021.

[32] K. Roy, A. Jaiswal, and P. Panda. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575(7784):607–617, 2019.

[33] D. Salaj, A. Subramoney, C. Kraisnikovic, G. Bellec, R. Legenstein, and W. Maass. Spike frequency adaptation supports network computations on temporally dispersed information. *eLife*, 10:e65459, July 2021. ISSN 2050-084X. doi: 10.7554/eLife.65459. URL https://doi.org/10.7554/eLife.65459.

[34] E. Sari, V. Courville, and V. P. Nia. iRNN: Integer-only Recurrent Neural Network, Feb. 2022. URL http://arxiv.org/abs/2109.09828.

[35] C. D. Schuman, T. E. Potok, R. M. Patton, J. D. Birdwell, M. E. Dean, G. S. Rose, and J. S. Plank. A survey of neuromorphic computing and neural networks in hardware. *arXiv preprint arXiv:1705.06963*, 2017.

[36] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus. Regularization of neural networks using dropconnect. In S. Dasgupta and D. 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.

[37] M. Yan, N. Meisburger, T. Medini, and A. Shrivastava. Distributed slide: Enabling training large neural networks on low bandwidth and simple cpu-clusters via model parallelism and sparsity. *arXiv preprint arXiv:2201.12667*, 2022.