Positional Encoding in the Context of Memristor-Based Analog Computation for Automatic Speech Recognition

Benedikt Hilmes [**], *Nick Rossenbach*, *Ralf Schlüter*

Machine Learning and Human Language Technology Group, Faculty of Computer Science,
RWTH Aachen University, Aachen, Germany
Apptek GmbH, Aachen, Germany
{hilmes, rossenbach, schlueter}@ml.rwth-aachen.de

Abstract

Memristors provide a new chance for resource-efficient computation of neural models for natural language processing by enabling analog execution of vector-matrix-multiplication. Yet, computations on these devices are currently subject to larger distortion, both in weight programming and execution. In this work, we identify large output values of transformed positional encodings to cause major degradation within analog-to-digital conversion (ADC) as part of memristor-based computation. By adjusting the proportion of weight and precision bits of the ADC of specific memristor layers, we reduce the degradation of the execution by $\sim$ 50% relative, while keeping the estimated energy consumption stable. Additionally, we investigate scenarios where the ADC cannot be modified. In that case the degradation can be reduced by $\sim$ 30% relative after removing encoding-related linear transformations.

Index Terms: speech recognition, memristive hardware, efficient computing, positional encoding

1. Introduction

The increase in model size of modern day natural language processing models pushes the need for efficient computing. One family of device types that offer efficient computation of vector-matrix-multiplication (VMM) based operations is so-called memristors. With physically programmable resistance states within a crossbar array, such devices can compute VMM operations in the analog domain. Yet, the devices are still in an early stage of development and exhibit significant stochastic behavior, both during programming and execution [1]. Combining a high number of memristors for larger matrices might cause the inaccuracies to ramp up and affect the performance. This means that especially for large neural networks, where a huge number of memristive devices might be required, the model behavior needs to be studied, to avoid large performance drops. There are a number of works that investigate the application of memristors both through simulation and small-scale real-life experiments, e.g. on datasets such as MNIST [2, 3, 4, 5, 6, 7]. A recent publication offers a first study related to applying the technology for speech processing by simulating the execution of Conformer-based automatic speech recognition (ASR) models [8] on memristor hardware. Yet, some aspects relevant for state-of-the-art speech recognition were missing. In particular, relative positional encodings (PEs) in the self-attention, which play an important role in speech modeling, were not part of the model in their analysis. While it is known that PEs are crucial for good performance, we found that they are particularly relevant when operating in low-precision environments.

With the introduction of the Transformer architecture [9] PEs became relevant, because the self-attention mechanism lacks any positional information. In the initial work absolute PEs were added to the input to give positional context. An alternative was proposed in [10], where the fixed absolute PEs were replaced by relative ones that can be learned during training. This was also adopted in other works such as [11], as well as the Conformer paper [12], which is one of the state-of-the-art models for ASR. Previous works comparing absolute PEs to relative ones for ASR come to the conclusion that especially for long speech inputs, absolute positional information is not as helpful as the relative counterpart [13, 14].

When performing computation with low precision, the absolute value distribution of each layer of the model plays an important role. Looking into the PE, we found the output range to be different for encoding layers compared to regular layers. This is especially relevant for specialized memristor hardware design, where the quantization range is predetermined by the hardware itself [15]. When using such consistent quantization settings, unwanted value clipping might occur.

1.1. Contributions

In this work, we investigate the behavior of a CTC-based Conformer with relative PE, when being programmed and executed on simulated memristor hardware using SynaptogenML [16, 8]. The core contributions are:

- We confirm that relative PE improves the model performance. This improvement is stronger for low-precision computation, e.g. using 4-bit weights. However, we observe degradation when adding PE for memristor-based execution.
- We analyze the qualitative differences in the two networks, showing that the output value range of the PE is not well captured by the default memristor configuration.
- We restore the performance gain of PE using adapted model and device configurations. We improve by $\sim$ 15% relative over the case without PE, reducing the relative degradation during memristor device execution by $\sim$ 50%.

Furthermore, we discuss the implications of the different solutions we investigate in this work in the context of real world deployment scenarios. The information can help researchers from both the hardware and the modeling community, when aiming to improve energy-efficient computing through application of memristive hardware. All experiments can be run on a single GPU with 24 GB VRAM. SynaptogenML, training code, recipes and other software [17, 18, 19] are publicly available. [1]

[**] indicates the corresponding author.

[1] https://github.com/rwth-i6/returnn-experiments/tree/master/2026-memristor-pe

2. Background

2.1. Memristors

Memristors are physical circuit elements that change their resistance through the application of a sufficiently large voltage or an electric field [20]. The resistance states within the device are non-volatile, meaning they are held until the next programming cycle. The major benefit of memristors for machine learning is the capability to provide an analog interface for VMM directly in memory [21]. Through connecting multiple devices on crossbar arrays, each memristor can represent one or multiple bits of an entry in a weight matrix. The input tensor of the operation can be converted into a voltage through digital-to-analog conversion (DAC) and applied to such an array. The resulting currents represent the VMM output and are converted back into the digital domain using analog-to-digital conversion (ADC).

Unfortunately, the circuits for DAC and ADC may require a large amount of energy and chip area compared to the actual computation network. Adding them to nanometer-scale circuits still poses relevant challenges, and the precision range is usually limited to 8-bit in order to design realistic circuits [15]. This means that in addition to the uncertainty in the computation itself due to variation in the resistance states [22], the computation results are limited to a narrow digital precision. Due to the necessity of physically storing weights and a slow programming process [23], only network layers with a static weight matrix, such as linear or convolutional layers, are currently feasible.

Chips integrating memristor technology are not available at a scale required for executing larger neural networks yet, so most experiments are run on small models for datasets such as MNIST [24], CIFAR [25] or Google Speech Commands [26]. In order to overcome these limitations, we use a hardware simulation [8] integrated into PyTorch based on the physically accurate device model Synaptogen [16]. In its current state, the software allows to map a full Conformer network onto the simulated memristor devices. For training, mapping and inference we refer to the public setup and recipes described in our previous work [8].

2.2. Relative Positional Encodings

Relative PEs have made widespread use for Transformer- and Conformer-based models. While absolute encodings model the absolute position in the sequence, relative encodings model the relative distances between different positions. In this work we stay conceptually close to the relative PE as described in [11] and base our implementation on ESPNet[2]. This results in different possibilities in configuration, which we are also going to experiment with in this work.

The initial positional encoding matrix can either be based on a combination of sine and cosine functions [9, 11], or can be learned during training [10]. After this, the computed values are fed through a learned linear layer, which does not include a learnable bias. Instead, two different biases are added to the query, one for the multiplication with the key and one for the multiplication with the encodings. During memristor execution the linear layer, which transforms the positional encodings as input to attention, is mapped onto the simulated devices.

[2] https://github.com/espnet/espnet/blob/b74395b4fca4377e46cda7794e7a2e33bf644590/espnet2/asr_transducer/encoder/modules/attention.py

3. Experimental Setup

We make use of a Conformer [12] ASR system with Connectionist-Temporal-Classification (CTC)-based [27] outputs. We conduct the majority of our studies on LibriSpeech [28] and verify our findings on Loquacious as a second task using the 250 hour subset for training. For LibriSpeech, we use ARPA-phonemes from the provided lexicon as target labels. For Loquacious, we use byte-pair-encoding [29] with 128 merges. As the memristor simulation requires a substantial amount of computation, we limit our evaluations to the *dev-other* subset for LibriSpeech and *dev* for Loquacious.

In recognition, we use a 4-gram count-based language model (LM) with KenLM [30], which is either the official LibriSpeech LM or the Loquacious LM from [31]. Our models are being trained for 100 epochs. We use a linearly increasing and decreasing learning rate schedule peaking at $5 \cdot 10^{-4}$, with RAdam [32] as optimizer and a decoupled weight decay of $1 \cdot 10^{-2}$.

We train our model using log-mel features with 10ms frame shift and downsample the frames by a factor of 4, through a convolutional front-end. For regularization we use SpecAugment [33]. The Conformer consists of 12 layers, with model dimension 512 and feed-forward dimension 2048, with relative PE added to the self-attention computation. As described in Section 2.2, we use sinusoidal inputs transformed with a linear layer. The total number of parameters of the models is $\sim$ 77M.

We base our quantization-specific training settings and memristor mapping on the setup of our previous work in [8], where we give a detailed description of the configuration. We place observers before and after the matrix-based operations and enforce symmetric quantization. The observers are used to calculate the mapping to the memristive devices. Activations will be quantized with 8-bit precision, while weights are quantized either to 8 or 4-bit precision. In practice lower weight precision is desirable, as it reduces the overall number of required devices.

Following the newest version of the SynaptogenML[3] framework, we map all matrix operations with static weights of the Conformer model onto the simulated devices. For more details on mapping, programming and execution of the simulated devices we refer to [8]. Each matrix operation is subdivided into computations of 128×128. Within the sub-matrices, each bit level of the weight will be represented by a single crossbar array. Each of the crossbars has its own ADC, before bit shifting and accumulation occurs.

When configuring the ADC in the simulation, the number of bits for precision and range of the conversion needs to be specified. While the precision defines the decimal resolution, the range scales the absolute output value range, meaning the output is captured as a fixed-point value. For ADC conversion our baseline uses the settings considered reasonable in [8], namely 4 bits for precision and 4 bits for range. To keep the number of simulation runs feasible, we run 5 recognitions per model, each with differently programmed devices. We report the average and the standard deviation of the programming runs.

4. Experiments

4.1. Baseline

For our first comparison, we run the pipeline for models with and without PEs. The results can be seen in Table 1. For the

[3] https://github.com/rwth-i6/SynaptogenML

Table 1: *Results comparing baseline recognition and memristor execution, ADC is configured to 4-bit precision and 4-bit range. Activations are quantized to 8-bit.*

Weight Prec.	PE	WER [%]			
		Librispeech dev-other		Loquacious dev	
		Base.	Memristor	Base.	Memristor
8	No	5.7	**7.2** $\pm$ 0.07	13.1	**14.5** $\pm$ 0.05
	Yes	**5.4**	7.6 $\pm$ 0.08	**12.7**	16.4 $\pm$ 0.13
4	No	6.5	**7.5** $\pm$ 0.07	13.8	**15.4** $\pm$ 0.10
	Yes	**5.6**	**7.5** $\pm$ 0.10	**12.8**	15.9 $\pm$ 0.20

baseline recognition of the models without PEs we can see that the model suffers from performance degradation when quantizing the weights down to 4 bits. Mapping the model onto the simulated devices yields another performance degradation from 10 to 25 % relative WER, which is in a similar range as the one reported in [8]. The slight differences in degradation can be explained by the fact that the mapping now includes convolution and the model is about twice the size. For the models with PE, we can see that not only the 8-bit baseline outperforms the one without PE, but also the 4-bit models are much more stable to the quantization. This means that the positional information helps the models improve especially in the low-precision scenario. Yet, when executing the trained models on the simulated devices, we can see that the performance degradation is higher (25 to 40% relative). Also the absolute performance does not beat the models without PE. Given the previous results in [8], we expected that a performance increase by adding PE to the baseline model would also improve the memristor performance.

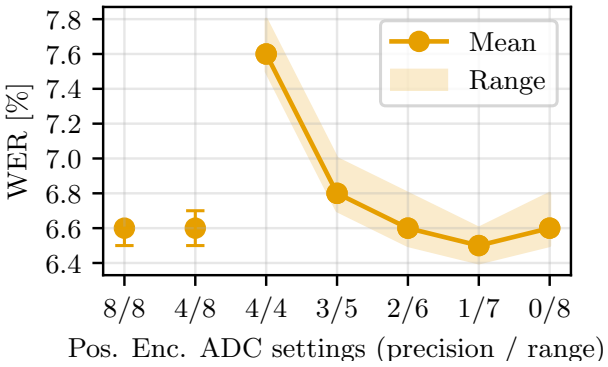
Table 2: *Comparison of different ADC configurations for the full model on Librispeech dev-other. Activations are quantized to 8-bit. Oracle refers to keeping the encoding computation in the digital domain, i.e. not mapping it to the memristor devices.*

Positional Encoding	ADC		WER [%]	
	Prec.	Range	8 - bit weights	4 - bit weights
Yes	4	4	7.6 $\pm$ 0.08	7.5 $\pm$ 0.10
		8	**6.6** $\pm$ 0.06	**6.8** $\pm$ 0.11
	8	8	7.0 $\pm$ 0.09	**6.9** $\pm$ 0.08
	3	5	7.3 $\pm$ 0.07	7.8 $\pm$ 0.04
	2	6	38.9 $\pm$ 1.11	57.8 $\pm$ 0.94
Oracle	4	4	**6.6** $\pm$ 0.09	**6.8** $\pm$ 0.06
No	4	4	7.2 $\pm$ 0.07	7.5 $\pm$ 0.07
		8	7.2 $\pm$ 0.06	7.5 $\pm$ 0.11
	8	8	8.3 $\pm$ 0.11	7.6 $\pm$ 0.08

4.2. ADC Variations

As the only difference between the two types of models lies in the addition of PEs, we will now investigate them in more detail. As mentioned in Section 3, the output value range of the default ADC conversion setting is bound by 4 bits. This is below the 8-bit value range that is theoretically reachable by the 128x128 VMM with binary weights. To verify that this is the issue, we track the ADC operation of the linear layer of the PE for each bit and device during a forward through the network. On average around 40 % of the time the result of the memristor computation is being clipped, with almost 100% for some crossbars in extreme cases. This causes a major shift in the output distribution of the PE. To get oracle results, we perform recognition runs for the models, where we do not map the linear layer of the PE onto the devices, but rather compute them digitally as usual. In Table 2, the oracle results show clear improvements

Figure 1: *Comparison of the different ADC settings for the PE of the 8-bit weight model. The ADC of other parts of the model is set to 4-bit range and 4-bit precision.*



over the baseline, almost halving the overall degradation caused by the memristor mapping. The ADC and DAC settings were not investigated by the authors in [8]. This means that it might also be an issue in other portions of the model. Because of this, we investigate multiple different configurations with respect to precision and range of the ADC.

For both 8 and 4-bit weight precision, increasing the range to 8-bit shows a clear improvement over the baseline setting. Interestingly, increasing the precision of the conversion improves the performance, but does not keep up with only increasing the range. A possible explanation for this is that a higher precision will enhance the memristor noise, while lower precision ADCs quantize some of it away. As this kind of ADC analysis is novel, we also run these experiments on the models without PE, visible in the last rows of Table 2. Not only does the increased range not help in this case, but the increased precision even degrades the model for 8-bit precision. From this we conclude that the other parts of the network do not suffer from the same range issues as the PE does.

In the next set of experiments we only change the settings of layers influencing the PE, but keep the rest of the network settings fixed at the default 4-bit range and precision. The first two values in Figure 1 show that in this case increasing the output range improves the model's performance, reaching the same optimum as an increase for the whole model. When also increasing the precision of the conversion, the model shows more robustness to the increased level of noise and does not degrade.

While studies with arbitrarily configured ADCs based on simulation are possible, in a real-life deployment setting, increasing the range or precision of the ADC might not be feasible. The increased bit levels will drastically increase the overall energy and area consumption of the ADC component. The result of this could be that the energy savings of the memristors might be lost in conversion. Because of this, we keep the overall bit budget fixed at 8 bits, and only shift bits from the precision towards the range. When applying the fixed budget to the full model in Table 2, we see minor improvements over the baseline for a precision/range split of 3/5. Below that, a strong degradation sets in. This means that the precision bits are required at least for some parts of the model. In contrast when only changing the conversion of the PE, we can see in Figure 1 a performance gain up to a range of 7 or 8 bits. Note that 0 precision bits imply that no fractional values are allowed, but a value range up to 128 integers is possible. While the best result is using 7 bits of range, we again assume that the difference between 6 to 8 bits in range reflects the uncertainty of the devices rather than an actual performance difference.

Table 3: *Final Results comparing the best approaches for degradation mitigation. Activations are quantized to 8-bit. The ADC of non-PE parts of the model is configured to 4-bit precision and 4-bit range. Oracle refers to keeping the encoding computation in the digital domain, i.e. not mapping it to the memristor devices.*

Positional Encoding	PE ADC		WER [%]							
			LibriSpeech dev-other				Loquacious dev			
	Prec.	Range	8 - bit weights		4 - bit weights		8 - bit weights		4 - bit weights	
			Base.	Memristor	Base.	Memristor	Base.	Memristor	Base.	Memristor
No	-	-	5.7	7.2 ± 0.07	6.5	7.5 ± 0.07	13.1	14.5 ± 0.05	13.8	15.4 ± 0.10
Yes	4	4	5.4	7.6 ± 0.08	5.6	7.5 ± 0.10	12.7	16.4 ± 0.13	12.8	15.9 ± 0.20
	4	8		6.6 ± 0.06		6.8 ± 0.06		14.2 ± 0.04		14.1 ± 0.07
	1	7		6.5 ± 0.06		6.8 ± 0.08		14.1 ± 0.06		14.1 ± 0.02
No Linear	-	-	5.6	6.9 ± 0.06	5.8	6.8 ± 0.07	12.9	14.6 ± 0.07	13.0	14.3 ± 0.06
Oracle	-	-	5.4	6.6 ± 0.09	5.6	6.8 ± 0.06	12.7	14.2 ± 0.08	12.8	14.1 ± 0.05

4.3. Pos Enc Variations

In physical circuits, dynamic changes to the ADC might be difficult to realize under desired space and power constraints. This is why we investigate whether we can improve the model itself in preparation for memristor deployment. We implement multiple changes to the PE, which can be seen in Table 4. One reason for the deviation of layer statistics in the PE transformation layer might be the artificial structure of sinusoidal encodings used as input. Because of this, we replace them by learned PE as initially proposed in [10].

The baseline performance, compared to the default configuration, is worse by 0.2% WER absolute. Yet, when mapping the model onto the devices, one can see that the degradation is slightly lower, especially for 4-bit weight precision. While this indicates that a part of the issue is caused by the sinusoidal encodings, it does not account for all of the degradation.

Secondly, we experiment with retraining the model without the linear transformation. This is different from the oracle case, where we just do not map the layer onto the devices. From the baseline result one can see that the performance of the model degrades by 0.2% WER absolute. Yet, as there is no linear to map to the devices, the overall performance during memristor execution surpasses the default configuration, only staying behind larger ADC ranges and the oracle performance. This means that in a setting where the ADC cannot be modified, this approach is a reasonable alternative. As the PE component in this case has no learnable parameters, we also try swapping the sinusoidal encodings for trainable ones. While the baseline performance stays about the same as for the sinusoidal encodings without a linear, the memristor execution is slightly worse. We conclude that the learnable encodings themselves are not able to compensate for the missing linear layer.

4.4. Final Results

The final results can be found in Table 3. For this we use the best performing configurations from the ablation studies in Section 4.2 and Section 4.3 and run them on Loquacious to verify our findings. While the overall error range of the corpus is higher, the models show a similar trend. Increasing the precision of the ADC helps both for the complete model, as well as only for the mapping of the PE. Shifting the precision bits into the range of the computation of the mapped encodings brings a similar improvement, while keeping the estimated energy consumption constant. One minor difference is the fact that the 4-bit model seems to handle the memristor execution slightly better than for LibriSpeech, as the performance is marginally better than the 8-bit case. A possible explanation for this might be the fact that the overall error rates are higher and the model already has to deal with increased uncertainty. As for LibriSpeech, dropping the linear mapping degrades the baseline model slightly, but produces a better model in the default memristor configuration. Interestingly, the memristor performance for the 8-bit weights does not surpass the model without PE, while for the lower bit precision the importance of the PE shows once again. Overall, we can conclude that our changes improve the memristor execution mostly independent of the dataset.

Table 4: *Comparison of modifications to the relative PE. ADC is configured to 4-bit precision and 4-bit range. Activations are quantized to 8-bit. Oracle refers to keeping the encoding computation in the digital domain, i.e. not mapping it to the memristor devices.*

Positional Encoding	WER [%]			
	8 - bit weights		4 - bit weights	
	Base.	Memristor	Base.	Memristor
No	5.7	7.2 ± 0.07	6.5	7.5 ± 0.07
Baseline	**5.4**	7.6 ± 0.08	**5.6**	7.5 ± 0.10
Oracle	**5.4**	**6.6** ± 0.09	**5.6**	**6.8** ± 0.06
Learnable	5.7	7.4 ± 0.14	5.8	7.0 ± 0.05
No Linear	5.6	**6.9** ± 0.06	5.8	**6.8** ± 0.07
+ Learn.	5.5	7.0 ± 0.10	5.8	7.0 ± 0.06

5. Conclusion

In this work, we investigated the issues that can arise when mapping a state-of-the-art Conformer model with relative PE onto simulated memristor devices. We identified the encodings to require specific care, as they do not integrate into the memristor execution as smoothly as the rest of the network. In order to mitigate the degradation caused by the increased noise and reduced output range of the memristor devices, we investigated a number of methods. Increasing the range bits of the ADC reduces the degradation by 50% relative, while changing the model architecture reduces it by 30%. Which method is applicable strongly depends on the use case and scenario. In a situation where the chip used for execution is still configurable in its ADC settings, or statically provides different configurations, a higher number of range bits restores the performance gain of the PE. Another possible scenario is where the chip configuration does not provide multiple levels and is static. There, removing the linear mapping from the model before training degrades the baseline performance slightly, but results in a better performance during memristor execution.

6. Acknowledgments

This work was partially supported by NeuroSys, which as part of the initiative "Clusters4Future" is funded by the Federal Ministry of Research, Technology and Space BMFTR (funding IDs 03ZU2106DA and 03ZU2106DD), and by the project RESCALE within the program *AI Lighthouse Projects for the Environment, Climate, Nature and Resources* funded by the Federal Ministry for the Environment, Nature Conservation, Nuclear Safety and Consumer Protection (BMUV), funding ID: 67KI32006A. The authors gratefully acknowledge the computing time provided to them at the NHR Center NHR4CES at RWTH Aachen University (project number p0023999). This is funded by the Federal Ministry of Education and Research, and the state governments participating on the basis of the resolutions of the GWK for national high performance computing at universities (`www.nhr-verein.de/unsere-partner`). We would like to thank Leon Brackmann and Stephan Menzel for their helpful input and discussions on the design and verification of the simulation framework.

7. Generative AI Use Disclosure

Generative AI was used in this work for proofreading and grammar correction, but was not used to generate content.

8. References

[1] D. J. Wouters, Y.-Y. Chen, A. Fantini, and N. Raghavan, "Reliability Aspects," in *Resistive Switching*. John Wiley & Sons, Ltd, 2016, ch. 21, pp. 597–622.

[2] W. Wan, R. Kubendran, C. Schaefer, S. B. Eryilmaz, W. Zhang, D. Wu, S. Deiss, P. Raina, H. Qian, B. Gao, S. Joshi, H. Wu, H.-S. P. Wong, and G. Cauwenberghs, "A compute-in-memory chip based on resistive random-access memory," *Nature*, vol. 608, no. 7923, pp. 504–512, Aug. 2022.

[3] Y. Huang, T. Ando, A. Sebastian, M.-F. Chang, J. J. Yang, and Q. Xia, "Memristor-based hardware accelerators for artificial intelligence," *Nature Reviews Electrical Engineering*, vol. 1, no. 5, pp. 286–299, May 2024.

[4] Y. He, C. Ma, Z. Jia, Y. Su, and J. Tang, "High-density integration for RRAM-based computing-in-memory," *Nature Reviews Electrical Engineering*, Jan. 2026.

[5] S. Jain, A. Sengupta, K. Roy, and A. Raghunathan, "RxNN: A Framework for Evaluating Deep Neural Networks on Resistive Crossbars," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 2, pp. 326–338, 2021.

[6] T.-H. Wen, J.-M. Hung, W.-H. Huang, C.-J. Jhang, Y.-C. Lo, H.-H. Hsu, Z.-E. Ke, Y.-C. Chen, Y.-H. Chin, C.-I. Su, W.-S. Khwa, C.-C. Lo, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, M.-S. Ho, C.-C. Chou, Y.-D. Chih, T.-Y. J. Chang, and M.-F. Chang, "Fusion of memristor and digital compute-in-memory processing for energy-efficient edge computing," *Science*, vol. 384, no. 6693, pp. 325–332, 2024.

[7] J. Souto, G. Botella, D. García, R. Murillo, and A. del Barrio, "Neuromorphic Circuit Simulation with Memristors: Design and Evaluation Using MemTorch for MNIST and CIFAR," *arXiv preprint arXiv:2407.13410*, 2024.

[8] N. Rossenbach, B. Hilmes, L. Brackmann, M. Gunz, and R. Schlüter, "Running Conventional Automatic Speech Recognition on Memristor Hardware: A Simulated Approach," in *Interspeech*, 2025, pp. 2560–2564.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is All you Need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[10] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-Attention with Relative Position Representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, 2018, pp. 464–468.

[11] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. Le, and R. Salakhutdinov, "Transformer-XL: Attentive Language Models beyond a Fixed-Length Context," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, A. Korhonen, D. Traum, and L. Màrquez, Eds. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 2978–2988.

[12] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, and R. Pang, "Conformer: Convolution-augmented Transformer for Speech Recognition," in *Interspeech*, 2020, pp. 5036–5040.

[13] F. Yue and T. Ko, "An Investigation of Positional Encoding in Transformer-based End-to-end Speech Recognition," in *12th International Symposium on Chinese Spoken Language Processing (ISCSLP)*, 2021, pp. 1–5.

[14] L. Lu, "A Transformer with Interleaved Self-attention and Convolution for Hybrid Acoustic Models," *arXiv preprint arXiv:1910.10352*, 2019.

[15] N. Laflamme-Mayer, G. Kowarzyk, Y. Blaquière, Y. Savaria, and M. Sawan, "A Defect-Tolerant Reusable Network of DACs for Wafer-Scale Integration," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 2, pp. 304–315, 2019.

[16] T. Hennen, L. Brackmann, T. Ziegler, S. Siegel, S. Menzel, R. Waser, D. J. Wouters, and D. Bedau, "Synaptogen: A Cross-Domain Generative Device Model for Large-Scale Neuromorphic Circuit Design," *IEEE Transactions on Electron Devices*, vol. 71, no. 9, pp. 5345–5353, 2024.

[17] J. Peter, E. Beck, and H. Ney, "Sisyphus, a workflow manager designed for machine translation and automatic speech recognition," in *EMNLP 2018: System Demonstrations, Brussels, Belgium, October 31 - November 4, 2018*, pp. 84–89.

[18] P. Doetsch, A. Zeyer, P. Voigtlaender, I. Kulikov, R. Schlüter, and H. Ney, "Returnn: The RWTH extensible training framework for universal recurrent neural networks," in *ICASSP 2017, New Orleans, LA, USA, March 5-9, 2017*, pp. 5345–5349.

[19] S. Wiesler, A. Richard, P. Golik, R. Schlüter, and H. Ney, "RASR/NN: The RWTH neural network toolkit for speech recognition," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, May 2014.

[20] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, no. 7191, pp. 80–83, May 2008.

[21] C. Li, M. Hu, H. Li, Yunningand Jiang, N. Ge, E. Montgomery, J. Zhang, W. Song, N. Dávila, C. E. Graves, Z. Li, J. P. Strachan, P. Lin, Z. Wang, M. Barnell, Q. Wu, R. S. Williams, J. J. Yang, and Q. Xia, "Analogue signal and image processing with large memristor crossbars," *Nature Electronics*, vol. 1, no. 1, pp. 52–59, 2018.

[22] M. R. Mahmoodi, A. F. Vincent, H. Nili, and D. B. Strukov, "Intrinsic Bounds for Computing Precision in Memristor-Based Vector-by-Matrix Multipliers," *IEEE Transactions on Nanotechnology*, vol. 19, pp. 429–435, 2020.

[23] A. Grossi, E. Vianello, C. Zambelli, P. Royer, J.-P. Noel, B. Giraud, L. Perniola, P. Olivo, and E. Nowak, "Experimental Investigation of 4-kb RRAM Arrays Programming Conditions Suitable for TCAM," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 12, pp. 2599–2607, 2018.

[24] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010.

[25] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," Tech. Rep., 2009.

[26] P. Warden, "Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition," *arXiv preprint arXiv:1804.03209*, Apr. 2018.

[27] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks," in *Proceedings of the 23rd International Conference on Machine Learning*, ser. ICML '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 369–376.

[28] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR Corpus Based on Public Domain Audio Books," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 5206–5210.

[29] R. Sennrich, B. Haddow, and A. Birch, "Neural Machine Translation of Rare Words with Subword Units," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Aug. 2016, pp. 1715–1725.

[30] K. Heafield, "KenLM: Faster and Smaller Language Model Queries," in *Proceedings of the Sixth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, Jul. 2011, pp. 187–197.

[31] N. Rossenbach, R. Schmitt, T. Raissi, S. Berger, L. Kleppel, and R. Schlüter, "Supplementary Resources and Analysis for Automatic Speech Recognition Systems Trained on the Loquacious Dataset," *arXiv preprint arXiv:2512.17915*, 2025.

[32] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, "On the Variance of the Adaptive Learning Rate and Beyond," in *Proceedings of the Eighth International Conference on Learning Representations (ICLR)*, Apr. 2020.

[33] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition," in *Interspeech*, 2019, pp. 2613–2617.