On Analog-to-Digital Conversion Configurations for Memristor-based Automatic Speech Recognition

Benedikt Hilmes[1, 2], Nick Rossenbach[1, 2], Leon Brackmann[3], Ralf Schlüter[1, 2]

[1] Machine Learning and Human Language Technology Group, RWTH Aachen University,

[2] Apptek GmbH, [3] Institute for Electronic Materials II, RWTH Aachen University

Modern neural networks achieve performance boosts by scaling the matrices for vector-matrix-multiplication (VMM) to larger dimensions. These weights are trained beforehand and stay fixed during inference. Non-volatile memristors can be used to physically save the static weights in analog hardware directly within a memory array[1]. By applying a voltage corresponding to the dynamic input of the VMM, the resulting current can be interpreted as multiplication result. As not all parts of a modern neural network can be calculated on device, digital-to-analog converter (DAC) and analog-to-digital converter (ADC) connect the analog and digital domains.

Current memristors are subject to major stochastic behavior, which can occur both during programming and execution. As the device availability is still limited, most studies rely on simulations in order to study the device behavior. Still, not many studies investigate the influences of the devices and different steps of the computation pipeline for large scale neural networks.

In this work we focus on the impact of ADCs on the memristor-based VMM execution pipeline, which we find to cause a bottleneck in the neural model for automatic speech recognition (ASR). We make use of the simulation framework SynaptogenML[1], which is based on the hardware-trained device model Synaptogen [2]. For the ADC we assume a configurable number of precision bits, defining decimal resolution, as well as a number of range bits, which scale the output value range. In the base configuration we use 4 bits for both precision and range as proposed in [3]. With this, we run our experiments on the Loquacious dataset [4], using a Conformer-based CTC model [5], one of the state-of-the-art models for ASR.

For the baseline, the positional encodings in the network improve the word-error-rate (WER) from 13.8% to 12.8%. Yet, the model without positional encodings performs better on the simulated devices than the model with positional encodings (15.4% vs. 15.9%). We propose that this difference in performance is caused by the ADC, which can be shown by capturing the value range of the positional encodings, both during baseline and memristor inference. On average, 40% of the positional encoding values are being clipped by the ADC. We run a number of studies where we adjust the configuration of the ADC and look for configurations which restore the baseline improvements.

When increasing the range bits both over the whole network as well as only for the devices involved in the computation of the positional encodings, the performance can be improved from 15.9 % to 14.1% WER. On the other hand, increasing the precision of the conversion does not yield performance improvements. We assume that increasing the precision enhances the memristor noise, which might otherwise be quantized away during conversion. As the number of used bits in a real world scenario highly increases the amount of energy required, we also study the behavior when only shifting bits from the precision to the range. Here we see that the performance significantly degrades below 3 bit precision and conclude that the overall model requires this precision. On the other hand, applying this shift only to the ADC for the positional encodings improves the performance on a similar level, without increase in projected energy consumption.

In total, in this work we present an analysis on different ADC configurations and their influence on the performance of executing a state-of-the-art ASR model on memristive devices. By adjusting the range and precision bits of the ADC used for the computation of the positional encodings we reduce the performance degradation caused by the devices by $\sim 40\%$ relative.

[1] D. B. Strukov et. al., Nature, 453, 80–83, 2008
[2] T. Hennen et al., IEEE Trans. Electron Devices, 71, 5345–5353, 2024
[3] N. Rossenbach et. al., Interspeech, 2560-2564, 2025
[4] T. Parcollet et. al., Interspeech, 4053-4057, 2025
[5] A. Gulati, et al., Interspeech, 5036-5040, 2020

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