Hardware-Accelerated Liquid Neural Network on an Adaptive System-on-a-Chip

Student

Yanik Kuster

Introduction: A recent paper by Ramin Hasani et al. introduced a new type of continuous-time recurrent neural network (CT RNN) called Liquid Time-Constant Recurrent Neural Network (LTC RNN). A CT RNN defines its states as a system of first order ordinary differential equations (ODEs) whose dynamics is governed by the time constant. An LTC RNN goes further by allowing the time constants to be dependent on the current state (Fig. 3). Thus, an LTC RNN can change its dynamics depending on the state of the system, which results in more powerful neurons. According to Ramin Hasani, LTC RNNs have improved performances in time-series prediction even if irregularly sampled. Due to the promising properties of LTC RNNs, it is of our interest to bring these networks to the edge device.

Approach: The project's goal is to implement an LTC RNN on an adaptive system-on-a-chip (ASoC), namely the Kria KV260 platform (Fig. 1). Solving the system of ODEs of an LTC RNN is computationally demanding. Ramin Hasani derived an approximation for the closed-form solution with which a so-called closed-form continuous-depth network (CfC) was designed (Fig. 2 and 3). The CfC has been chosen for the implementation since it omits solving the system of ODEs. For comparison, two different system implementations were realized: One using only a single core of the Cortex-A53 processing system @1.333 GHz and the other making use of the FPGA fabric @214 MHz.

Conclusion: The CPU implementation uses floating point numbers, which takes 3.49 ms to compute. The FPGA implementation uses a custom quantization based on power-of-two scale quantization with which the complete network is computed in 22.5 us. This means that a speedup in the order of 155 has been achieved. The cost of this speedup shows in the degrading mean squared error (MSE), which increased to 1.58 from the baseline of 0.659 given by the TensorFlow model. There are multiple ways to reduce the MSE, e.g., by increasing the model resolution or by computing parts of the model on the CPU instead of the FPGA. Additionally pruning the model and exploiting the resulting sparse representation could improve the speedup further.

Fig. 1. The Kria KV260 platform is connected to an oscilloscope to measure the execution times.
Own presentment

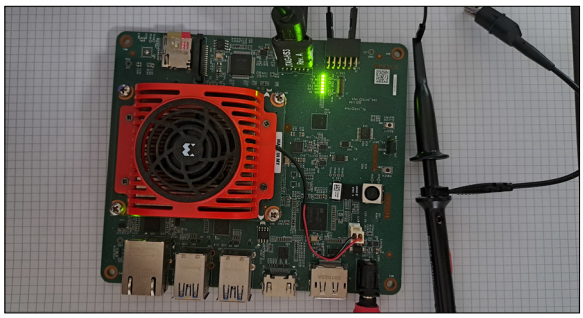


Fig. 2. LTC RNN's output compared to its CfC version. The CfC has been generalised to the equation in Fig. 3.
Source: https://www.nature.com/articles/s42256-022-00556-7

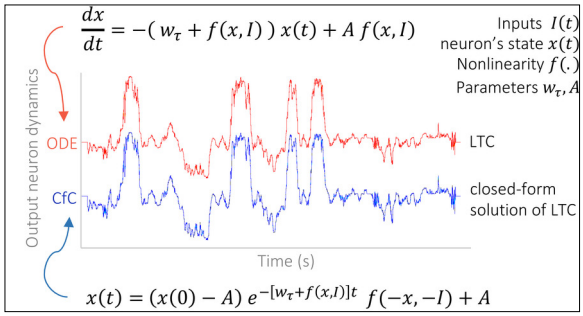


Fig. 3. The equations specify the dynamics of the RNN's states. The block diagram shows the network structure of a CfC.
Based on: https://www.nature.com/articles/s42256-022-00556-7

CT RNN

$$\frac{dx(t)}{dt} = -\frac{x(t)}{\tau} + f(x(t), I(t), \theta)$$

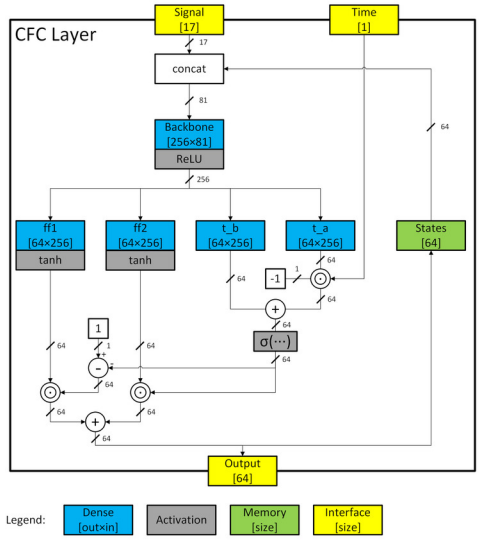
LTC RNN

$$\frac{dx(t)}{dt} = -\left(\frac{1}{\tau} + f(x(t), I(t), t, \theta)\right) x(t) + f(x(t), I(t), t, \theta) A$$

CfC RNN

$$x(t) = B \odot e^{-\left(\frac{1}{\tau} + f(x,I,\theta)\right)t} \odot f(-x, -I, \theta) + A$$

States: x
Inputs: I
Time: t
Neural Network: $f(\ldots)$
Network Parameters: Θ
Parameter Vector: A, B
Time Constant: τ
Liquid Time Constant: $\frac{1}{\frac{1}{\tau} + f(x,I,\theta)}$



Advisor
Prof. Dr. Andreas Breitenmoser

Subject Area
Data Science, Electrical Engineering