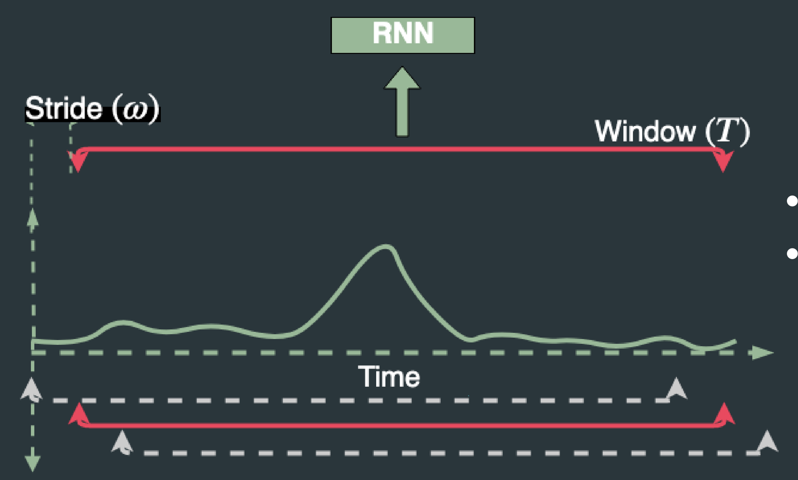
Carnegie Mellon University · Microsoft Research · TEXAS The University of Texas at Austin · BOSTON UNIVERSITY

Shallow RNN: Accurate Time-series Classification on Tiny Devices

Don Kurian Dennis Dumrus Alp Emre Acar Vikram Mandikal Vinu Sankar Sadasivan Harsha Vardhan Simhadri Venkatesh Saligrama Prateek Jain

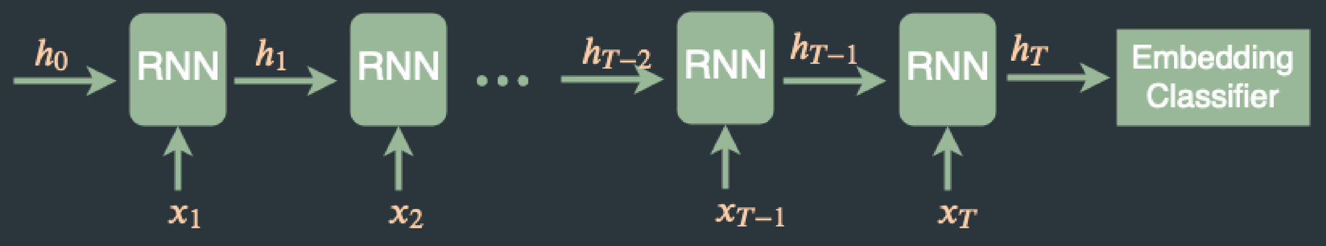
github.com/Microsoft/EdgeML

Classification for Streaming Data



- State-of-the-art for time series.
- Data is divided into overlapping windows and an RNN is run over each window.

- Involves sequential evaluation of a state update rule.
- The state update rule: complicated, non-linear and expensive.



$$\mathbf{h_t} = \sigma(\mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1} + \mathbf{b})$$

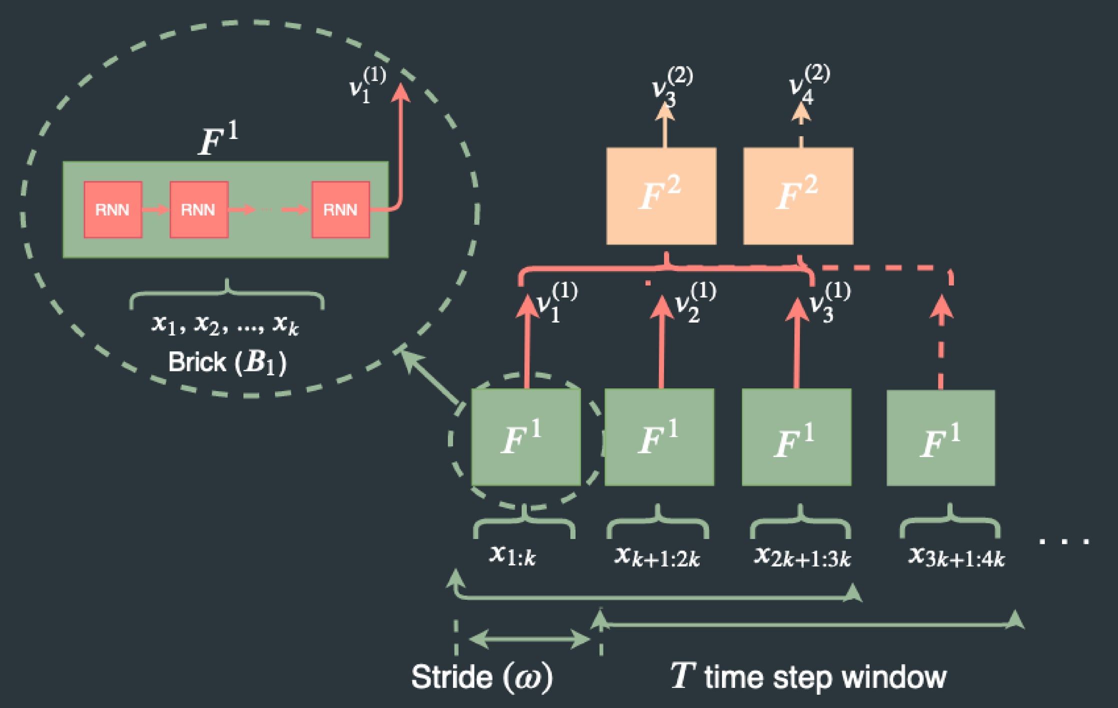
- Evaluation of each window restarts at h_0
  $O(T/\omega)$ computation per prediction.
- No computation 'reuse', even with high overlap in successive windows.
- For single threaded devices, low memory, evaluation should complete in $O(\omega)$ to prevent data loss.
- **Prohibitively expensive for tiny devices!**

Can we perform subsequent evaluations in O(1)?

Contributions

- ShaRNN: Shallow, parallelizable RNN variant.
  USP: a) higher accuracy than vanilla RNN and low resource RNNs.
  b) parallelizable + compute reuse at same memory footprint.
  c) allows deployment on tiny devices MXChip, cortex M4.
  d) general technique; example online LAS.
- Analysis: theoretical justification for ShaRNN. Comparisons with other truncated RNN works.
- Demo: dkdennis.xyz/static/sharnn-neurips19-demo.mp4
  Code: github.com/Microsoft/EdgeML

ShaRNN: Shallow Recurrent Neural Networks



- Divide each window into 'bricks' B_i of length k
- Simplest version – two layers;
  - Lower layer RNN (F^1) evaluates each brick and buffers them; computes intermediate state $v_i^{(1)} \in R^d$
    $$v_i^{(1)} = F^1(x_{ik+1:(i+1)k})$$
  - Second layer RNN (F^2) evaluates all $v_i^{(1)}$ corresponding to the current window.
    $$v_i^{(2)} = F^2(v^{(1)}_{(i-1)k+1:(i-1)k+k})$$
  - Each brick evaluated only once, hence computation reuse
- Increased sophistication
  - ✚ more layers, exploit cheaper cells (FastRNN, UGRNN)
  - ✚ multiple instance framework (MI-RNN)

Theoretical Insights

Claim [Streaming, 2-layer ShaRNN]: Amortized cost for processing each window is:

$$\text{cost} = O(T/(\omega k) + k/\omega)$$

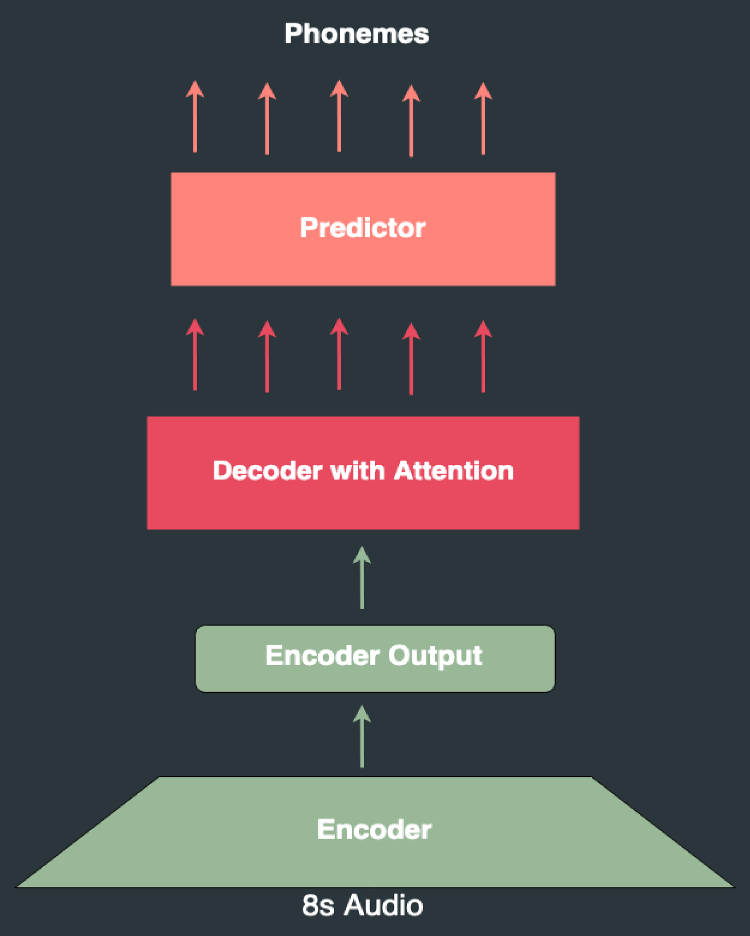
where T is the number of time-steps in a window, ω, is the stride and k is the brick length. Thus if we choose, $k = \sqrt{T}$ and if $\omega = \sqrt{T}$, then:

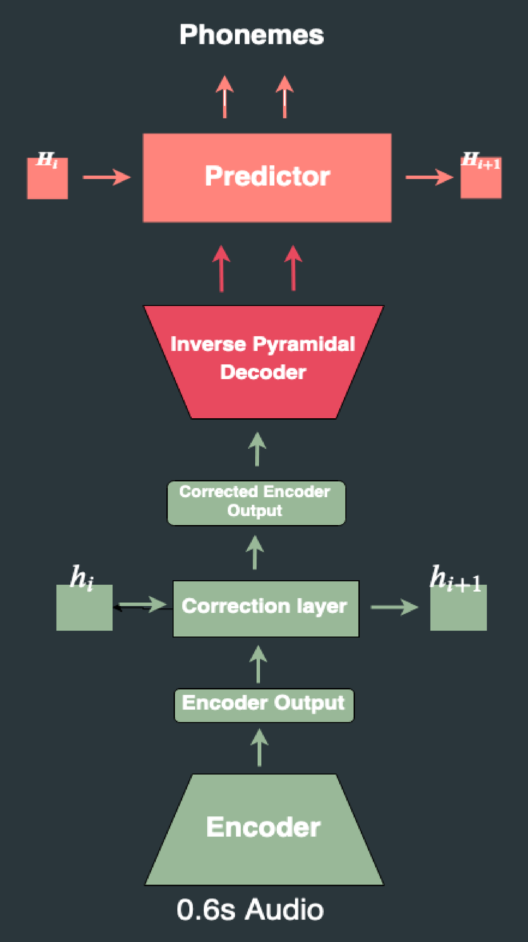
$$\text{cost} = O(1)$$

Claim [Approximation Error]: ShaRNN well approximates fully recurrent RNN (F) if its *higher order* derivatives are bonded. That is,

$$||F(h_0, x_{1:T}) - ShaRNN(h_0, x_{1:T})|| \leq \epsilon \text{MT},$$

if the M-th order derivative bounded in its norm as $O(\epsilon M!)$ for some small $\epsilon > 0$.





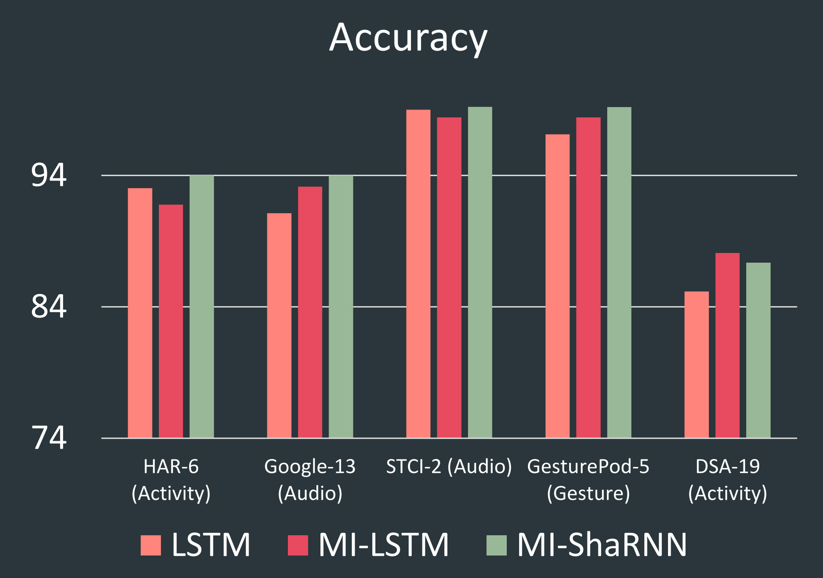
Online LAS

Benefits of ShaRNN can be reaped in more general settings. Example : Online Listen Attend Spell.

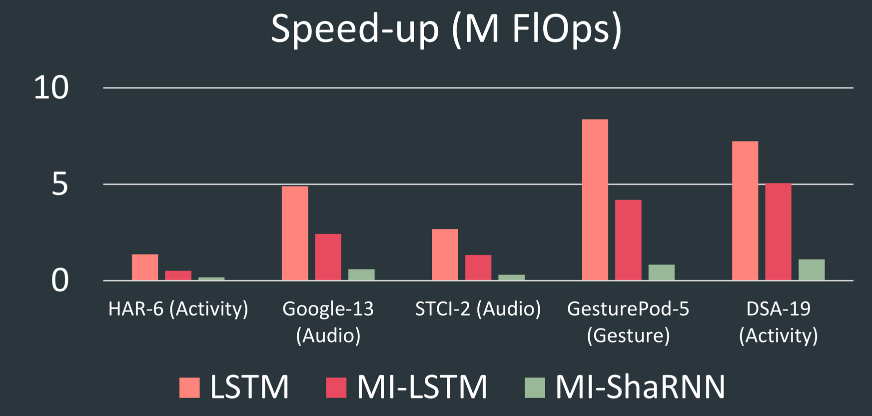
- Standard LAS for static input; transcribes 8 second audio.
- Introduce ShaRNN, break recurrence; ShaRNN encoder and decoder along with correction layers - **no attention layer.**
- Now response within 1 second.

Phoneme error rate improvement from 0.251 to 0.240 (on TIMIT dataset)

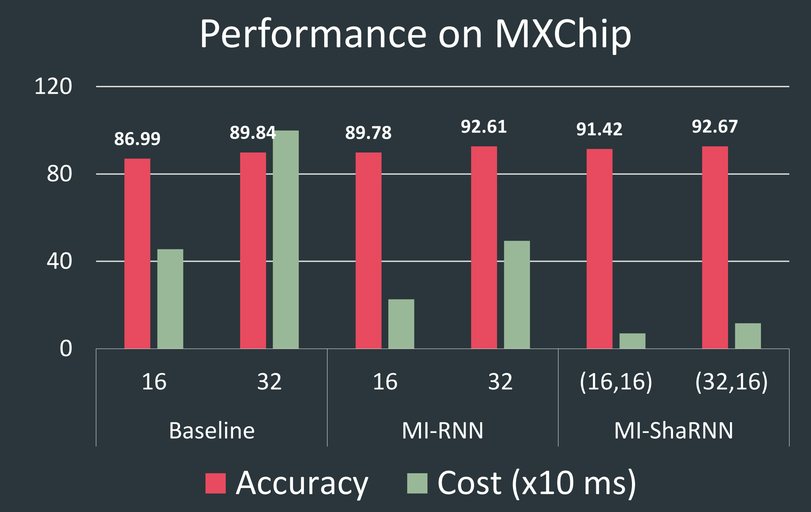
Experimental Results



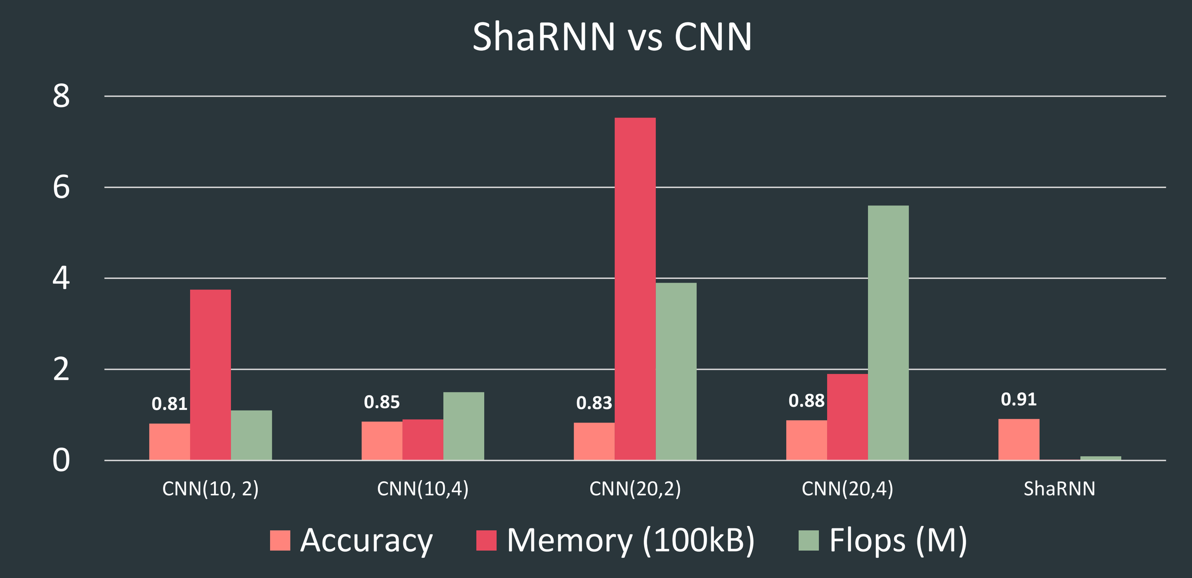
Accuracy of ShaRNN compared to MI-RNN and a fully recurrent LSTM. MI-LSTM is current state of the art for streaming classification. LSTM cells used everywhere.



Million floating point operations required for evaluation of one new window in the streaming setting.



Accuracy vs prediction cost on an MXChip (cortex M4). Prediction budget is 100 ms. Hidden dimensions in parenthesis.



Computation and memory comparison to various CNN models. Number of layers and filter size indicated in parenthesis.