We thank the reviewers for their careful reading of our manuscript and their many insightful comments and suggestions towards improving our paper. Below we provide a single response to all the comments of the reviewers, which will be added to the paper.

FSM-based networks: FSM-based networks are constructed using weighted linear FSMs (WLFSMs) and perform their computations on stochastic bit streams. The main computational core of FSM-based networks involves additions and indexing operations. In such networks, the non-linear combinations of given data are learned by WLFSMs which are more expressive than linear transformations.

Why are WLFSMs more expressive than linear transformations: The computing power of linear FSMs comes from a simple fact in stochastic automata [26]: if elements to a stochastic automaton are from a Bernoulli sequence, the state sequences are then Markov chains of the first order while the output sequences are Markov chains of generally of higher orders. In stochastic computing (SC), this fact is interpreted as FSMs whose state transitions are linear, inputs are from a Bernoulli sequence, and outputs are sequences generally of higher orders.

Why using binary-stream representations: The choice of using binary-stream representations in SC-based systems has been made to accommodate hardware implementations. More precisely, performing computations on bit streams requires simple bit-wise operations, resulting in a better hardware performance (e.g., power consumption) compared to fixed-point/floating-point computations.

Unipolar format vs bipolar format: In FSM-based networks, all the computations are performed on random bit streams where each zero represents either -1 or 0 depending on the range of the real value that is being represented by stochastic bit streams (i.e., unipolar format vs bipolar format). For example, the binary sequence of "0 1 0 0" is a stochastic representation of length 4 for both real values of $x = 1/4$ and $y = -2/4$ where $x = (y + 1)/2$. Please note that the conversion (i.e., $x = (y + 1)/2$) allows to follow the flow of information in floating-point format while the computations in SC are still performed on bit streams of 0 and 1.

Computations of FSM-based networks: To visualize the computations of FSM-based networks, we have illustrated the inference computations of a neuron with two inputs as an example in Figure A where $B(*)$ denotes a Bernoulli function. In this example, the inputs (i.e., $4/8$ and $6/8$) are represented using unipolar format whereas the output (i.e., $-2/8$) is in bipolar format. Each input individually is passed to a 4-state to obtain the state values of **s** which are the indices to the weight matrix W. Once the weight elements corresponding to their state values are selected, a Bernoulli sample of them is passed to a scaled adder. The scaled adder performs element-wise additions and scales the result of additions by a factor of $1/2$ (since there is two inputs). The result of the scaled adder is then sampled to obtain a binary stochastic stream which could be an input to the next layer.

Inference latency of FSM-based networks: In FSM-based network, the length of stochastic streams (i.e., l) denotes the latency of the inference computations.

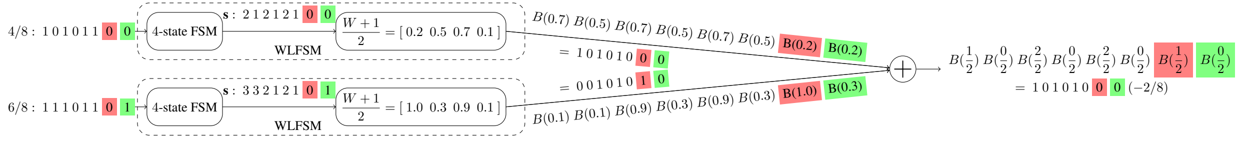
4/8 : 1 0 1 0 1 1 0 0
4-state FSM
s : 2 1 2 1 2 1 0 0
$\frac{W+1}{2} = [0.2\ 0.5\ 0.7\ 0.1]$
WLFSM
$B(0.7)\ B(0.5)\ B(0.7)\ B(0.5)\ B(0.7)\ B(0.5)$ B(0.2) B(0.2)
= 1 0 1 0 1 0 0 0
6/8 : 1 1 1 0 1 1 0 1
4-state FSM
s : 3 3 2 1 2 1 0 1
$\frac{W+1}{2} = [1.0\ 0.3\ 0.9\ 0.1]$
WLFSM
= 0 0 1 0 1 0 1 0
$B(0.1)\ B(0.1)\ B(0.9)\ B(0.3)\ B(0.9)\ B(0.3)$ B(1.0) B(0.3)
$B(\frac{1}{2})\ B(\frac{0}{2})\ B(\frac{2}{2})\ B(\frac{0}{2})\ B(\frac{2}{2})\ B(\frac{0}{2})\ B(\frac{1}{2})\ B(\frac{0}{2})$
= 1 0 1 0 1 0 0 0 (−2/8)

Figure A: An example illustrating the computations of a two-input neuron in FSM-based networks.

FSM-based Model for temporal tasks: Similar to FSM-based networks, FSM-based models for temporal tasks are constructed by instantiating several WLFSMs in parallel along with two fully-connected layers serving as a transition function and an output generator. In FSM-based models, WLFSMs are treated as a memory to hold the past information of the data sequence. Unlike FSM-based networks that perform their computations on stochastic bit streams, the computations of FSM-based models for temporal tasks are performed on real-valued numbers.

Why not backpropagating through time: Since state transitions in FSM-based models occur based on the present state of FSMs and the present inputs only, the training process (i.e., backpropagation) of such models is decoupled from time. More specifically, FSM-based models encode the input sequence into a distinct set of state values based on the present entry of the input sequence and the present value of states. The encoding capacity of FSM-based models is determined by the number of FSMs and their number of states. To better understand the computational process of such models, let us provide an example for a character-level language modeling task on the sequence of "b c a" (see Figure B) when considering a dictionary containing the three characters of "a", "b" and "c" only. For the given example, the state values are initialized with zeros at the beginning. Once the first character (i.e., "a") in a form of one-hot encoding is received, it is passed to an embedding layer (i.e., a fully-connected layer which is also referred to as transition function) to provide a dense representation of characters similar to LSTMs. Since WLFSMs only take binary values of 1 and 0 which respectively increment and decrement the state values by 1, a sample of embedded values is passed to 3-state FSMs. Once the state values are determined, their corresponding weights are selected and added with the scaling factor (i.e., α) of $1/2$. The results of FSM-based layer are then classified using a fully-connected layer to predict the next character. For the given example, the state value (i.e., **s**) of "0 1" denotes the character "a" whereas the state value (i.e., **s**) of "1 2" denotes the sequence of "c a". In fact, the FSM-based model adjusts the weights of the transition function (i.e., W_x) such that FSMs generate a unique set of state values based on the present input and the present state values.

Why is epoch-to-convergence affected less in FSM-based Models: Since the training process is decoupled from time and the backpropagation is performed based on the current time step, the gradients are not compromised by the length of input sequences and consequently epoch-to-convergence is affected less for long input sequences compared to LSTMs.

Comparison with other networks: It is worth mentioning that accuracy performance and hardware performance of GRUs are similar to those of LSTMs based on our simulations. However, accuracy performance of vanilla RNNs are considerably worse especially when considering their epoch-to-convergence for long input sequences. We will include the simulation results of GRUs and vanilla RNNs in both Figure 4 and Table 2 of the paper. Moreover, all the typos raised by reviewers will be fixed and missing literature and explanations will be added to the final version of the paper.

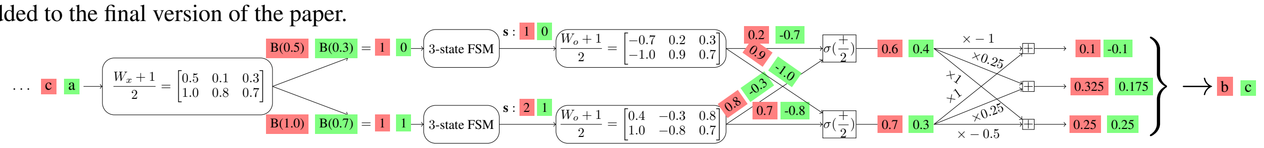
... c a
$\frac{W_x+1}{2} = \begin{bmatrix}0.5 & 0.1 & 0.3\\1.0 & 0.8 & 0.7\end{bmatrix}$
B(0.5) B(0.3) = 1 0
B(1.0) B(0.7) = 1 1
3-state FSM
s : 1 0
s : 2 1
$\frac{W_o+1}{2} = \begin{bmatrix}-0.7 & 0.2 & 0.3\\-1.0 & 0.9 & 0.7\end{bmatrix}$
$\frac{W_o+1}{2} = \begin{bmatrix}0.4 & -0.3 & 0.8\\1.0 & -0.8 & 0.7\end{bmatrix}$
0.2 -0.7
0.9 -1.0
0.8 -0.3
0.7 -0.8
$\sigma(\frac{+}{2})$
0.6 0.4
0.7 0.3
× −1
×0.25
×1
×1
×0.25
× −0.5
0.1 -0.1
0.325 0.175
0.25 0.25
b c

Figure B: An example of FSM-based model performing a character-level language modeling.