Exploring Spike Encoder Designs for Near-Sensor Edge Computing

JINGANG JIN, ZHENHANG ZHANG, QINRU QIU
SYRACUSE UNIVERSITY, USA

Outline

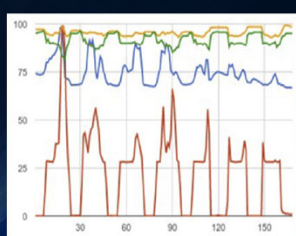
- Introduction and Motivation
- Proposed Method
- Experiments
- Conclusions

Neuromorphic Computing

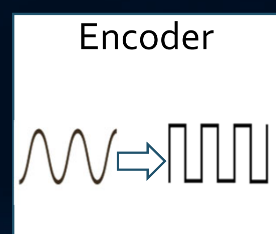
- Cyber-physical-systems and embodied AI call for edge capabilities
  - Continuous sensing, monitoring, detecting and responding to temporal and spatial patterns in physical environment.
  - Low cost, small footprint, high energy efficiency.
  - Flexibility and adaptivity to users and application context.
- Neuromorphic computing offers a promising solution by leveraging spiking neural networks (SNNs)
  - Information is represented through sequences of sparse spiking activities
  - Neurons communicate and compute only when there are output or input activities.
  - Event-driven approach, combined with closely integrated memory and computation, results in reduced workload and improved energy efficiency.

Spike Encoding

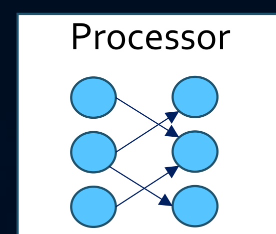
- To effectively utilize SNNs, input signals must be encoded into spike sequence
  - Sparsely project input sequences into hyperdimensional vectors distributed evenly across the feature space for better separation
- Well-designed spike encoder should
  - Convert numerical sensor readings into sparse spike trains
  - Preserve the temporal and spatial features in the input sequence
  - Be built with low-cost hardware, simple operations, and minimum memory



Multi-channel Sensor Inputs



Frontend



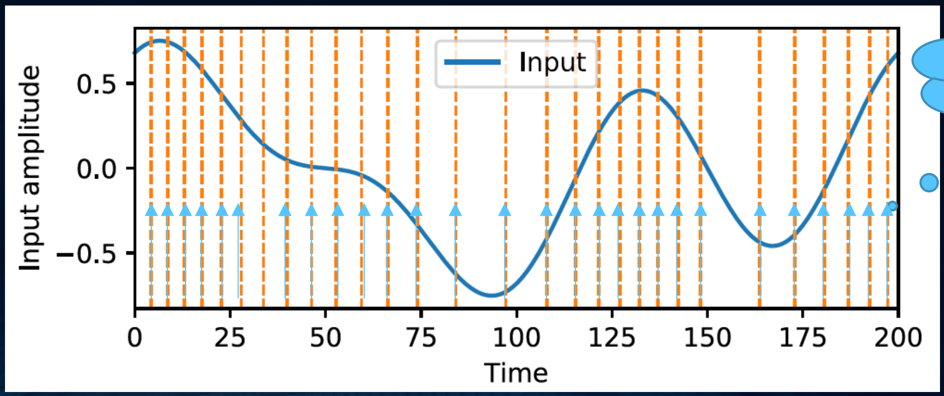
Backend

Conventional Spike Encoding

- Spike Rate Coding
  - Encoders periodically sample the sensor readings and convert them to floating point numbers in the digital domain (ADC)
  - The numerical values are represented by the spike count generated within the sampling interval
  - The maximum spike count within a sampling interval defines the data precision
  - Cons: High spiking activity and an increased computation workload
- Spike Temporal Coding
  - Input value is encoded as the interval between consecutive spikes (Spike Interval Coding)
  - Input value is encoded as the time delay between a spike and a reference start time (Time to First Spike Coding)
  - At most one spike event is generated for each sampling interval
  - Sparse communication and computation activities

Temporal Coding Using LIF neurons

- Current-based (CUBA) Leaky Integrate-and-Fire (LIF) neurons can be used for spike interval coding
  - $\tau \frac{dv(t)}{dt} = -v(t) + \boldsymbol{w} * \boldsymbol{x}(\boldsymbol{t}) + b$
  - $O(t) = H(v(t) - v_{th}) = \begin{cases} 1, & if\ v(t) > v_{th} \\ 0, & otherwise \end{cases}$, $H(\,)$ is the Heaviside activation function.



Limitations of Traditional Approaches

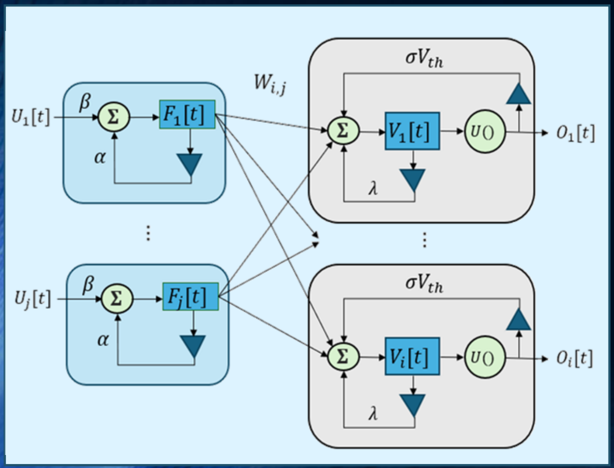
- Mapping each channel of sensor reading to one spike train
  - The one-to-one mapping cannot effectively preserve the spatiotemporal features of input sequences
  - Not every sensor's output will be in the sensitivity range of the LIF neuron
  - Subtle differences between sequences need to be accumulated for a long time before they can affect the spike output
- Use the spike encoder to collect training dataset for offline training
  - Difficult to guarantee that the training dataset provides an unbiased and comprehensive representation of the testing data distribution
  - If the encoder is implemented using the analog or mixed signal circuit design, significant device-to-device variations and limited hardware precision are typically expected
  - The frontend sensor and backend processor often operate asynchronously, relying on independent local clocks, no predetermined timing relationship can be assumed

Our Approaches to Address the Limitations

- New spike encoder designs
  - Population coding-based encoder (PopEnc): a layer of LIF neurons fully connected to the sensor inputs
    - Compared optimized and random PopEnc
  - Reservoir encoder (ResEnc): inspired by Reservoir Computing models using a recurrent network with randomly generated weight coefficients
    - With and without skip connections which directly connects the input to the output port
- Training on the edge, directly in conjunction with the encoder
  - Utilize online learning algorithm to train the backend classifier alongside the encoder to address encoders' variability

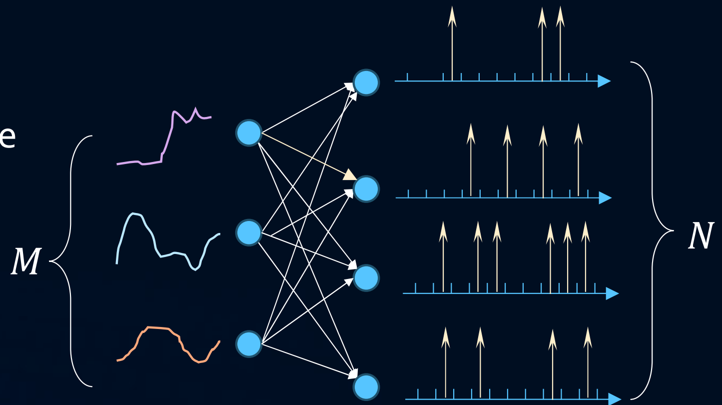
Population Encoder (PopEnc)

- Neuron Model: LIF neuron with post-synaptic potential (PSP)
  - Model synaptic dynamics as a first order IIR (Infinite Impulse Response) filter:
    - $F_i[t] = \alpha F_i[t-1] + \beta U_i[t]$, $U_i[t]$ -- ith input of the neuron
  - Membrane potential is the leaky accumulation of PSP with weight w:
    - $V_j[t] = \lambda V_j[t-1] + \sum_i w_{i,j} F_i[t] - \zeta V_{th} O_j[t-1]$, ζ – reset strength, empirically set to 0.5
- Population coding: using N neurons to jointly encode M inputs



Optimized PopEnc
- Optimized $w_{i,j}$ and α, β
- Better theoretical performance
- High implementation effort

Random PopEnc
- Random $w_{i,j}$ and α, β
- Lower performance
- Less implementation effort

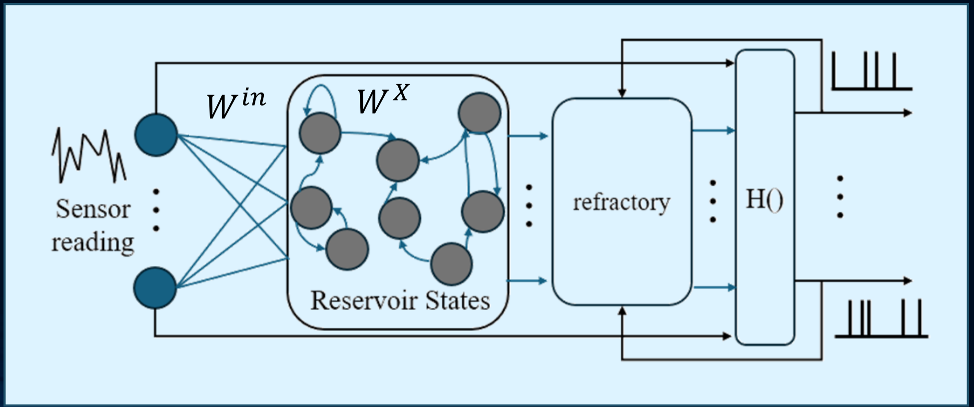


Reservoir Encoder

- Reservoir network structure
  - Total I input channels (represented as $U[t],\ U[t] \in \mathbb{R}^{I}$).
  - The J neurons form a complete directed graph
- Neuron states ($X[t] \in \mathbb{R}^{J}$) are determined by the previous state of the reservoir and the current input
  - $X[t] = (1-k)W^{X}X[t-1] + kW^{in}U[t] + B$
  - B is drawn from a uniform distribution in the range [0, 1)
  - W^{X} and W^{in} are random matrices following uniform distribution in the range $[-\varepsilon, \varepsilon]$, $\varepsilon = \sqrt{\frac{c}{fanin+fanout}}$
  - To ensure bounded neuron states, k lies within the range of [0, 1] (set to 0.9 in experiment)

Refractory and Output Generation

- Without resetting the neuron state, neurons may remain at high values for an extended period, leading to constant spiking activity
- Introducing refractory mechanism
  - $Z[t] = X[t] - \lambda V_{th} O[t-1]$
  - $O[t]$ is the output spike, $O[t] = \mathrm{H}(Z[t] - V_{th})$
  - λ controls the strength of the refractory process (we set it to 0.5)
  - If a neuron generates an output spike at the current time step, then it is less likely to fire in the next time step



Skip Connection Structure

- Limitations of stateful encoder networks
  - Due to the recurrent structure of the network, it may take a while for the change in the input pattern to affect the output activities
  - For sequential data, especially long sequence, the accumulated state of early input can blur the influence of later input.
- Skip connection can mitigate these drawbacks
  - The skip connection directly connects the input signals to the output port
    - No stateful neuron, no latency
  - The only processing is the Heaviside activation function
- With the skip connection, we have two sets of outputs $\{O^X, O^U\}$
  - O^X are the outputs from the stateful encoders (e.g., PopEnc or ResEnc)
  - $O^U = H(U)$ are the outputs from the skip connection

Backend SNN Processor

- SNN model
  - Multiple fully connected layers are used to classify the encoded sensor data
  - Each neuron is a LIF neuron with PSP similar to the model used for the PopEnc
- Online learning
  - We employ the SOLSA learning algorithm to enable online adaptation
  - SOLSA combines backpropagation and three-factor Hebbian learning and treats the LIF neuron as a recurrent network
  - It does not require unrolling the network over time, hence, can fit on edge devices

SOLSA Learning

- At each time step, the L2 error $E[t]$ is evaluated by comparing classifier's output and the target output
- Update rule: $\frac{dE}{dw_{ij}^{l}} = \sum_t \mu_j^l[t] \cdot \varepsilon_{i,j}^{l}[t]$
  - $\mu_j^l[t]$ is upper-level gradient backpropagated disregarding temporal dependencies
  - $\mu_j^l[t] = \frac{dE}{dO_j^l[t]} \frac{\partial j[t]}{\partial V_j^l[t]} = \sum_k \frac{\partial E[t]}{\partial O_k^{l+1}[t]} \frac{\partial O_k^{l+1}[t]}{\partial V_k^{l+1}[t]} \frac{\partial V_k^{l+1}[t]}{\partial F_{ki}^{l+1}[t]} \frac{\partial F_{jk}^{l+1}[t]}{\partial O_j^l[t]} \frac{\partial O_j^l[t]}{\partial V_j^l[t]}$
    - $\epsilon_j^l[t]$ is the surrogate gradient of $\mathrm{H}(V_j^l[t] - V_{th})$
  - $\varepsilon_{i,j}^{l}[t]$ is local trace updated incrementally, which stores history information
    - $\varepsilon_{i,j}^{l}[t] = \left(\lambda - \zeta \cdot v_{th} \cdot \epsilon_j^l[t]\right) \cdot \varepsilon_{i,j}^{l}\,[t-1] + F_{i,j}^{l}[t]$
  - Every time step, partial gradient $\mu_j^l[t] \cdot \varepsilon_{i,j}^{l}[t]$ are calculated and accumulated
    - Model can be updated using the partial gradient before receiving the entire input sequence

Surrogate Gradient Function

- Heaviside activation function is non-differentiable
- We use the gradient of spiking probability as a surrogate function
  - Under a Gaussian noise $z \sim N(0, \sigma)$, the probability that a neuron with membrane potential V and threshold V_{th} will fire an output spike
    - $P(V + z > V_{th}) = \frac{1}{2}\operatorname{erfc}\left(\frac{V_{th}-V}{\sqrt{2}\sigma}\right)$
  - The complementary error function is differentiable:
    - $\frac{d\ \operatorname{erfc}(x)}{dx} = -\frac{2}{\sqrt{\pi}}e^{-x^2}$
  - Use surrogate gradient, $\frac{\partial o_i^l[t]}{\partial V_i^l[t]} \approx \frac{dP(V_i^l[t]+z>V_{th})}{dV_i^l[t]}$
    - $\epsilon_i^l[t] = \sqrt{\frac{2}{\pi\sigma^2}}e^{\left(\frac{V_{th}-V_i^l[t]}{\sqrt{2}\sigma}\right)^2}$

Experiment : Datasets

- All datasets are processed by fully connected network
  - Multivariate Time Series datasets
  - Sequence length varies from 30 to 1000 time steps
  - Sequences are recorded sensor readings. The data are floating point real numbers representing the readings of the sensors

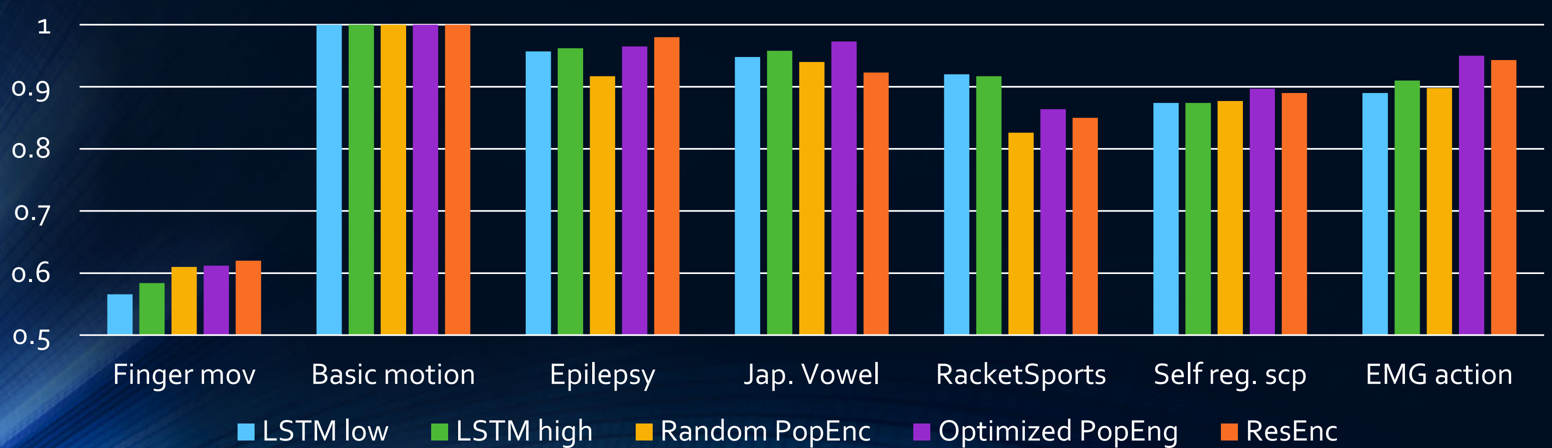
	Dataset Name	# of Input Channels (I)	Sequence length	# of Classes
Regular	Finger mov.[24]	28	50	2
	Basic motion[22]	6	100	4
	Epilepsy[23]	3	207	4
	Jap. Vowel[26]	12	29	9
	RacketSports[22]	6	30	4
Long	Self reg. scp1[25]	6	896	2
	EMG action[27]	8	1000	10

Experiment: Settings

- Three different types of spike encoders with backend SNN running SOLSA learning
  - Random PopEnc (the connections of the input layer have random weights)
  - Optimized PopEnc (the connections of the input layer are also optimized using SOLSA learning)
  - ResEnc (with random weight matrices and bias)
- Use Long Short-Term Memory (LSTM) with similar complexity (i.e., the number of trainable parameters) as baseline
  - One LSTM layer and one fully connected layer
  - Exact match in complexity is hard to achieve, hence we created two models
    - LSTM-Low, which is slightly smaller than SNN
    - LSTM-High, which is slightly larger than SNN
  - The frontend of the LSTM-based system requires high precision analog-to-digital converters (ADC)

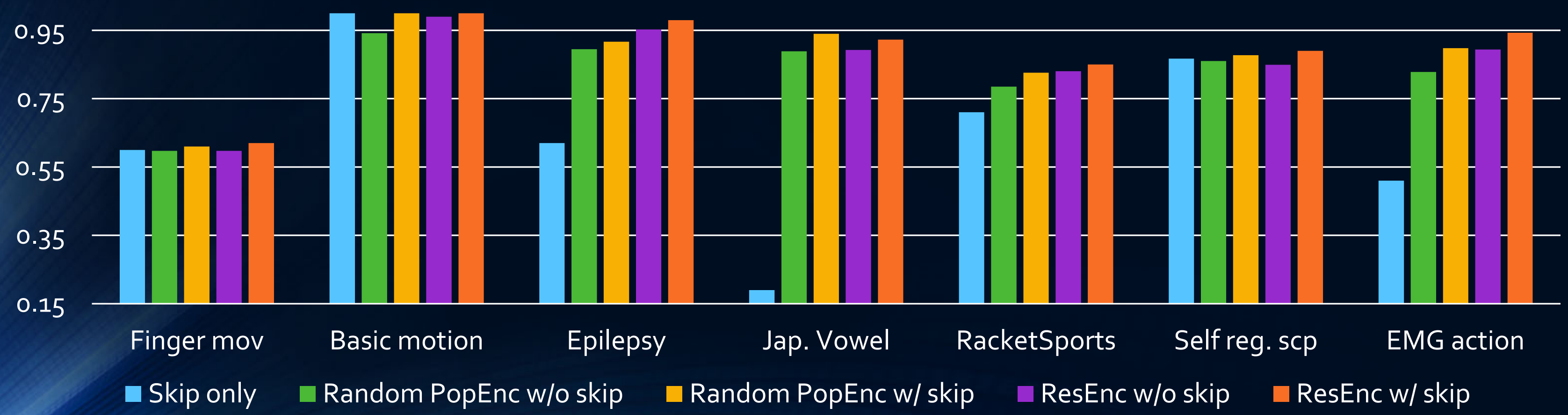
Experiment: Performance Comparison

- Overall, the three SNN-based systems deliver similar or even superior performance compared to the LSTM-based systems
  - Optimized PopEnc perform slightly better than the ResEnc
  - ResEnc achieves performance comparable to the optimized PopEnc with lower implementation effort (no optimization)
  - Random PopEnc has lowest performance among the three encoders



Ablation Study: Skip Connection

- Compared the performance of random PopEnc and the ResEnc with and without skip connections, and a skip only frontend
  - Results show that the skip connection by itself is insufficient to function as a general-purpose spike encoder
  - However, in average, it improves the classification accuracy by 4% and 3% for systems with random PopEnc and ResEnc respectively



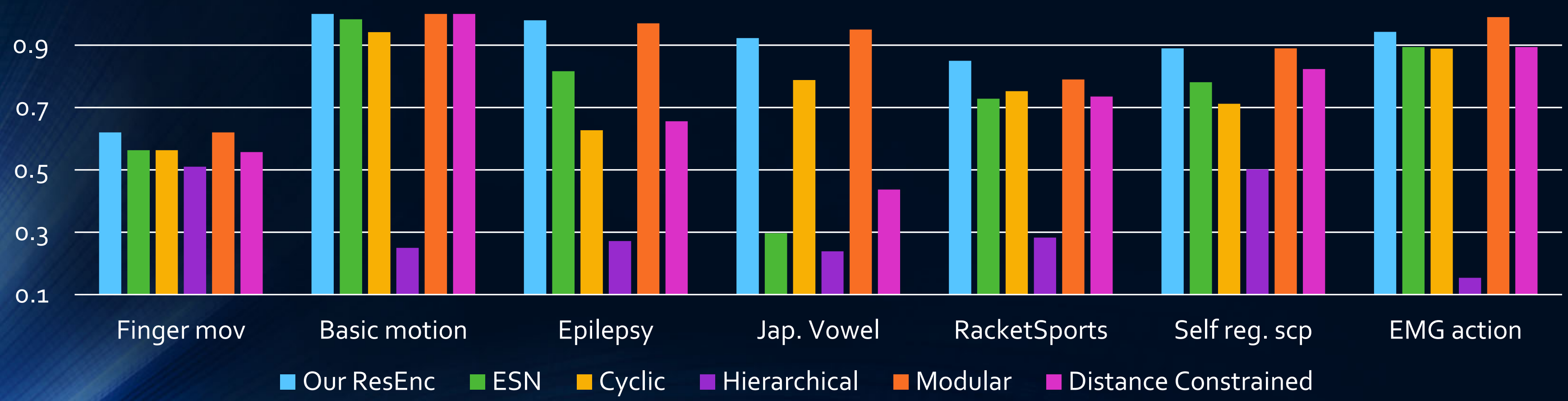
Ablation Study: Reservoir Network Complexity

- Compare our simplified reservoir encoder with encoders based on some traditional reservoir models
  - All encoders are implemented using the same number of neurons and same method of weight coefficient generation
  - Hardware complexity is expressed as a function of input channel size I
    - For all encoders, the output size is $5I$
  - Our encoder requires much less memory and simpler activation functions

	Ours	ESN	Cyclic	Hierarchical	Modular	Distance Constrained
# of connections	$20 \cdot I^2$	$30 \cdot I^2$	$30 \cdot I^2$	$80 \cdot I^2$	$10 \cdot I^2$	$30 \cdot I^2$
# of neurons	$4 \cdot I$	$5 \cdot I$	$5 \cdot I$	$15 \cdot I$	$5 \cdot I$	$5 \cdot I$
Memory	1	1	5	1	1	1
Activation function	Heaviside	tanh	tanh	tanh	tanh	tanh

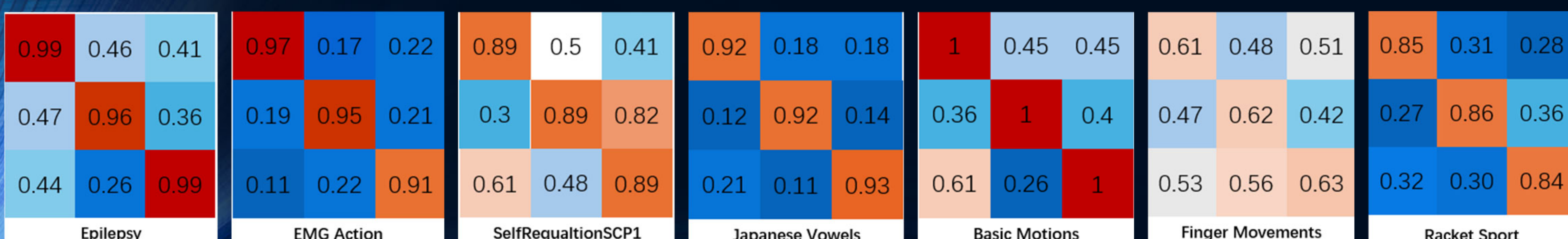
Ablation Study: Reservoir Network Performance

- We compared the classification accuracy achieved when different reservoir-based encoders are used as the frontend
  - Our ResEnc consistently delivers the highest or near-highest performance across all datasets
  - The simplification of our ResEnc network structure does not compromise its sparse coding capabilities



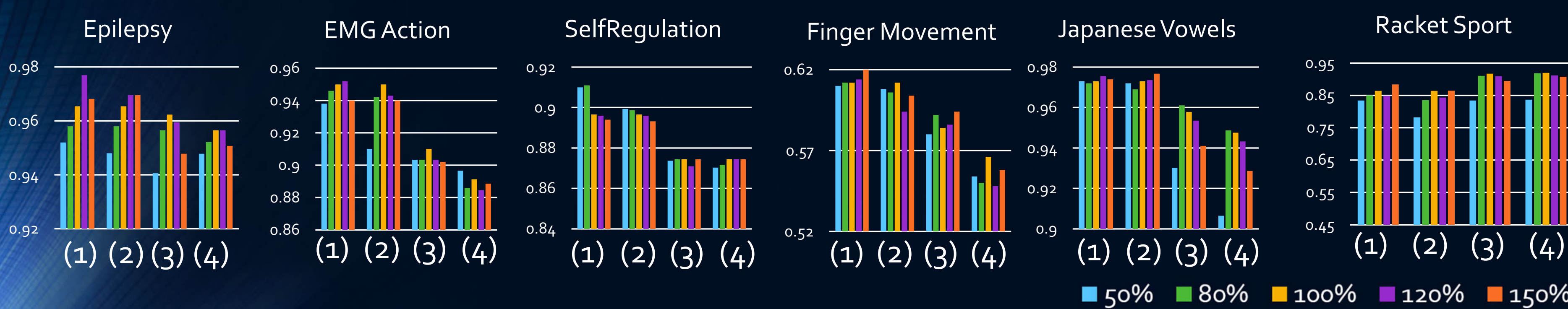
Experiment: Hardware Variations (ResEnc)

- Variation arises from the random weights in the encoders
  - Three different versions of ResEnc's are created
  - For each encoder, a backend SNN classifier was trained, and the combinations of different encoders and classifiers were then tested
  - When the classifier and encoder do not match, performance drops significantly (from around 15% to over 80%)
  - Without in-hardware adaptability, variations in the encoder hardware have a profoundly negative impact on model performance



Experiment: Timing Variations (PopEnc)

- Variation arises from sampling rate deviation of the sensor output
  - Train the model using sequences sampled using nominal sampling rate
  - We varied the sampling rate from 150% to 50% of the nominal value and regenerated sequences of sensor readings using linear interpolation for testing
  - In extreme cases, offline approaches accuracy drop 8.3%, while the systems with online adaptation only drop 3%



(1) PopEnc+SNN with online adapted frontend and backend
(2) PopEnc+SNN without online adaptation
(3) Fixed LSTM-high model
(4) Fixed LSTM-low model

Conclusion

- We proposed two new spike encoder designs
  - Population Coding-based encoder (PopEnc)
  - Reservoir Computing-based encoder (ResEnc)
- Experiment results show that
  - Encoders effectively preserve the temporal and spatial features of the sensor sequences and enhancing the performance of the SNN-based backend classifiers
  - Skip connection structure can enhance the performance of spike encoder
- We also showed the importance of the backend SNN's online learning capability, which enables adaptation to the frontend encoder
  - The adaptability accommodates randomness and variations in the encoder, significantly reduce design and implementation complexity

Thank you!