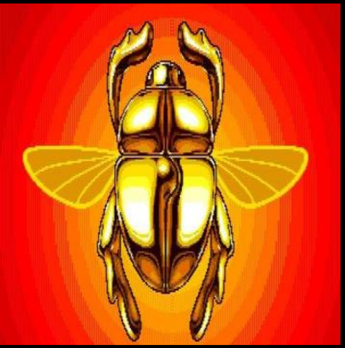


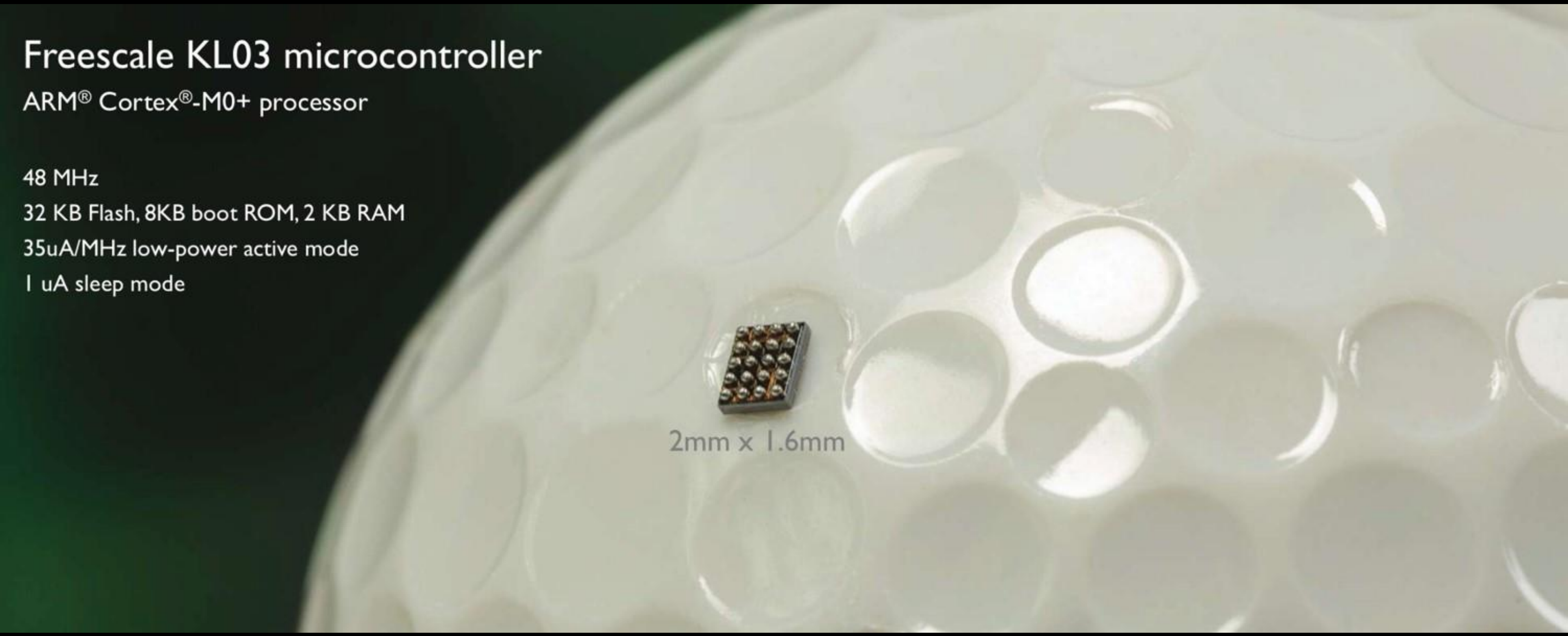

The Edge of Machine Learning

Resource-efficient ML in 2 KB RAM for the Internet of Things

Prateek Jain
Microsoft Research India

D. Dennis, P. Jain, A. Kusupati, N. Natarajan,
S. Patil, R. Sharma, H. Simhadri & M. Varma

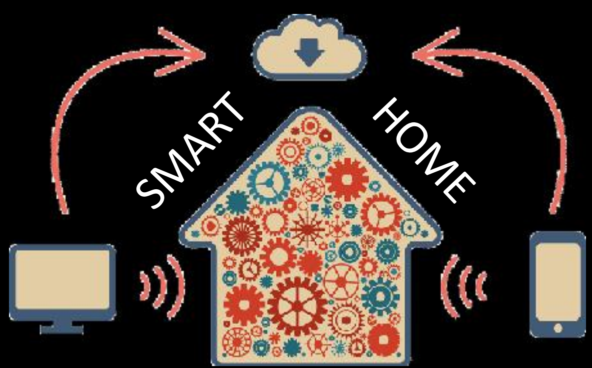
Resource-constrained IoT Devices

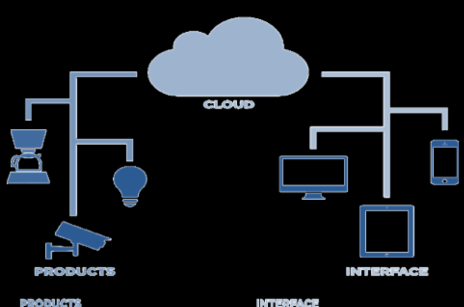


ARM Cortex M0+ at 48 MHz & 35 μA/MHz
with 2 KB RAM & 32 KB read only Flash

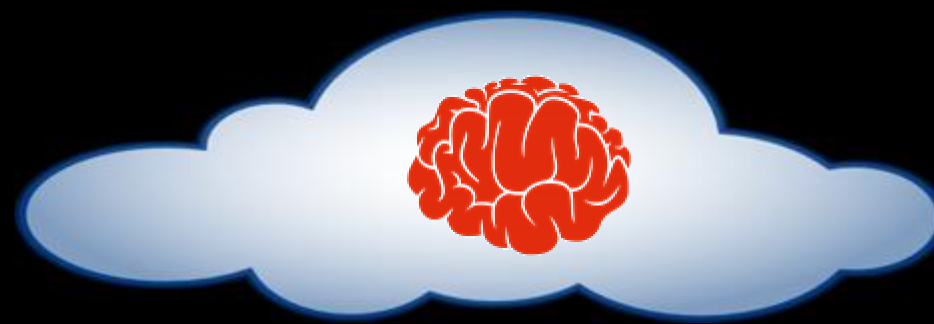
The Internet of Things

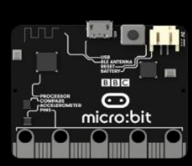
Smart City

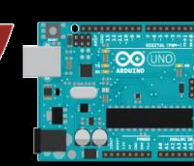




Smart Appliances

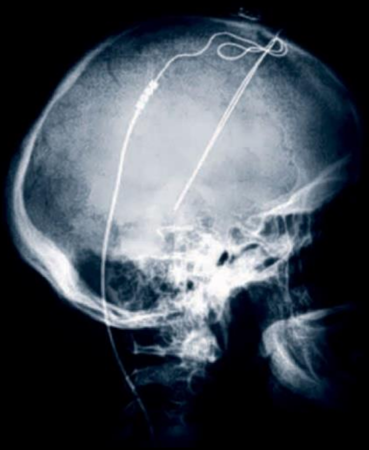




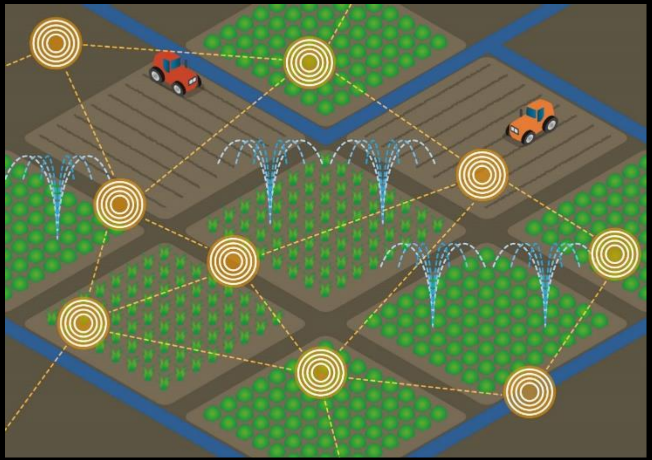


Intelligent IoT Devices

- Intelligent IoT devices can help deal with latency, bandwidth, privacy and energy concerns



Low latency brain implants



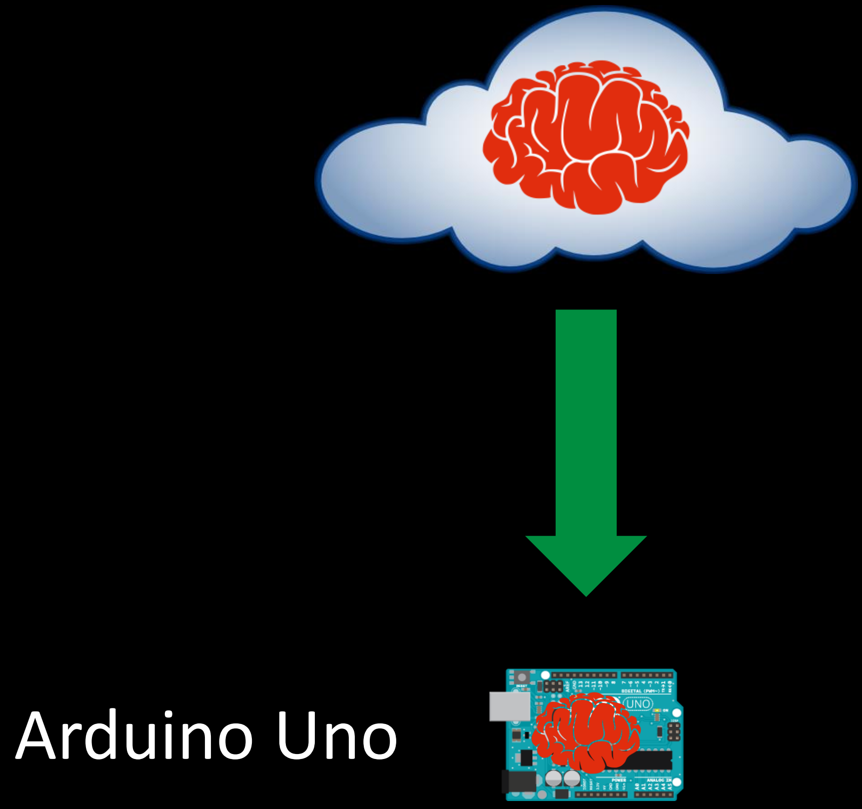
Smart agriculture for disconnected farms

Privacy preserving smart glasses

Energy efficient smart forks

Algorithms for the IDE – Objectives

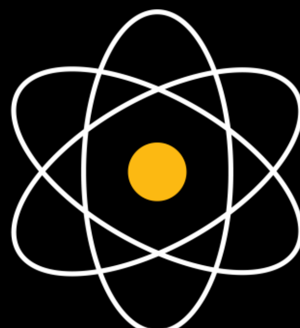
- To build a library of machine learning algorithms
  - Which can be trained on the cloud
  - But which will run on tiny IoT devices



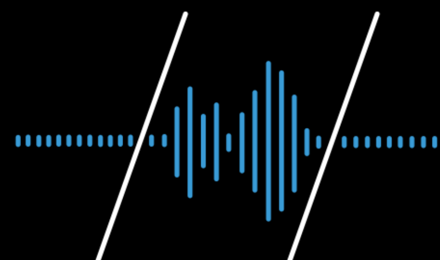
Microsoft’s EdgeML Library

- Compact tree, kNN and RNN algorithms for classification, regression, ranking, time series, *etc*.

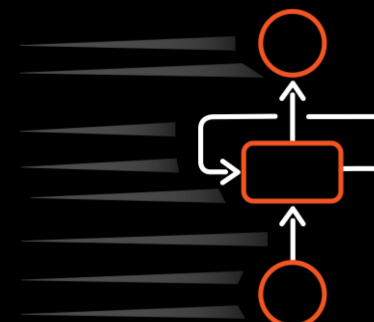
Bonsai



ProtoNN



EMI-RNN



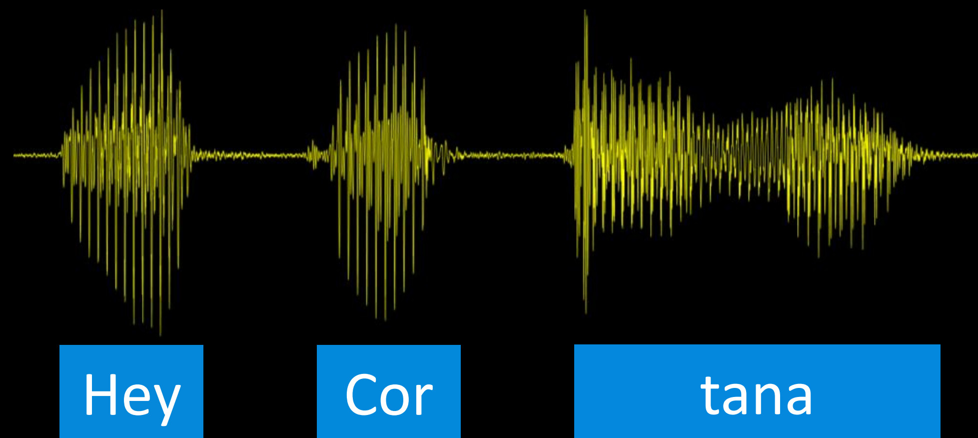
FastGRNN

https://github.com/Microsoft/EdgeML

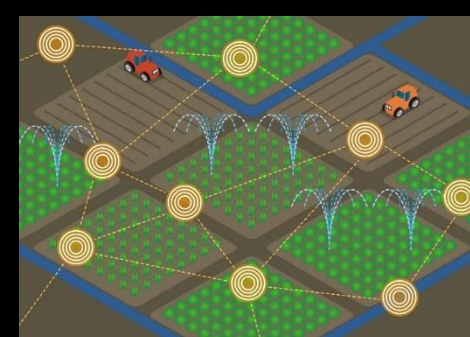
Time series data

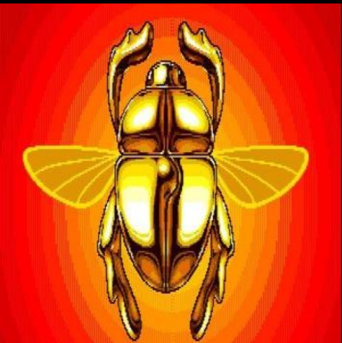
- Time series is ubiquitous in most of the IoT devices as the sensory data is often modelled as Time series.

“Hey Cortana”



Soil moisture during a day





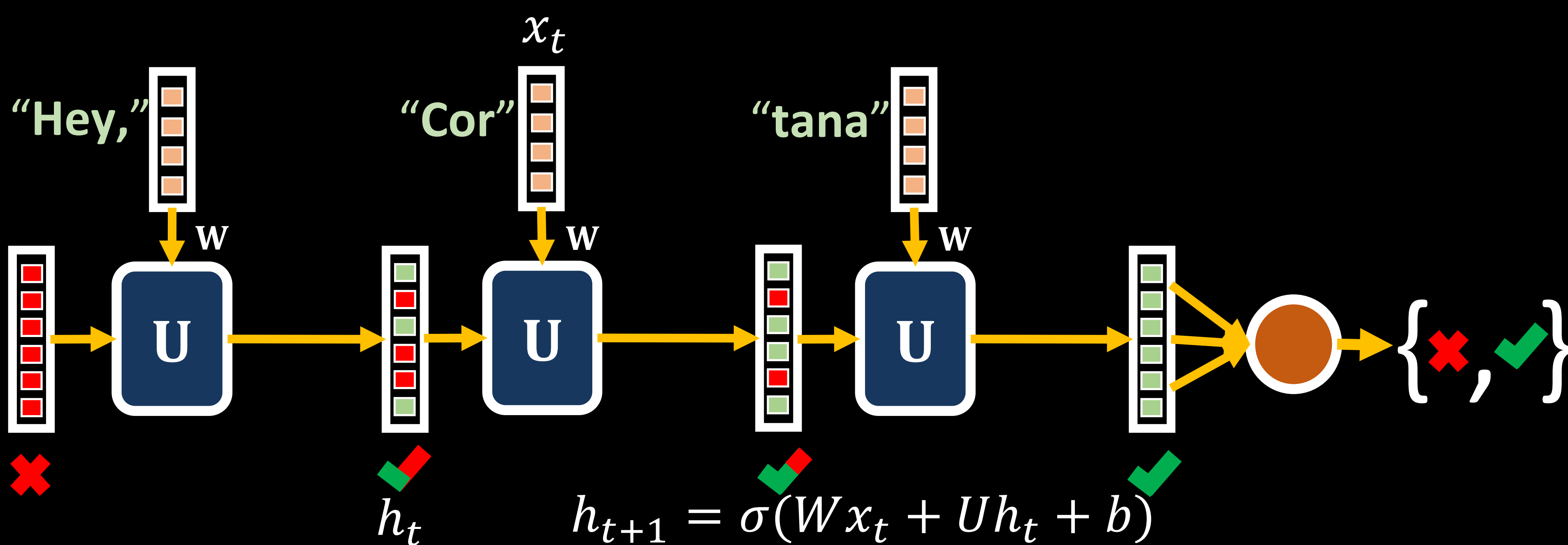

FastGRNN

A Fast, Accurate, Stable & Tiny (Kilobyte Sized) Gated RNN

A. Kusupati (MSRI), M. Singh (IITD), K. Bhatia (Berkeley), A. Kumar (Berkeley), P. Jain (MSRI) & M. Varma (MSRI)

Recurrent Neural Networks (RNNs)

- State-of-the-art for analyzing sequences & time series
- Training is unstable due to exploding & vanishing gradients

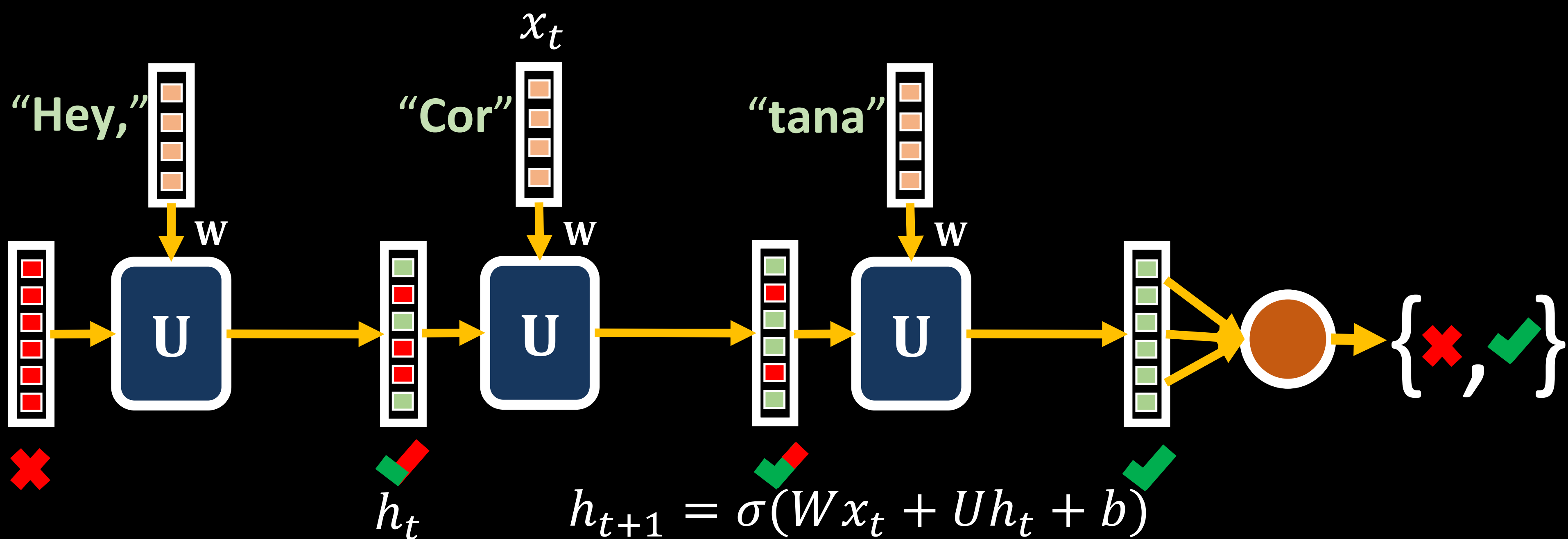


$$h_{t+1} = \sigma(Wx_t + Uh_t + b)$$

$$\nabla = f\left(\dots, \mathbf{U}^T = \mathbf{Q}\begin{bmatrix}(1+\lambda)^T & & \\ & \ddots & \\ & & (1-\gamma)^T\end{bmatrix}\mathbf{Q}^\top, \dots\right)$$

Unitary RNNs – uRNN, SpectralRNN, …

- Unitary RNNs force all the eigenvalues of $\mathbf{U}$ to be ≈ 1
- Unfortunately, they are expensive to train & lack accuracy



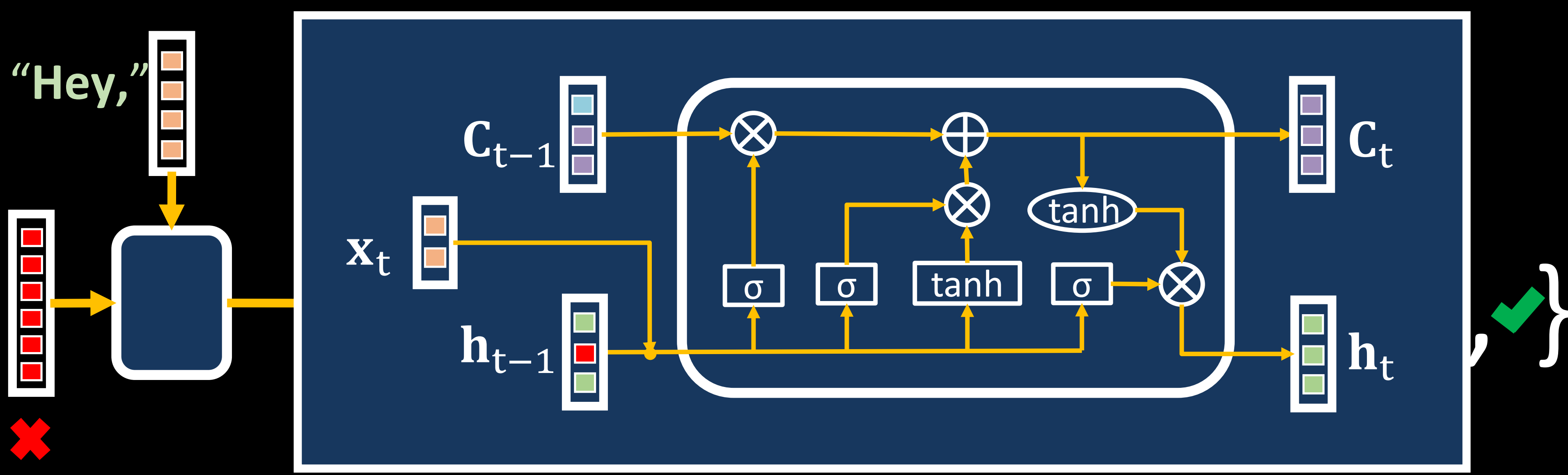
$$h_{t+1} = \sigma(Wx_t + Uh_t + b)$$

$$\nabla = \; f\left(\ldots, \mathbf{U}^T = \mathbf{Q}\begin{bmatrix} 1^T & & \\ & \ddots & \\ & & 1^T \end{bmatrix}\mathbf{Q}^\top, \ldots\right)$$

Gated RNNs – LSTM, GRU, …

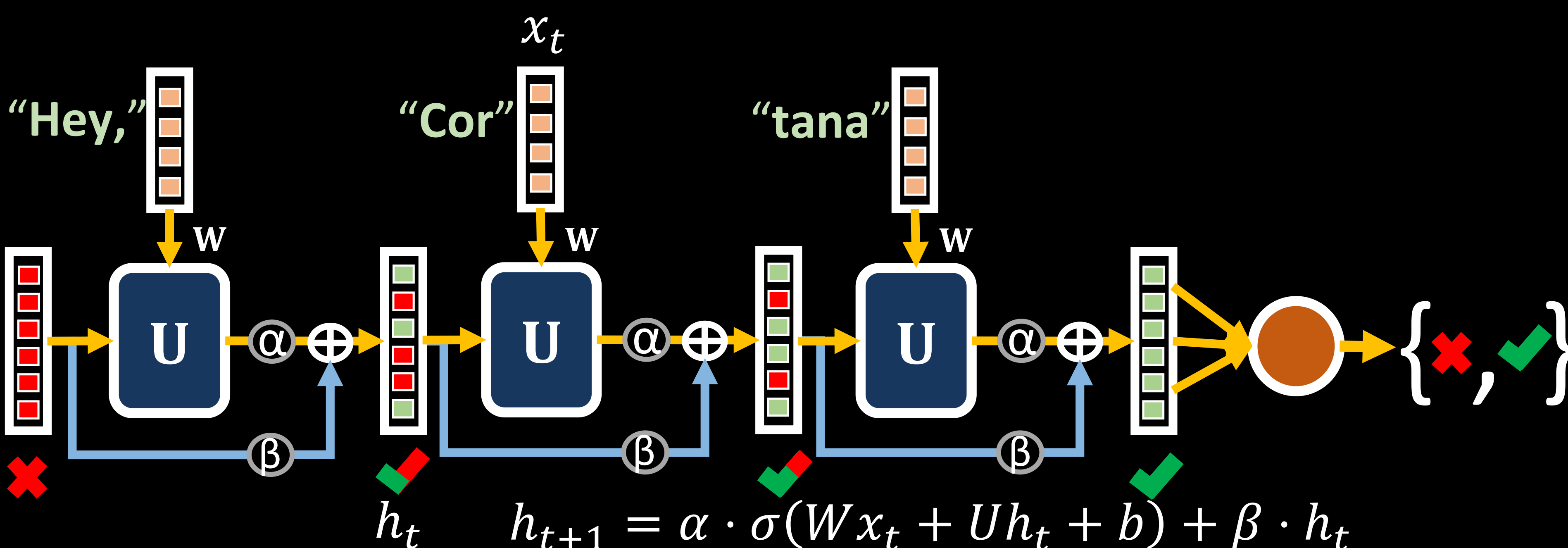
- Add extra parameters to stabilize training
- Have increased prediction costs on IoT microcontrollers
- Have intuitive explanations but lack formal guarantees

LSTM Cell



FastRNN

- Provably stable training with 2 additional scalars
- Accuracy: RNN ≪ Unitary RNNs < FastRNN < Gated RNNs



$$h_{t+1} = \alpha \cdot \sigma(Wx_t + Uh_t + b) + \beta \cdot h_t$$

$$\nabla = \; f\left(\ldots, (\alpha\mathbf{UD} + \beta\mathbf{I})^T = \mathbf{Q}\begin{bmatrix} (\beta + \alpha\|\mathbf{UD}\|)^T & & \\ & \ddots & \\ & & (\beta - \alpha\|\mathbf{UD}\|)^T \end{bmatrix}\mathbf{Q}^\top, \ldots\right)$$

Theorems

Convergence Bound:

- If: $\alpha \approx O(1/T)$, $\beta = 1 - \alpha$
- Convergence to station point in $O(1/_{\epsilon^2})$ iterations
  - Independent of T!

Generalization Error Bound:

- $O(\frac{1}{\sqrt{n}})$, where $\alpha \approx O(1/T)$, $\beta = 1 - \alpha$
- Independent of T!

Similar analysis in each case shows exponential bounds (in T) for standard RNN

Theorems

Convergence to stationary point:

$$\mathrm{E}\left[\left\|\nabla_\theta L(\hat{\theta})\right\|_2^2\right] \leq \frac{O(\alpha \mathrm{T}) L(\theta_0)}{N} + \left(\bar{D} + \frac{4R_{\mathbf{W}}R_{\mathbf{U}}R_{\mathbf{v}}}{\bar{D}}\right)\frac{O(\alpha \mathrm{T})}{\sqrt{N}}$$

Generalization Error Bound:

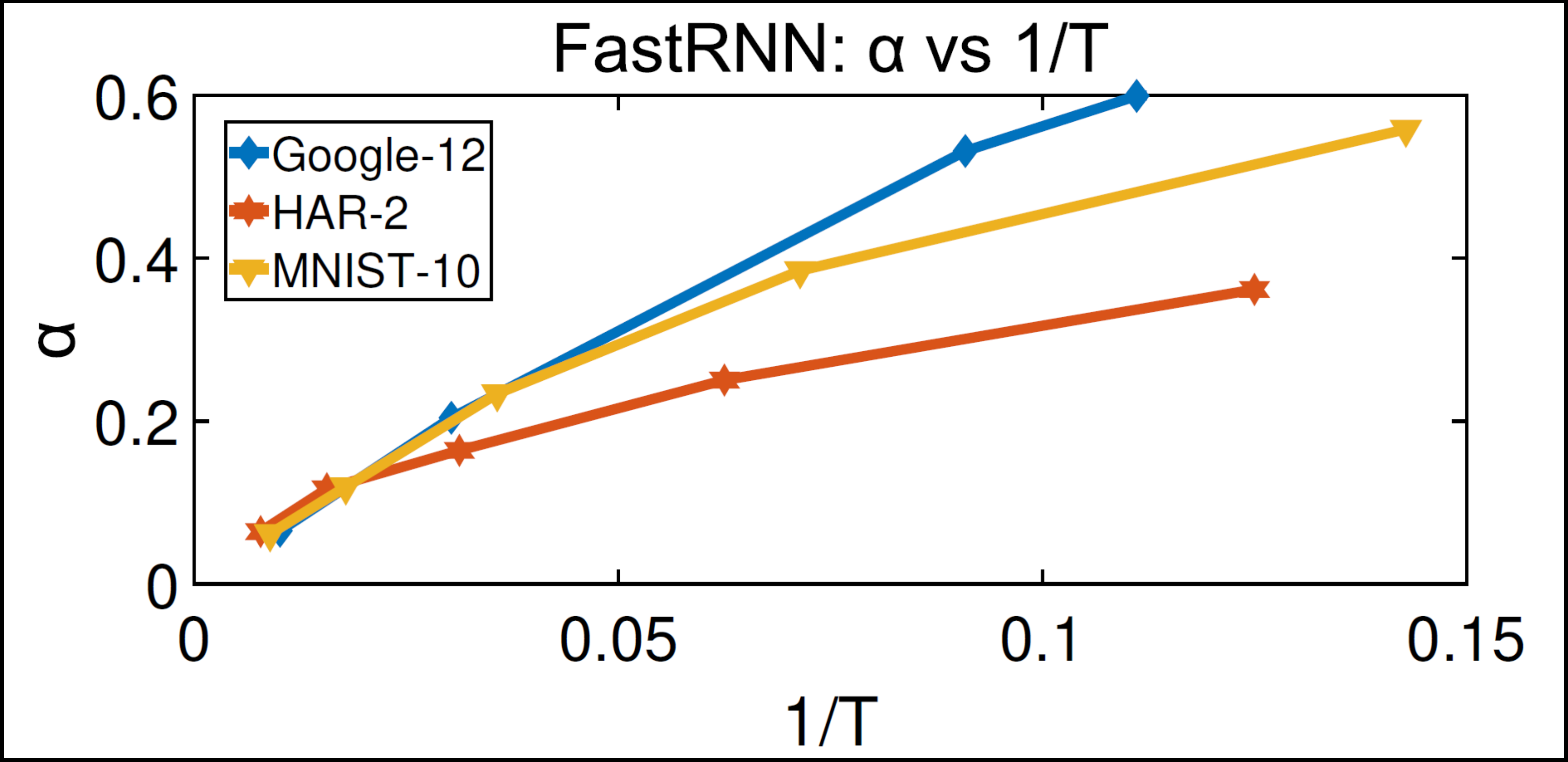
$$\varepsilon \leq C\frac{O(\alpha \mathrm{T})}{\sqrt{n}} + B\sqrt{\frac{\ln\left(\frac{1}{\delta}\right)}{n}}$$

$\alpha \approx O\left(\frac{1}{T}\right), \beta = 1 - \alpha \Rightarrow$ independent of T!!!

N is # SGD iterations and n is # datapoints.

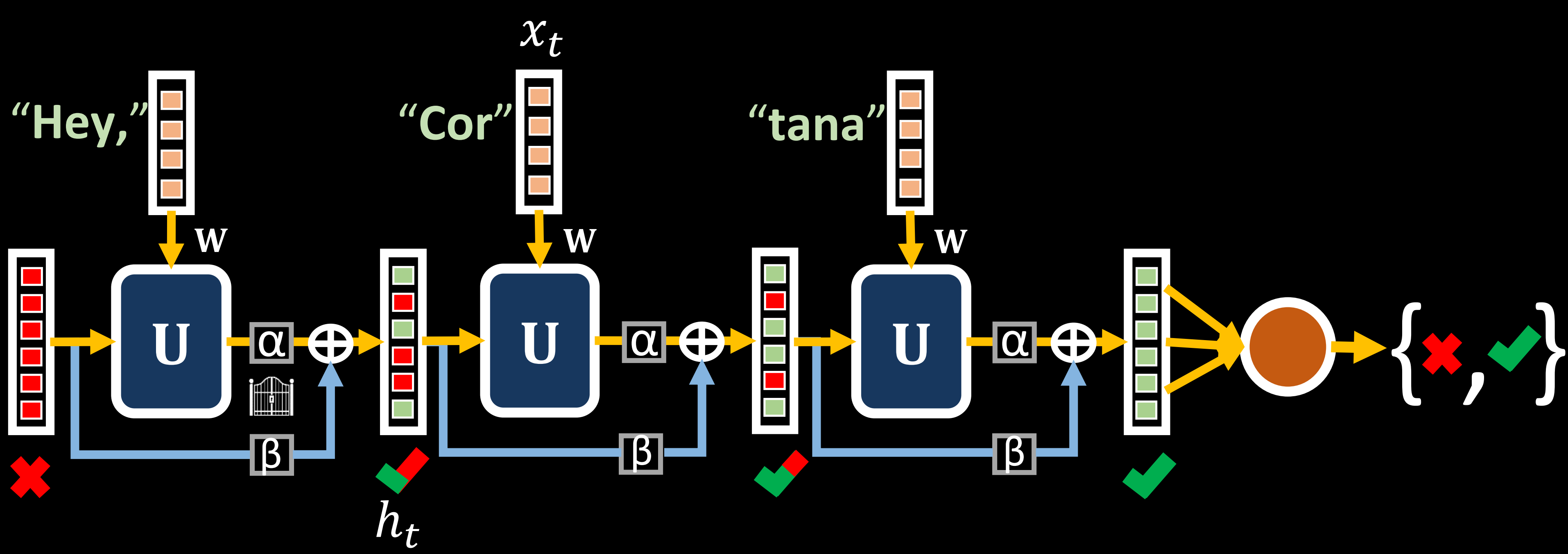
$\bar{D} \geq 0$ helps in choosing right learning rate. $R_{\mathbf{X}} = max_{\mathbf{X}} \|\mathbf{X}\|_{\mathrm{F}}$.

FastGRNN: α?



FastGRNN

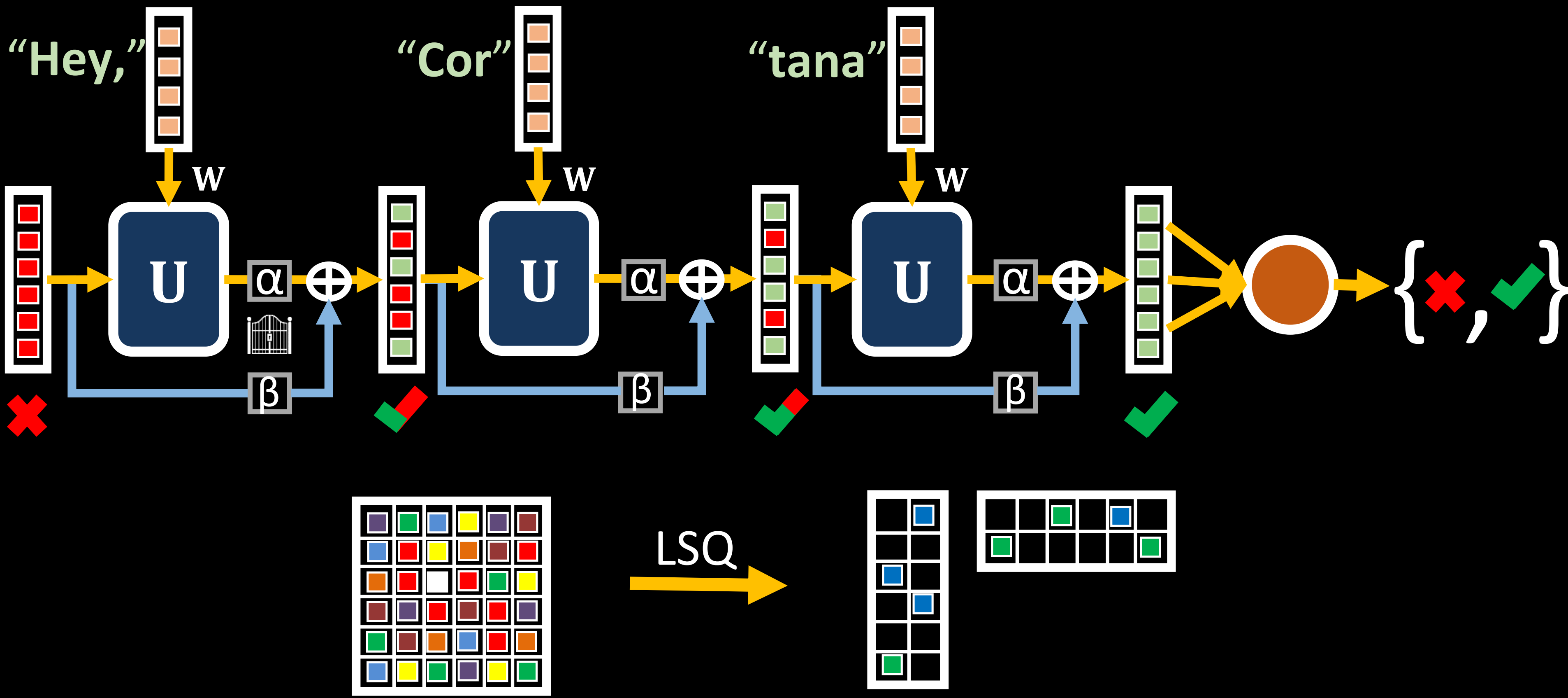
- Extend α & β from scalars to vector gates
- Accuracy: RNN ≪ Unitary RNNs < Gated RNNs ≈ FastGRNN



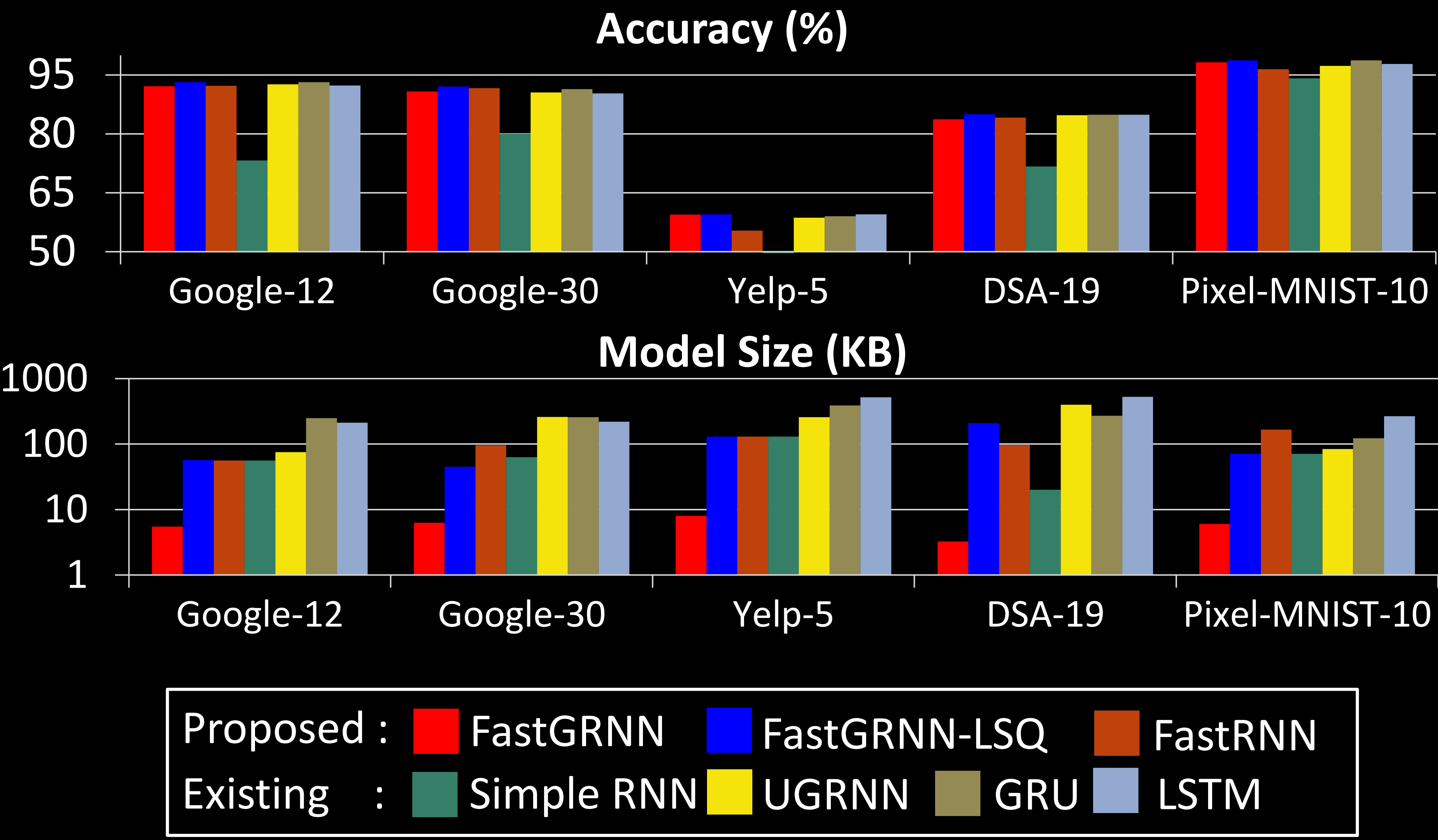
$$h_{t+1} = \alpha \odot \sigma(Wx_t + Uh_t + b) + \beta \odot h_t$$
$$\alpha = \tanh(Wx_t + Uh_t + b_\alpha), \beta = 1 - \alpha$$

FastGRNN

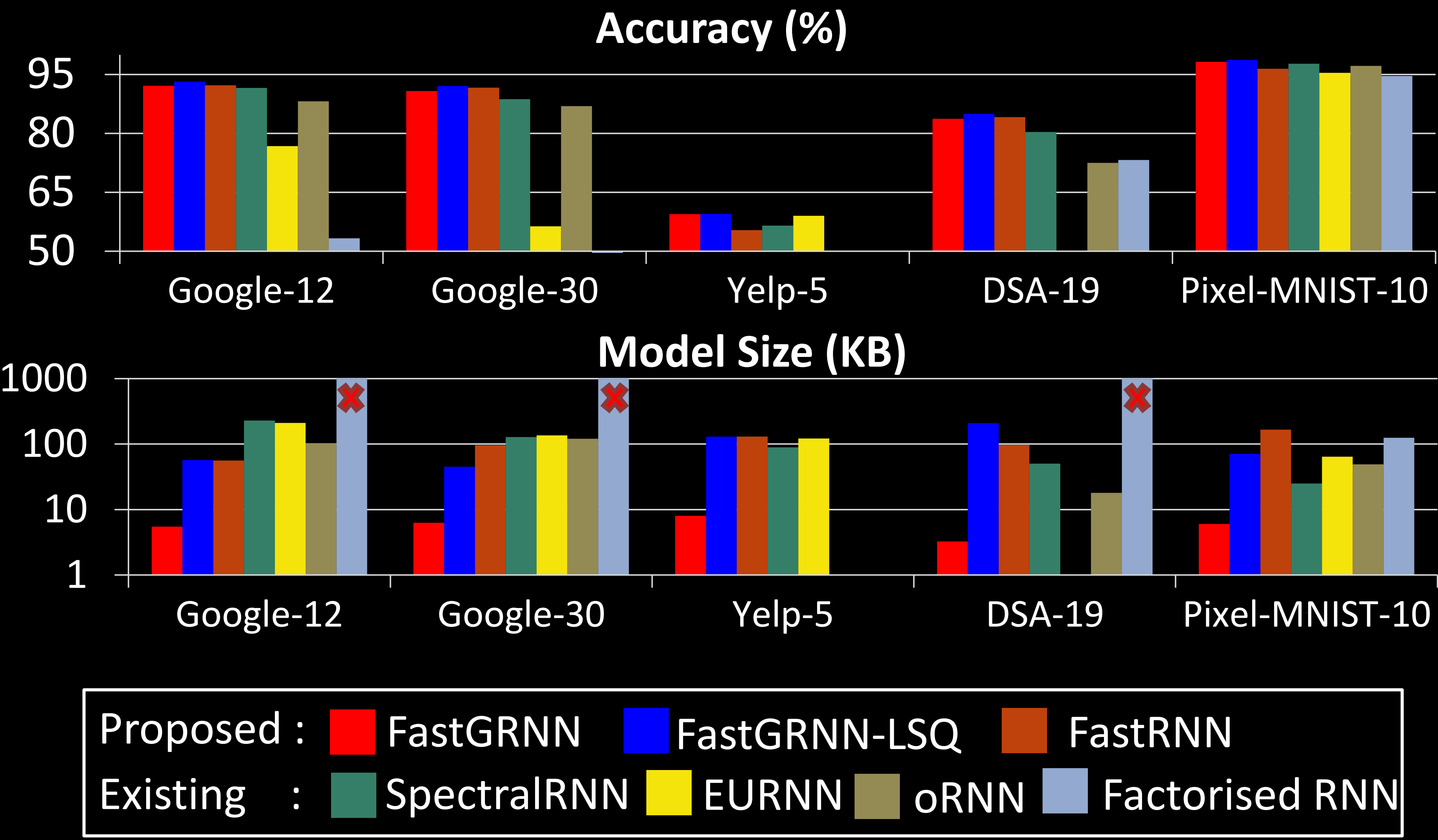
- Make **U** and **W** low-rank (L), sparse (S) and quantized (Q)
- Model Size: FastGRNN ≪ RNN ≈ Unitary RNNs < Gated RNNs



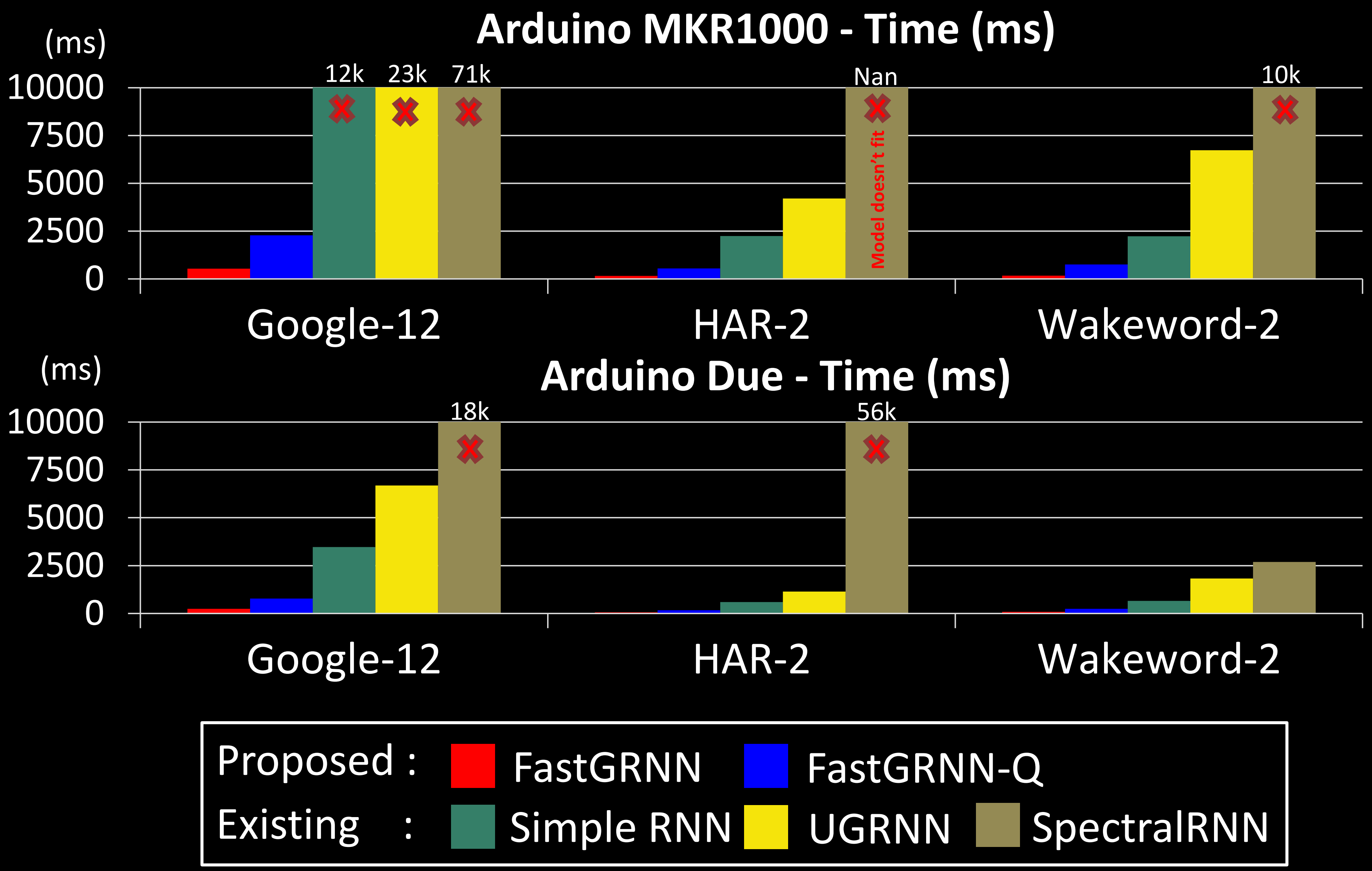
Comparison to Gated Architectures

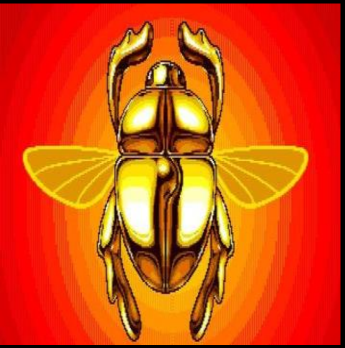


Comparison to Unitary Architectures



Prediction on Edge Devices



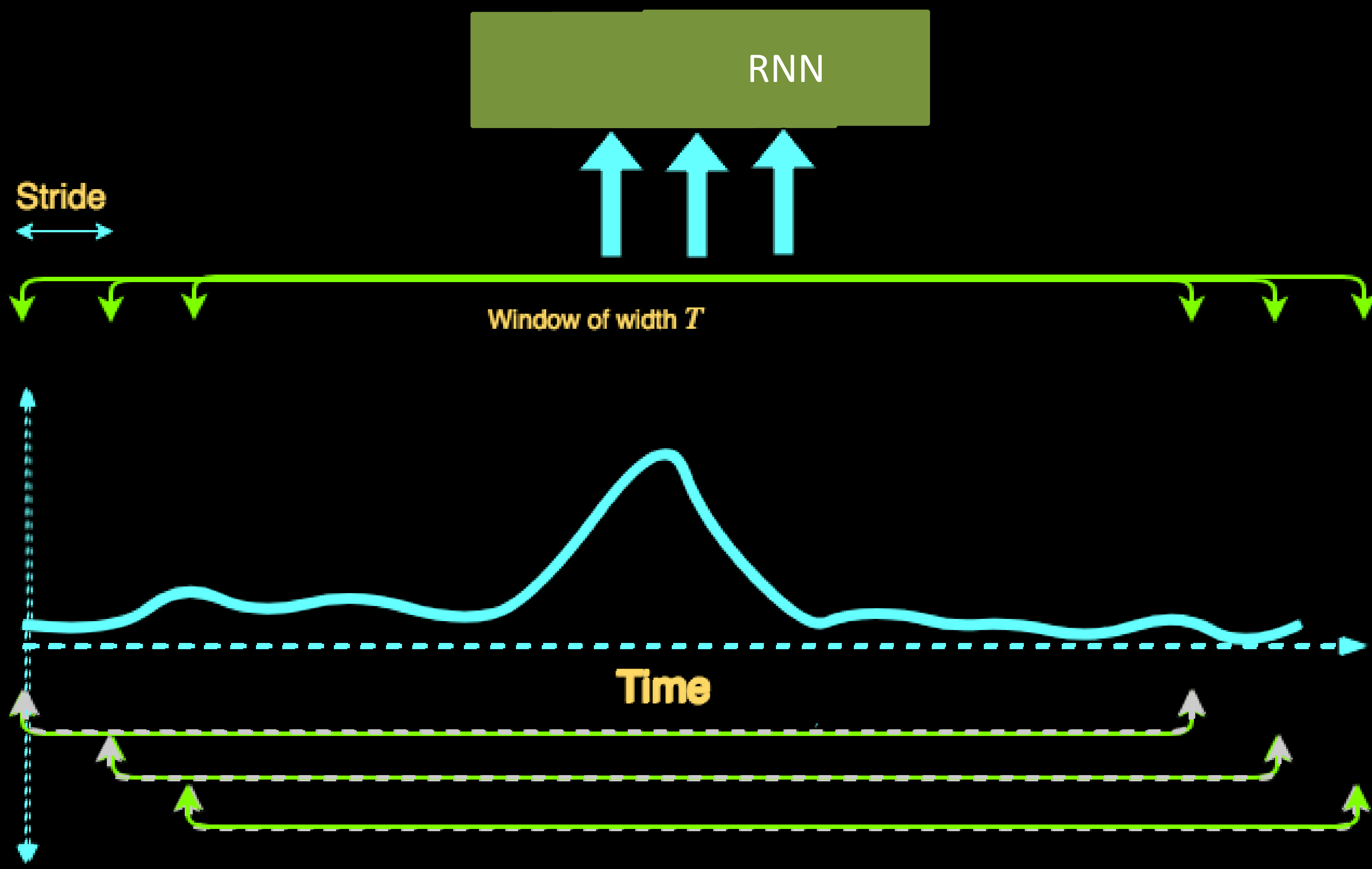



EMIRNN

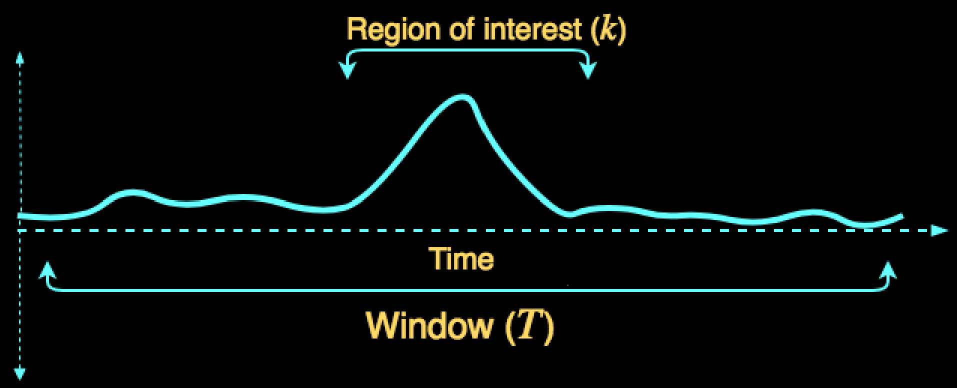
Simultaneous segmentation and early prediction in RNNs

Don Dennis (MSRI), Chirag P (MSRI),
Harsha Simhadri (MSRI), P. Jain (MSRI)

Time-series Analysis: Sliding Windows



Class Signatures are Tiny

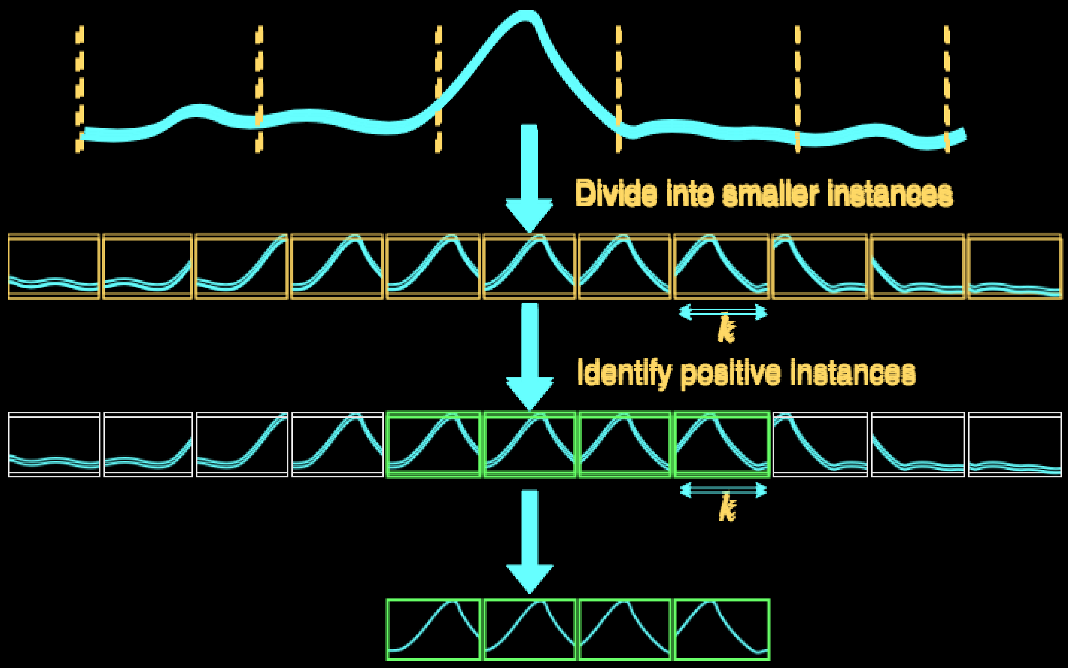


Typically $k \ll T$, *i.e.,* actual signature of event is tiny

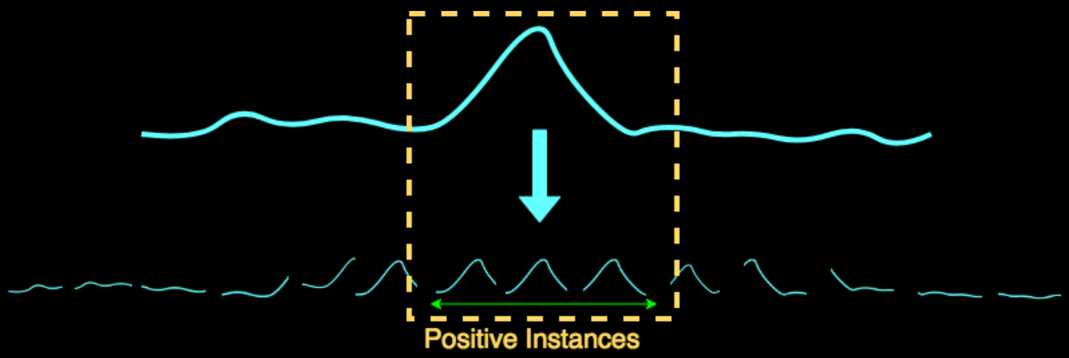
- Audio clips: 2-5secs but “Hey Cortana” typically spoken in <1sec
- *Unnecessarily large T ---* longer prediction time, lag
- Predictors must recognize signatures with different offsets - *requires larger* predictors.

EMI-RNN: Approach

- STEP 1: Divide *X* into smaller instances.
- STEP 2: Identify positive instances. Discard negative (noise) instances.
- STEP 3: Use these instances to train a smaller classifier.



EMI-RNN: Approach



Exploit temporal locality with MIL/Robust learning techniques

Property 1: Positive instances are clustered together.

Property 2: Number of positive instances can be estimated.

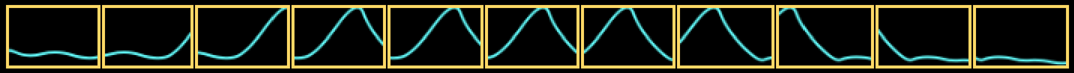
EMI-RNN: Algorithm

Two phase algorithm – alternates between identifying positive instances and training on the positive instances.

EMI-RNN: Algorithm

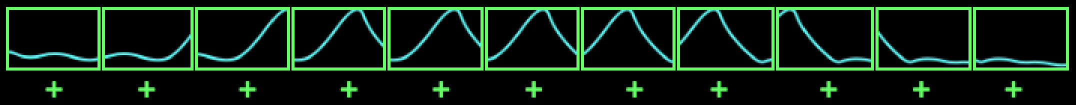
- Step 1:
  Assign labels
  Instance = source
  data

EMI-RNN: Algorithm



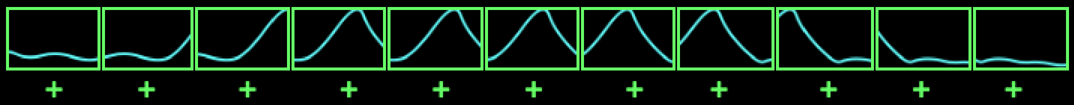
- Step 1:
  Assign labels
  Instance = source
  data

EMI-RNN: Algorithm



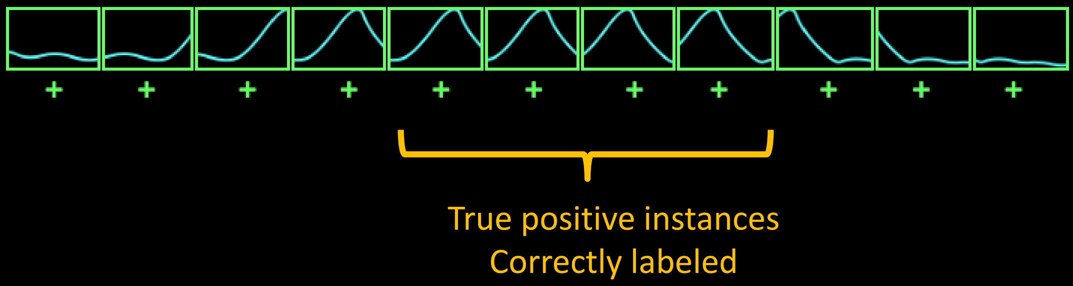
- Step 1:
  Assign labels
  Instance = source data

EMI-RNN: Algorithm



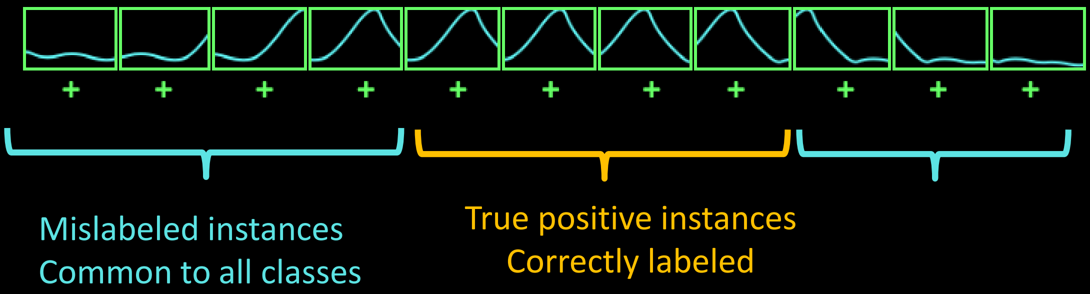
- Step 2:
  Train classifier on this data

EMI-RNN: Algorithm



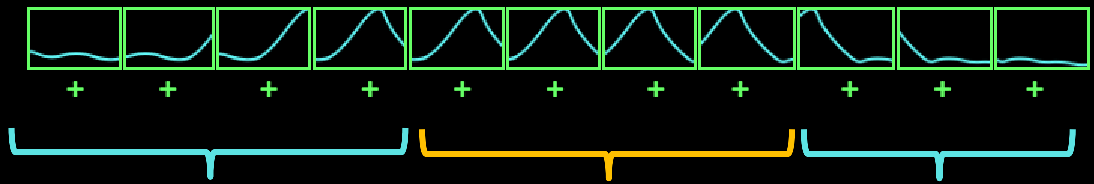
- Step 2:
  Train classifier on this data

EMI-RNN: Algorithm



- Step 2:
  Train classifier on this data

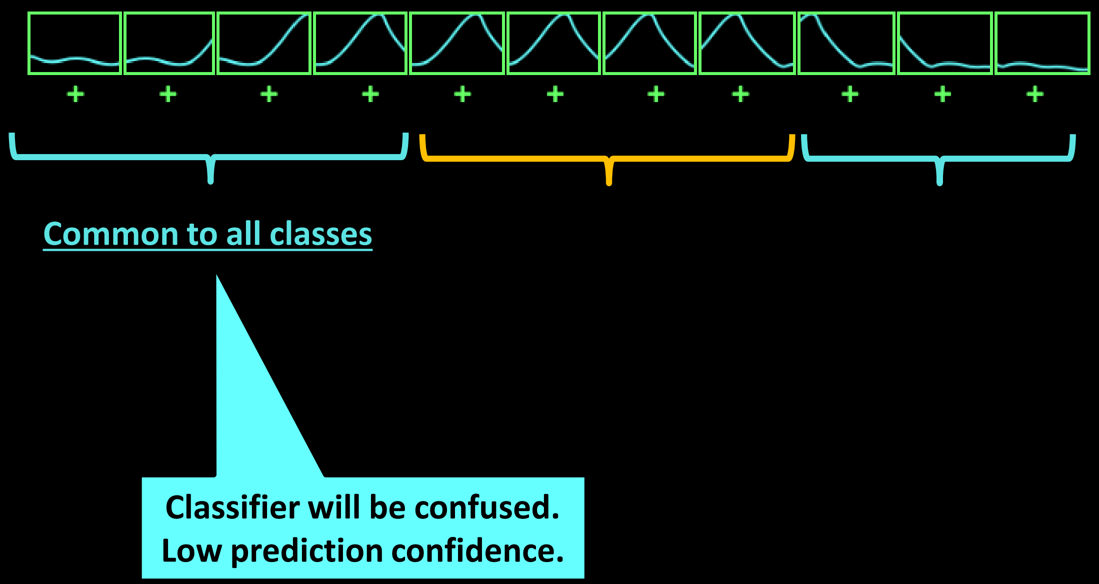
EMI-RNN: Algorithm



Common to all classes

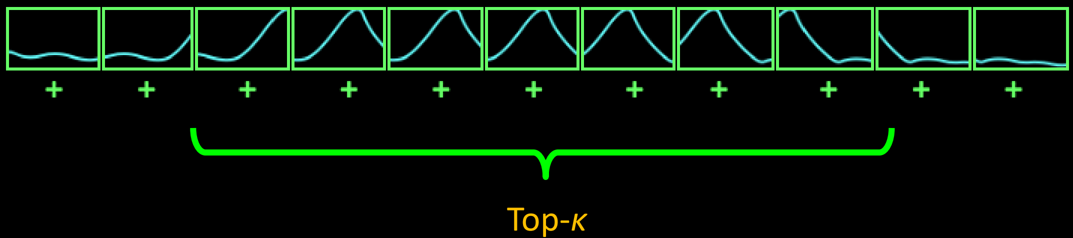
- Step 2:
  Train classifier on this data

EMI-RNN: Algorithm



- Step 2:
  Train classifier on this data

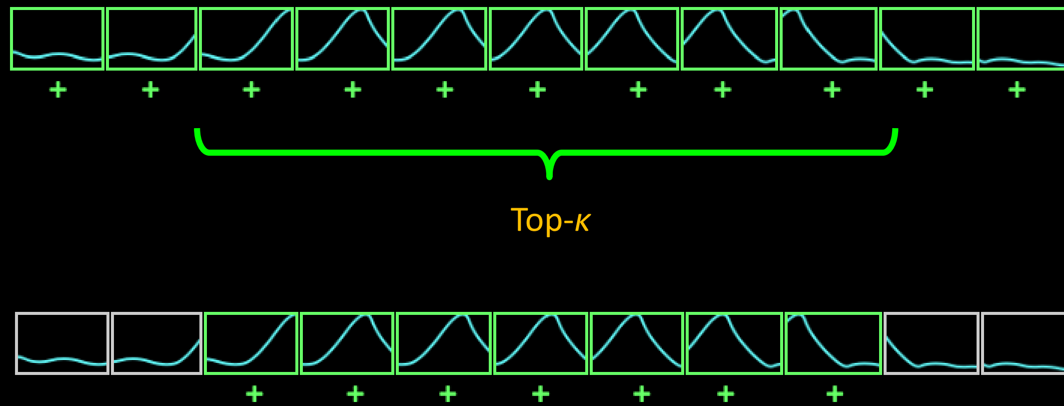
EMI-RNN: Algorithm



- Step 3:
  Wherever possible, use classifier’s prediction score to pick top-κ

  Should satisfy property 1 and property 2

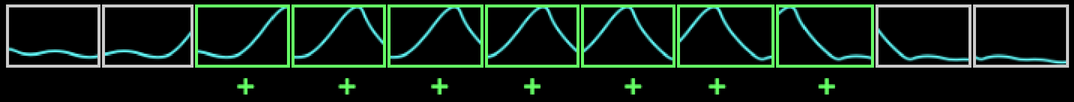
EMI-RNN: Algorithm



- Step 3:
  Wherever possible, use classifier's prediction score to pick top-κ

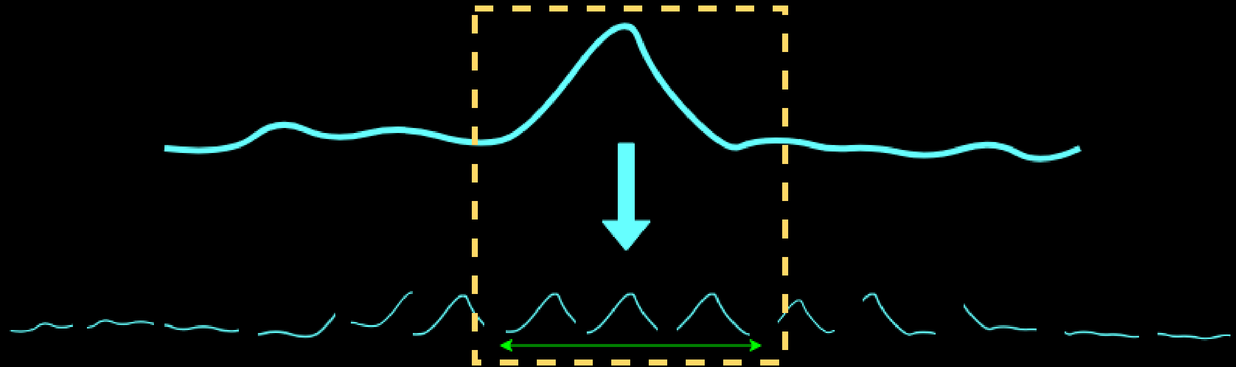
  Should satisfy property 1 and property 2

EMI-RNN: Algorithm



- Step 4:
  Repeat with new labels

EMI-RNN: Analysis?



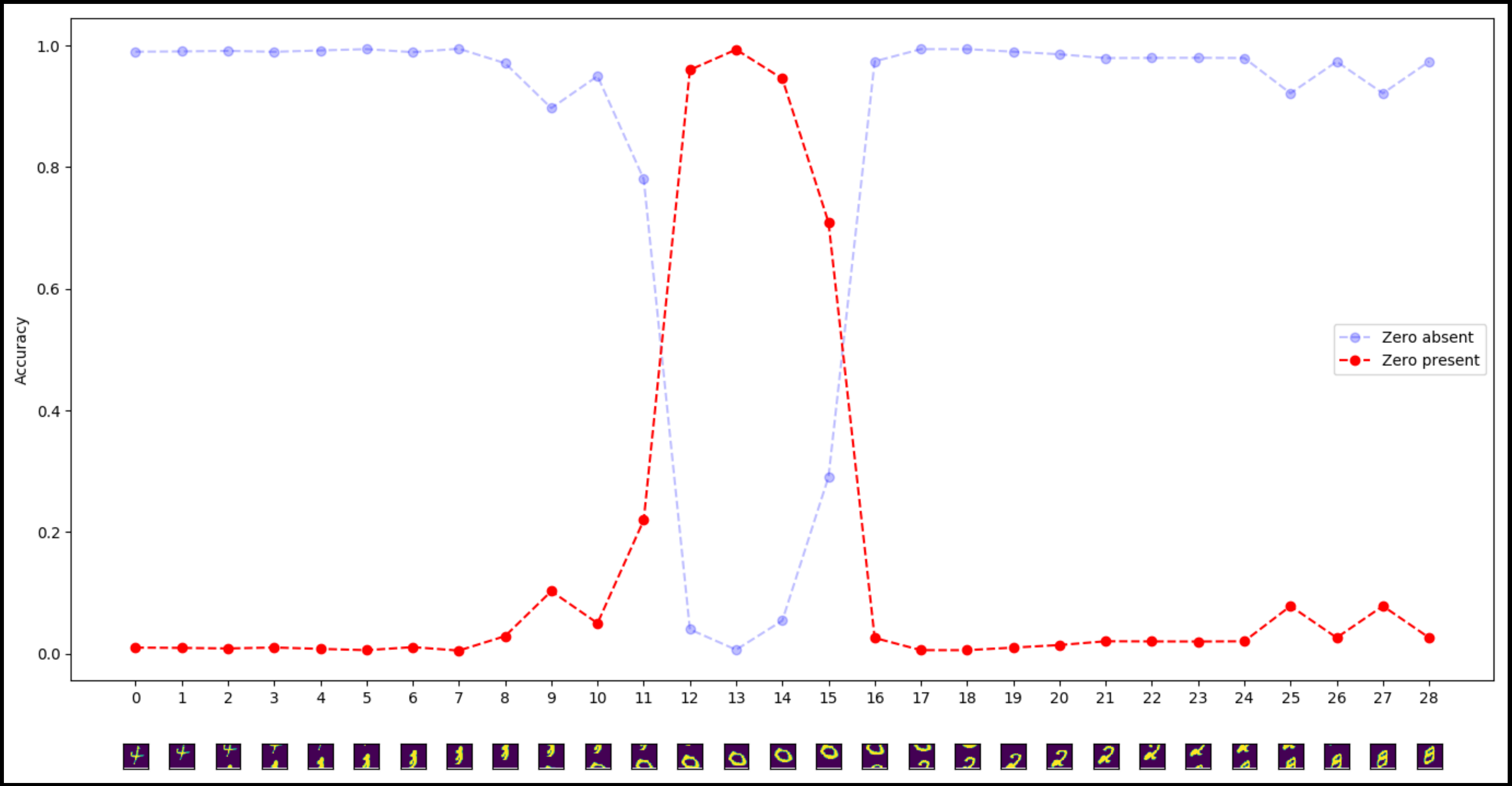
Theorem: In $\log n$ iterations, the true positive set

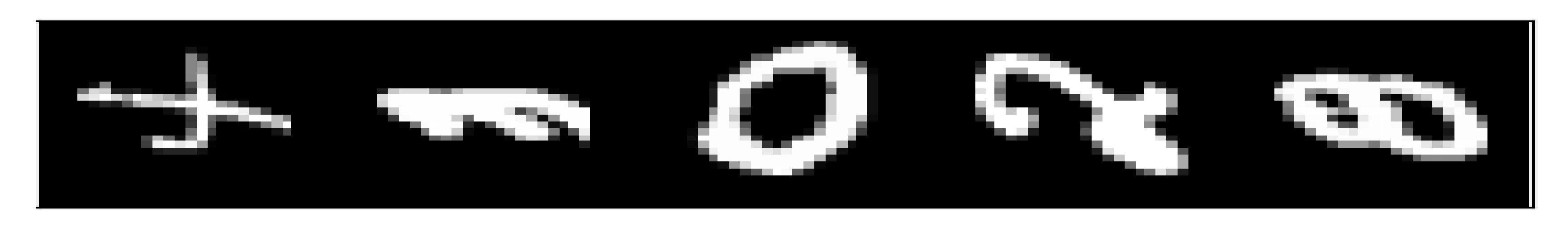
$$S_* = \{(i,\tau), \bar{y}^P_{i,\tau} = +1\}$$

will be recovered exactly, with high probability.

Problem setting is non-homogeneous and interesting. First such result!
Details in paper.

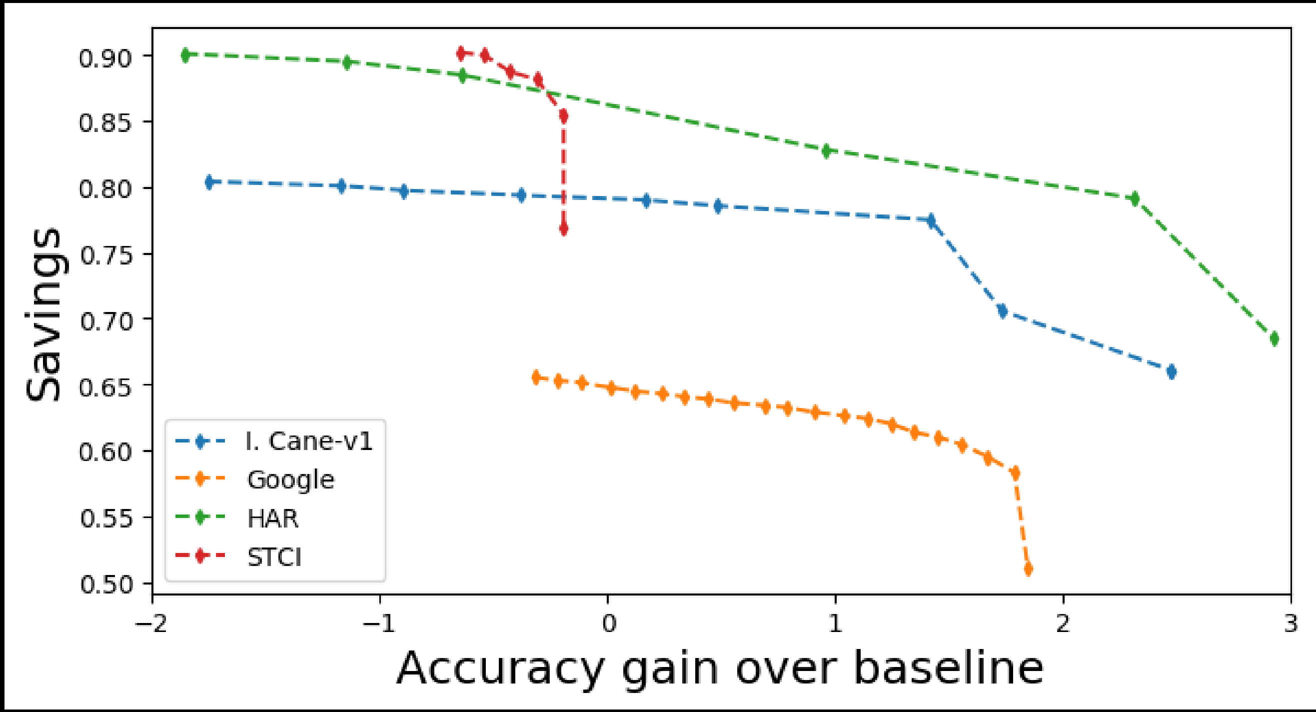
EMI-RNN: Empirical Results





EMI-RNN: Empirical Results

Dataset	Accuracy Gain (over Baseline LSTMs)	Prediction Time Reduction
HAR	0.8%	8x
Sports	2.0%	9x
Google	1.5%	8x
Interactive Cane	1.0%	45x



Deployment on tiny-devices?

- Deployment on Pi0 device: audio keyword detection
  - Total prediction budget: 22.5ms
  - Baseline LSTM prediction time: 226ms (with 91% accuracy)
  - Our method: 14.9ms (with 94% accuracy)

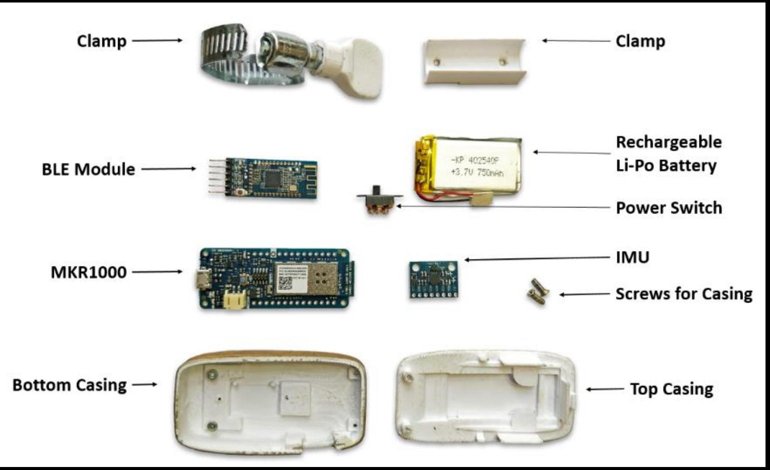
Demos: Algorithms in real-time

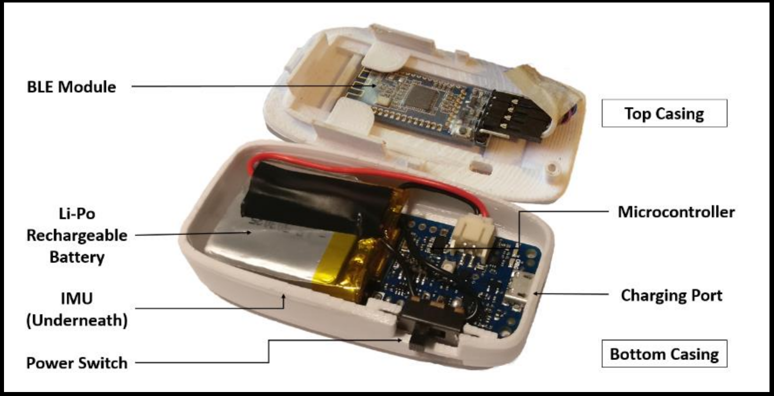
Interactive Cane / GesturePod:

- 5-class gesture recognition on M0+ microcontroller
- 6KB ProtoNN model

Wakeword detection

- Detect “Hey Cortana” on Raspberry Pi0
- Process 800millisecond audioframe in <10ms





Conclusions

- ML for IoT devices might provide many high-impact opportunities

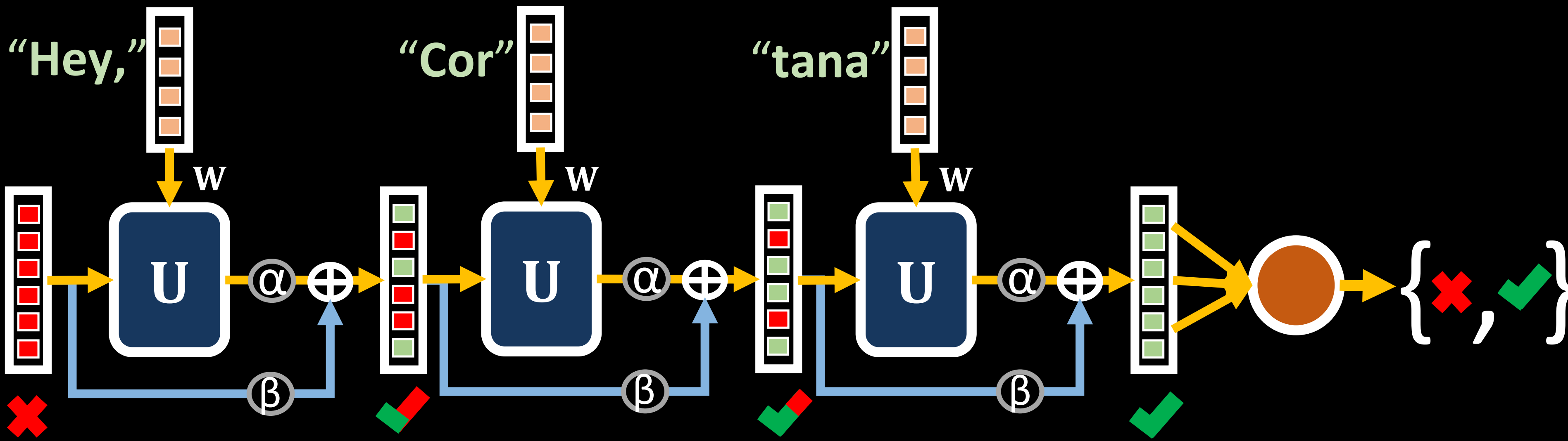
- Microsoft's Edge Machine Learning Library
  - https://github.com/Microsoft/EdgeML

- Bonsai, ProtoNN, FastGRNN & EMI-RNN
  - Fit into a few Kilobytes of memory
  - Make predictions in milliseconds
  - Are energy efficient & extend battery life
  - Have state-of-the-art prediction accuracies

Appendix

FastRNN

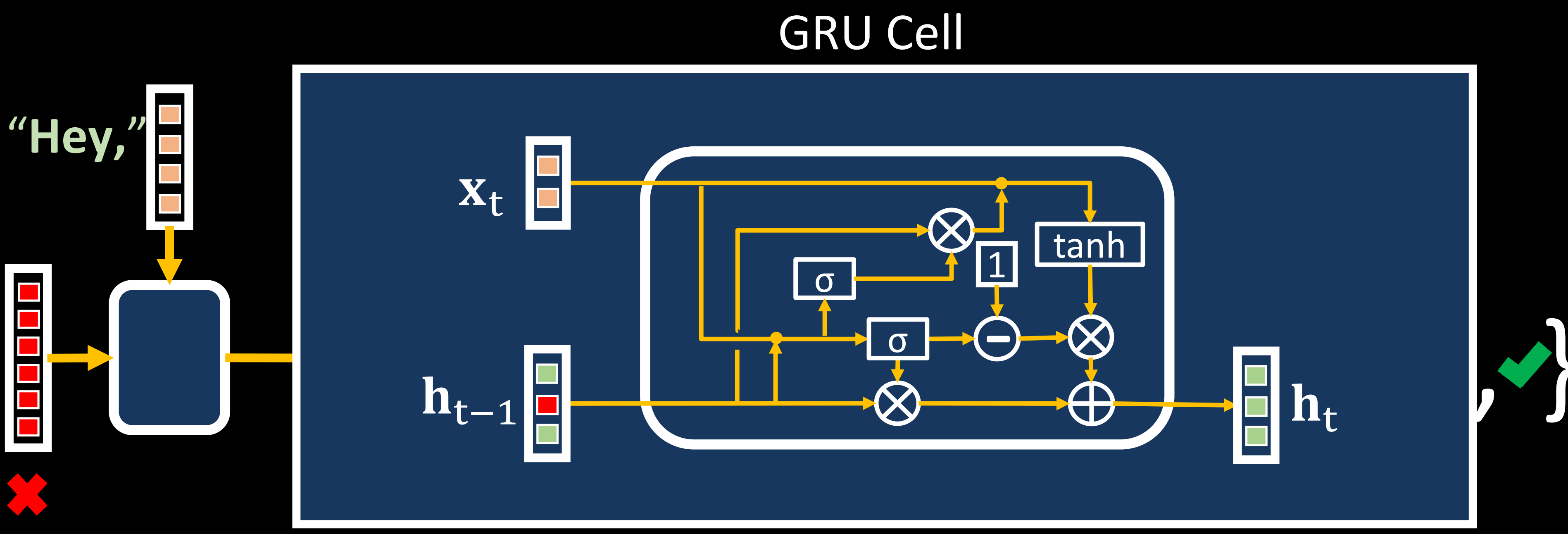
- Provably stable training with 2 additional scalars
- Accuracy: RNN ≪ Unitary RNNs < FastRNN < Gated RNNs



$$\nabla = \ f(\ldots, (\alpha\mathbf{UD} + \beta\mathbf{I})^T = \mathbf{Q}\begin{bmatrix}(1+\varepsilon)^T & & \\ & \ddots & \\ & & (1-\varepsilon')^T\end{bmatrix}\mathbf{Q}^\top, \ldots)$$

Gated RNNs – LSTM, GRU, …

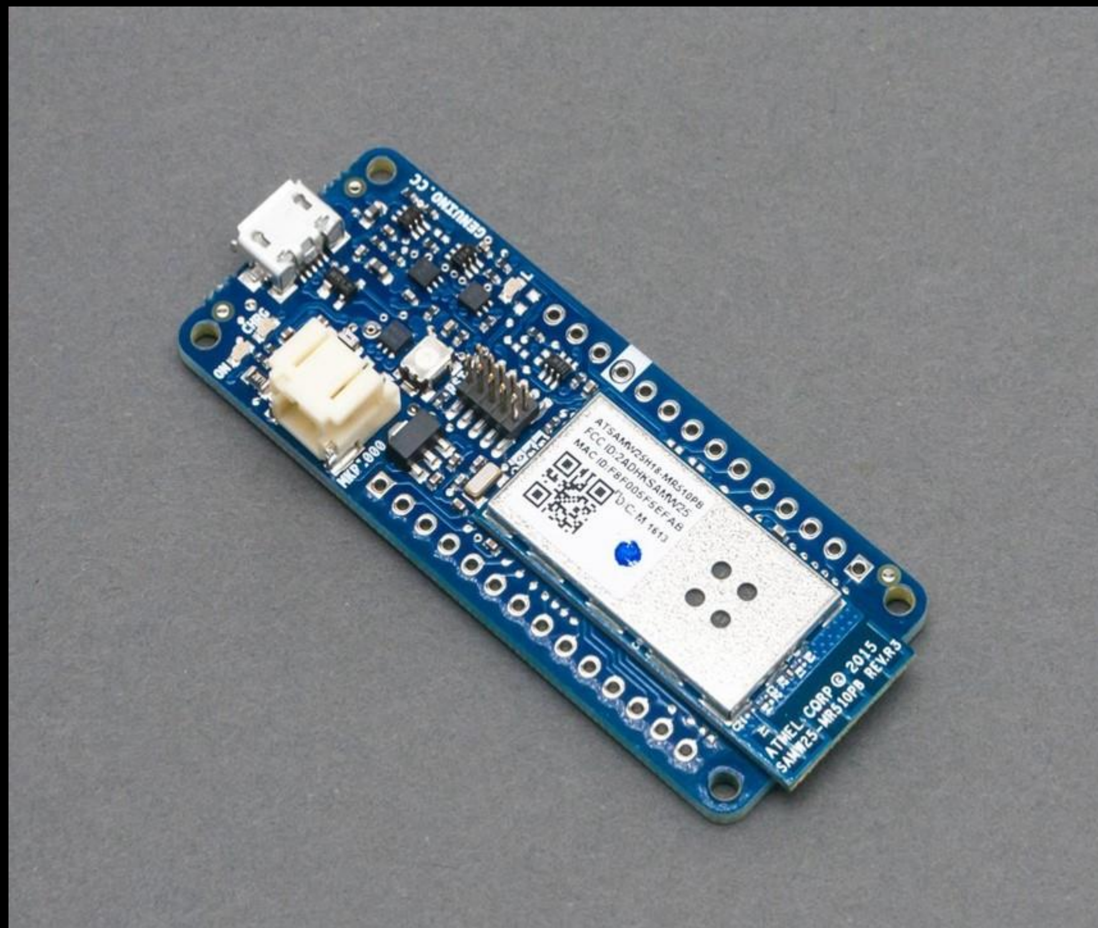
- Add extra parameters to stabilize training
- Infeasible edge prediction due to large model size
- Have intuitive explanations but no formal guarantees



Dataset Statistics

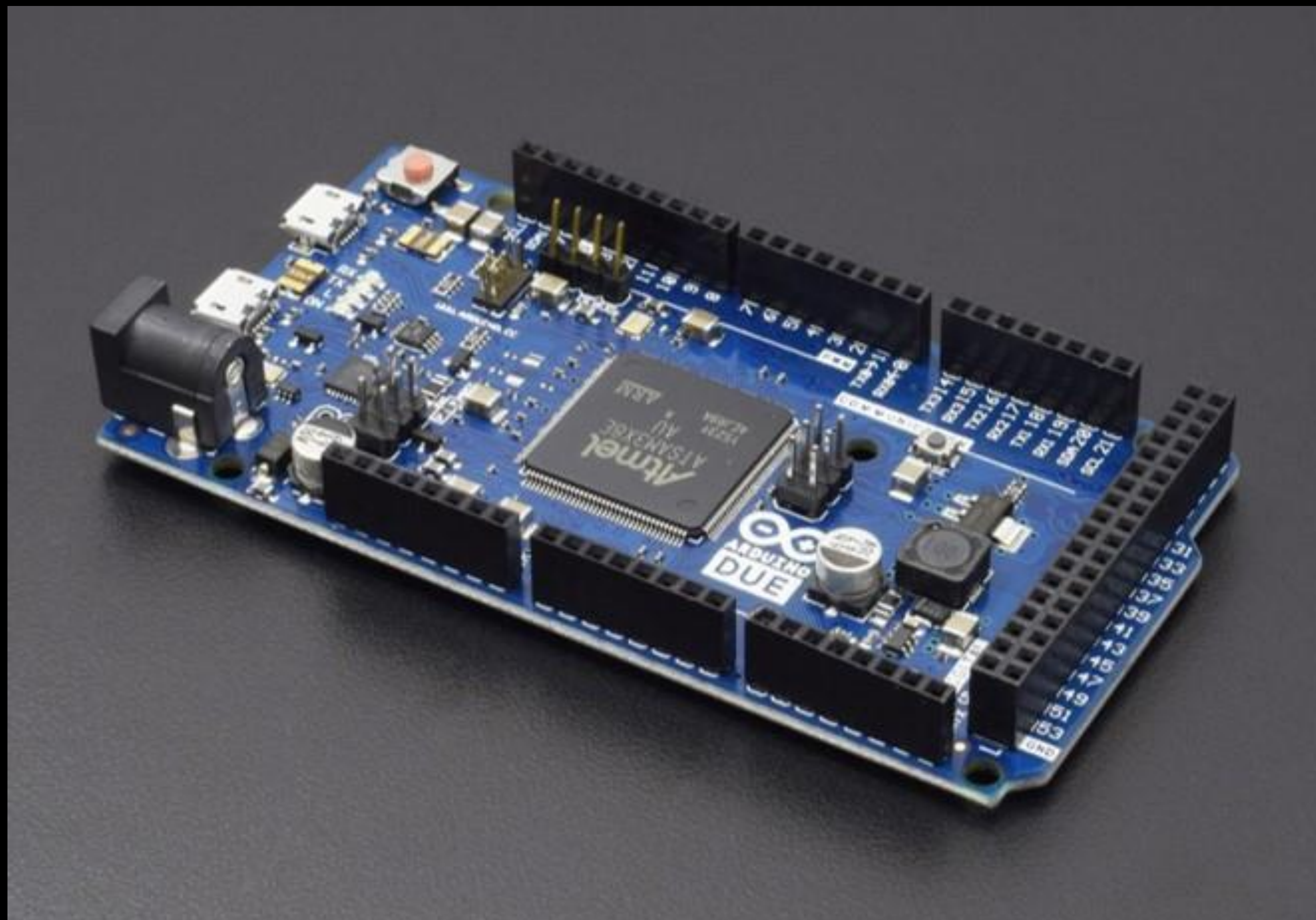
	Dataset	# Train	# Features	# Time Steps	# Test
Speech	Google-12	22,246	3,168	99	3,081
Speech	Google-30	51,088	3,168	99	6,835
Speech	Wakeword-2	195,800	5,184	162	83,915
NLP	Yelp-5	500,000	38,400	300	500,000
NLP	PTB-10000	929,589	---	300	82,430
Activity	HAR-2	7,352	1,152	128	2,947
Activity	DSA-19	4,560	5,625	125	4,560
Image	Pixel-MNIST-10	60,000	784	784	10,000

The Arduino MKR1000



32 bit SAMD21 Cortex-M0+ Processor at 48 MHz
with 32 KB RAM & 256 KB read only Flash

The Arduino Due



32 bit AT91SAM3X8E Processor at 84 MHz
with 96 KB RAM & 512 KB read only Flash

RNN gradients

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

$$\frac{\partial L}{\partial \mathbf{U}} = \sum_{t=0}^{T} \mathbf{D}_t \left(\prod_{k=t}^{T-1} \mathbf{U}^\intercal \mathbf{D}_{k+1} \right) (\nabla_{\mathbf{h}_T} L) \mathbf{h}_{t-1}^\intercal$$

$$\frac{\partial L}{\partial \mathbf{W}} = \sum_{t=0}^{T} \mathbf{D}_t \left(\prod_{k=t}^{T-1} \mathbf{U}^\intercal \mathbf{D}_{k+1} \right) (\nabla_{\mathbf{h}_T} L) \mathbf{x}_t^\intercal$$

$$\mathbf{D}_k = \mathrm{grad}(\mathbf{h}_k) \qquad \mathbf{D}_k = \mathrm{diag}(\sigma'(\mathbf{W}\mathbf{x}_k + \mathbf{U}\mathbf{h}_{k-1} + \mathbf{b}))$$

FastRNN gradients

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

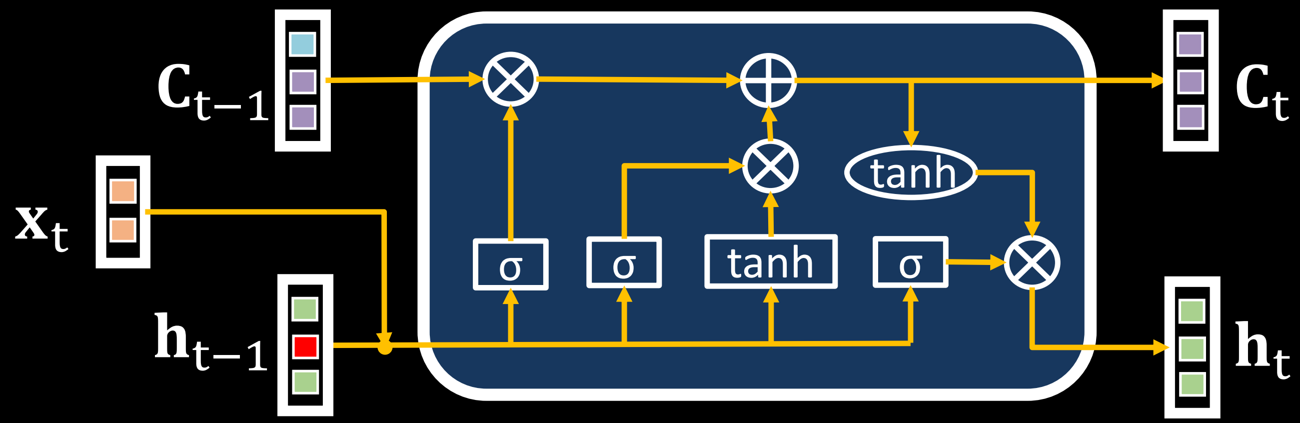
$$\frac{\partial L}{\partial \mathbf{U}} = \alpha \sum_{t=0}^{T} \mathbf{D}_t \left(\prod_{k=t}^{T-1} (\alpha \mathbf{U}^\top \mathbf{D}_{k+1} + \beta \mathbf{I}) \right) (\nabla_{\mathbf{h}_T} L) \mathbf{h}_{t-1}^\top$$

$$\frac{\partial L}{\partial \mathbf{W}} = \alpha \sum_{t=0}^{T} \mathbf{D}_t \left(\prod_{k=t}^{T-1} (\alpha \mathbf{U}^\top \mathbf{D}_{k+1} + \beta \mathbf{I}) \right) (\nabla_{\mathbf{h}_T} L) \mathbf{x}_t^\top$$

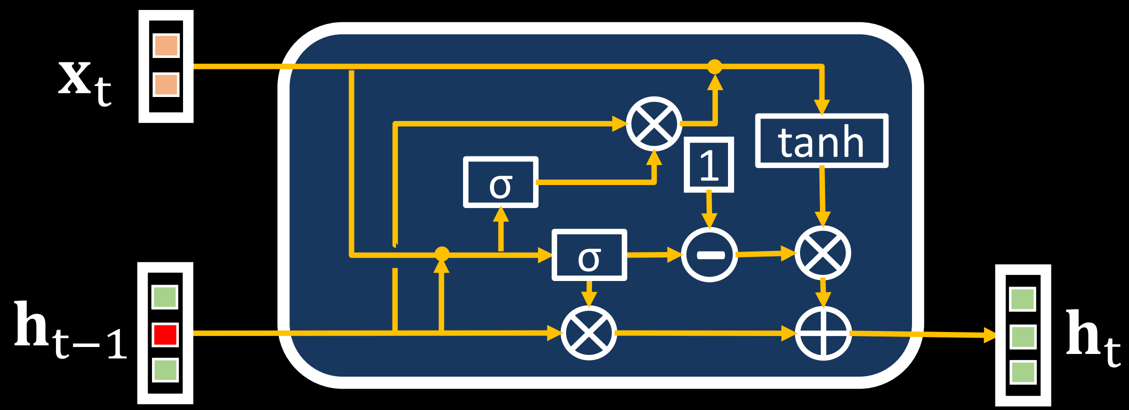
$$\mathbf{D}_k = \mathrm{grad}(\mathbf{h}_k) \qquad \mathbf{D}_k = \mathrm{diag}(\sigma'(\mathbf{W}\mathbf{x}_k + \mathbf{U}\mathbf{h}_{k-1} + \mathbf{b}))$$

More Block Diagrams

LSTM ≡

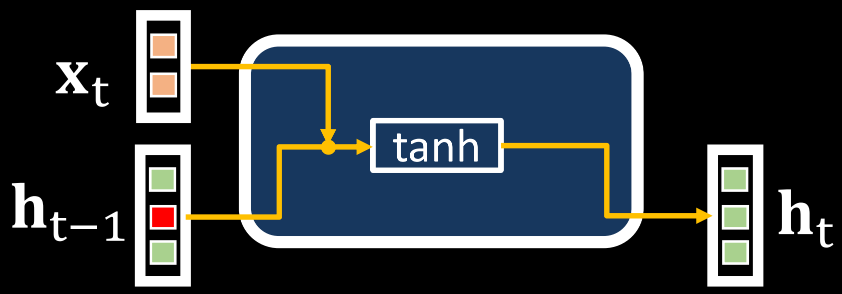


GRU ≡

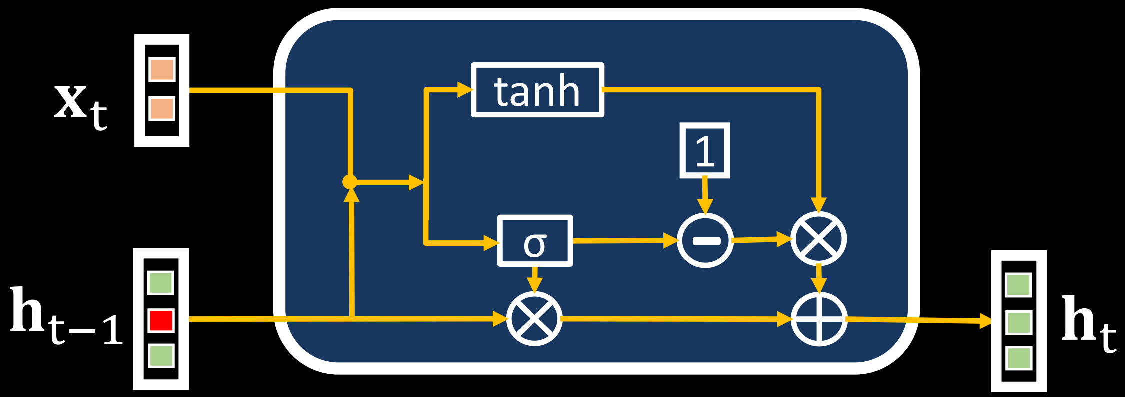


More Block Diagrams

RNN $\equiv$

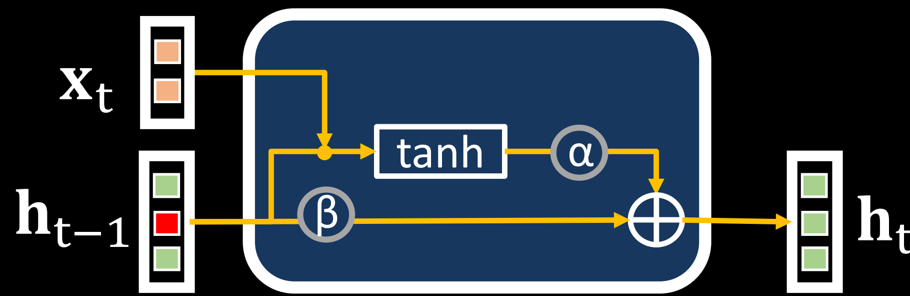


UGRNN $\equiv$

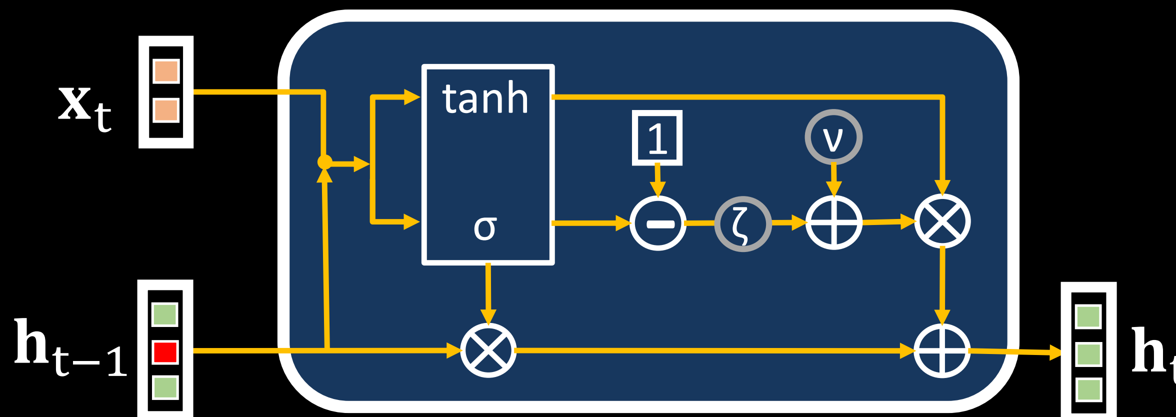


More Block Diagrams

FastRNN $\equiv$

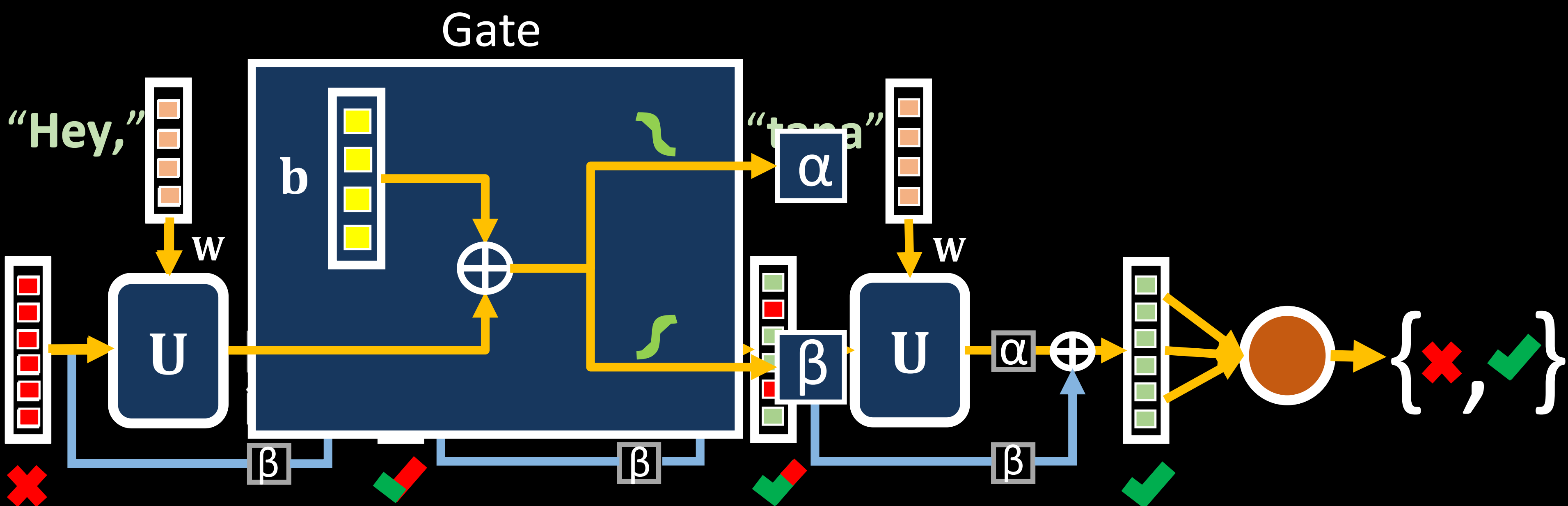


FastGRNN $\equiv$

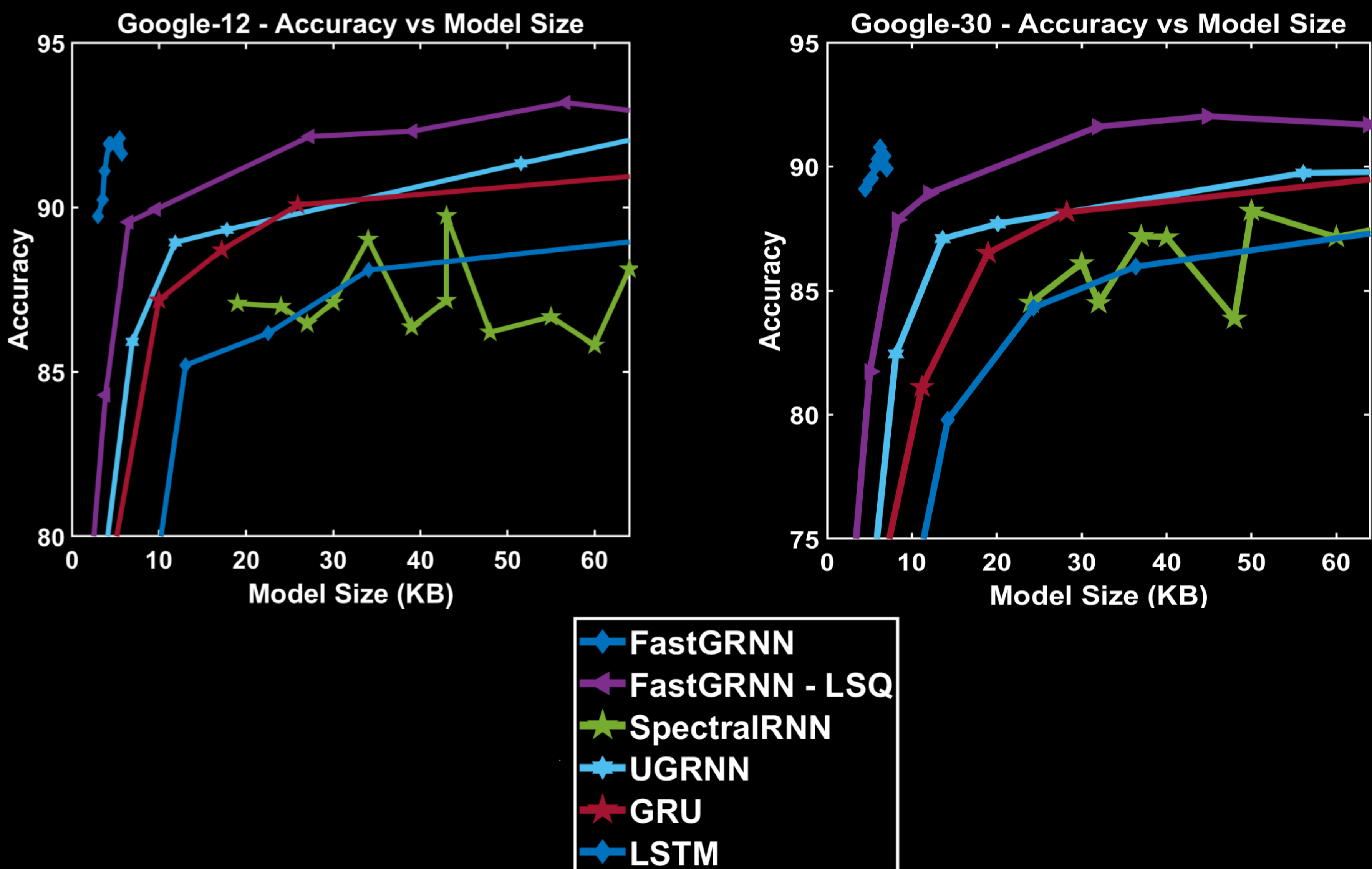


FastGRNN

- Extend α & β from scalars to vector gates
- Accuracy: RNN ≪ Unitary RNNs < Gated RNNs ≈ FastGRNN



Prediction Accuracy vs Model Size



Effects of Compression (LSQ)