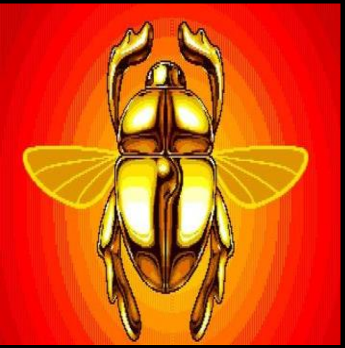


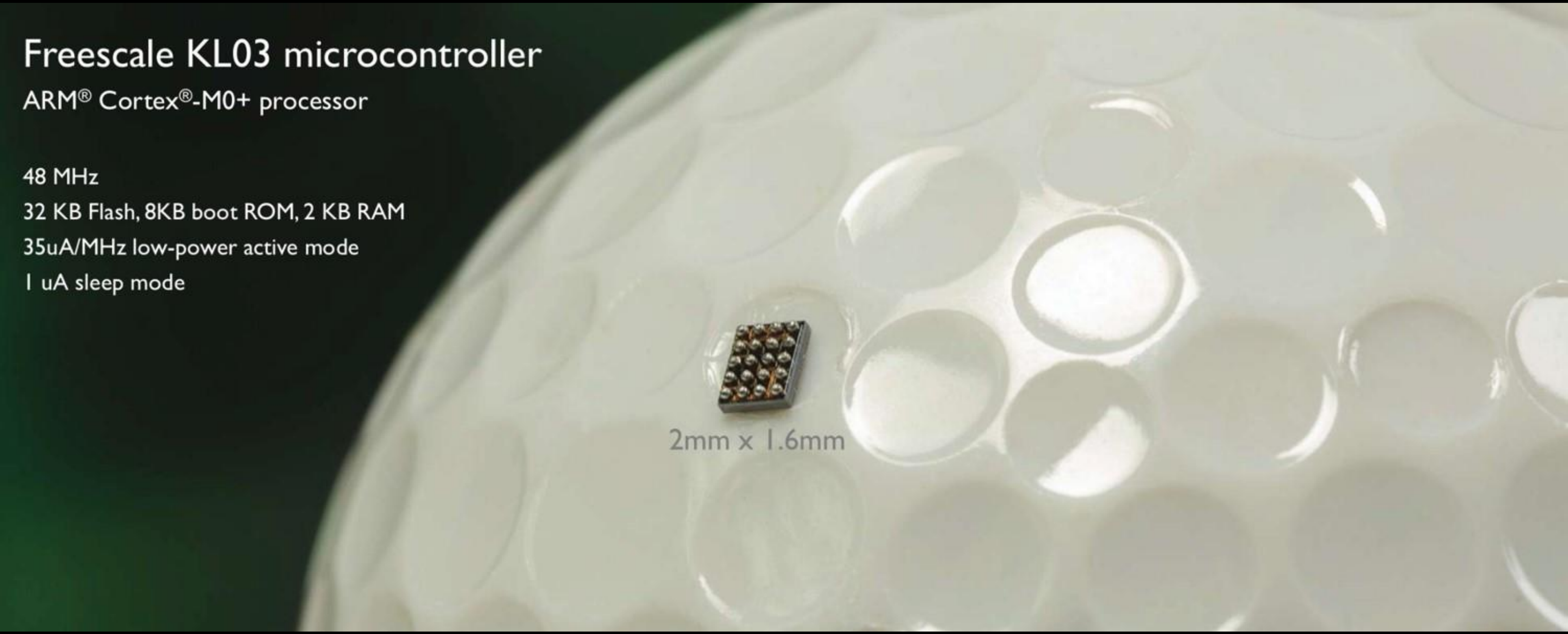

The Edge of Machine Learning

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

Aditya Kusupati & Don Dennis
Microsoft Research India

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

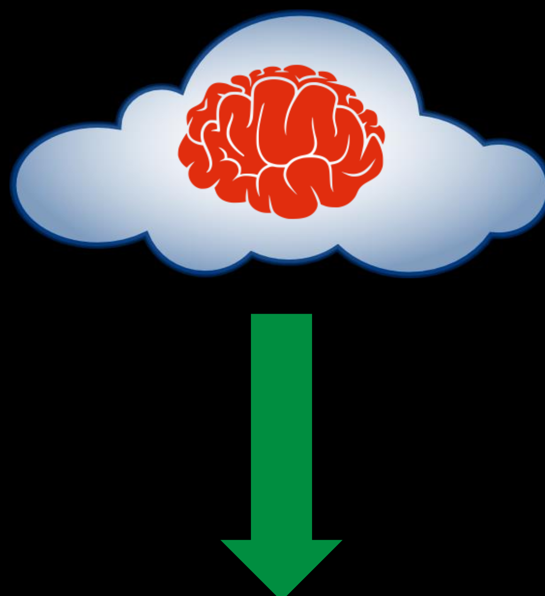
Resource-constrained IoT Devices



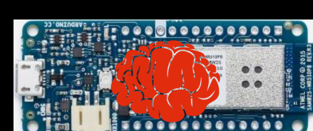
ARM Cortex M0+ at 48 MHz & 35 μA/MHz
with 2 KB RAM & 32 KB read only Flash

Edge Machine Learning – Objectives

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



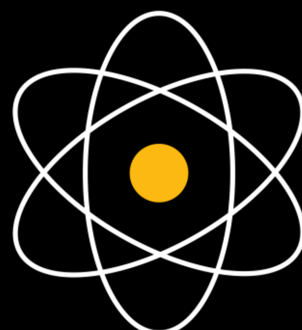
ARM Cortex M0+



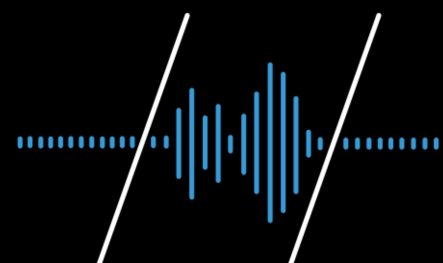
Microsoft’s EdgeML Library

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

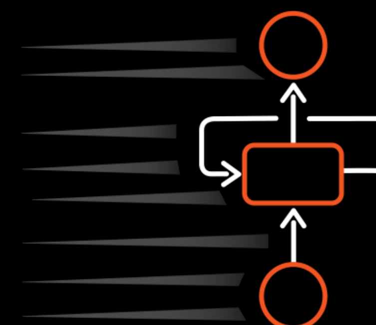
Bonsai



ProtoNN



EMI-RNN



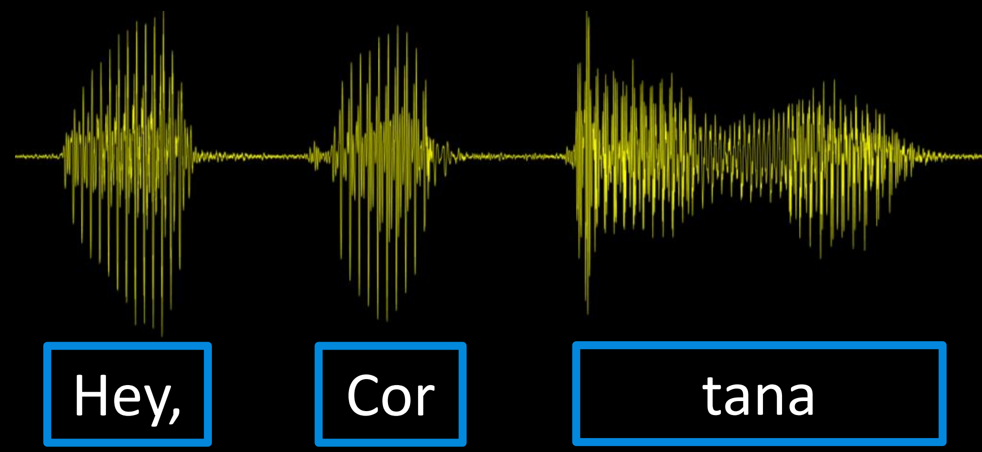
FastGRNN

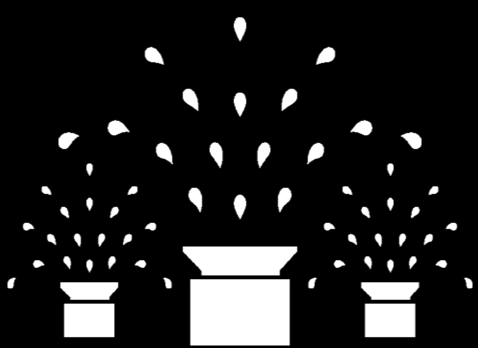
https://github.com/Microsoft/EdgeML

Time Series

- Time series are the most frequently occurring types of signals found in the IoT domain

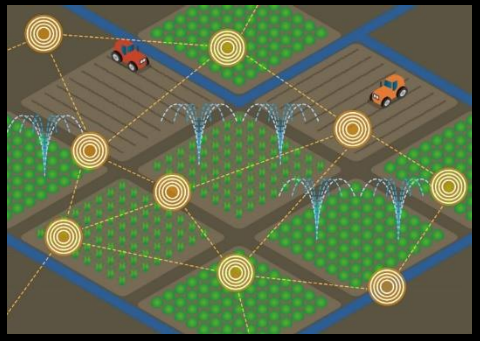
"Hey, Cortana"



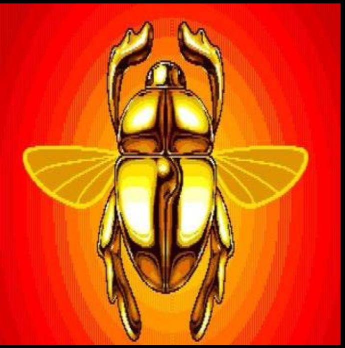


Sprinklers

Soil moisture during a day



Smart farm



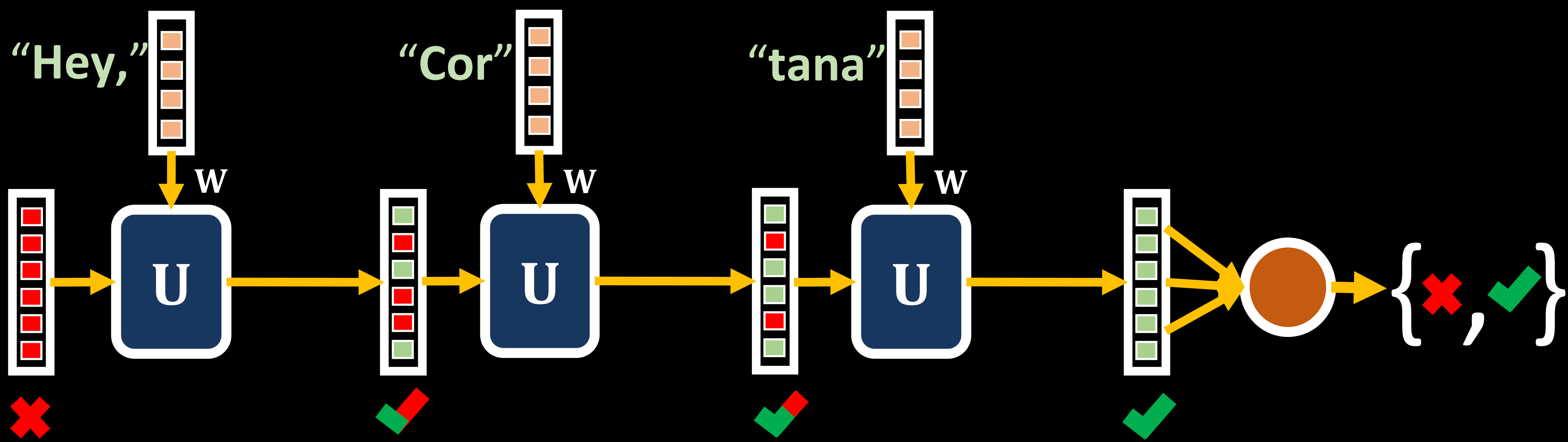

FastGRNN

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)

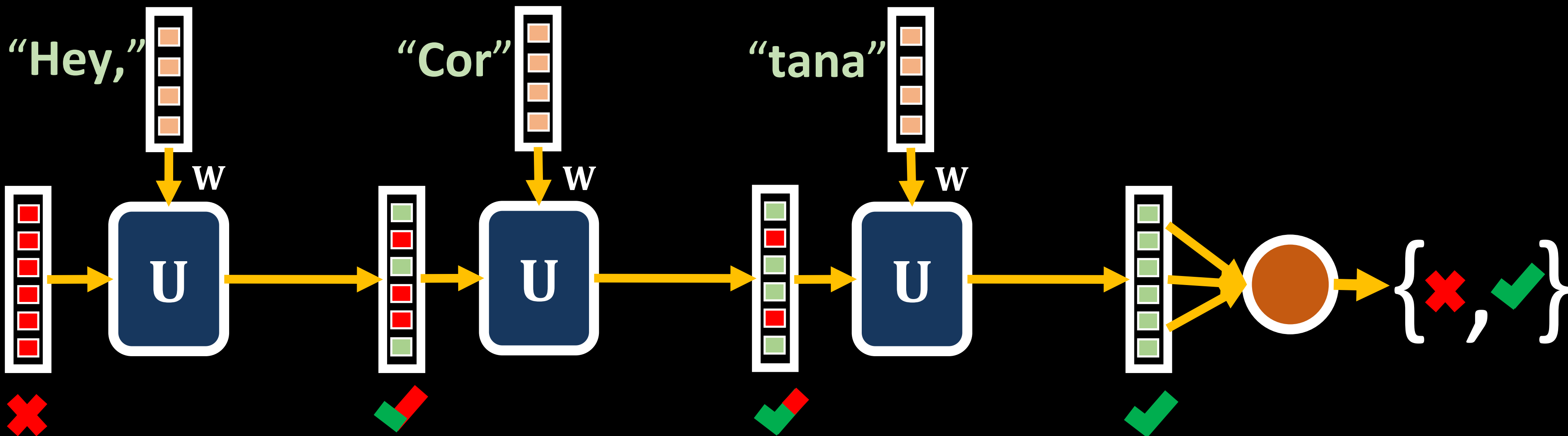
- State-of-the-art for analyzing sequences & time series
- Training is unstable due to exploding & vanishing gradients



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

Recurrent Neural Networks (RNNs)

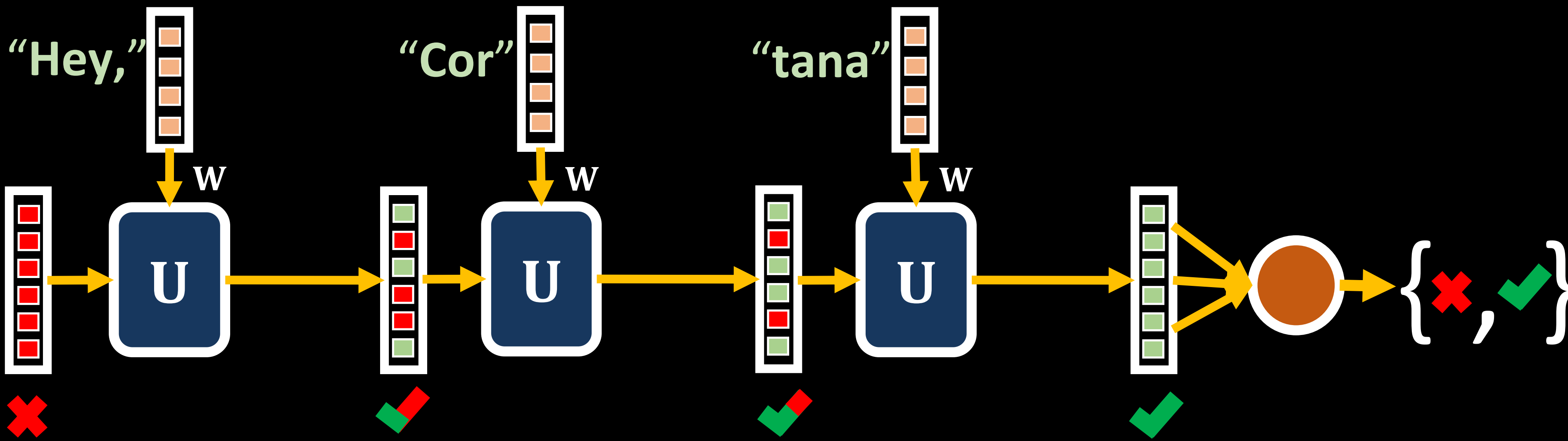
- State-of-the-art for analyzing sequences & time series
- Training is unstable due to exploding & vanishing gradients



$$\nabla = f\left(\ldots, \mathbf{U}^T = \mathbf{Q} \begin{bmatrix} 1.2^{100} & & \\ & \ddots & \\ & & 0.8^{100} \end{bmatrix} \mathbf{Q}^\top, \ldots\right)$$

Unitary RNNs – uRNN, SpectralRNN, …

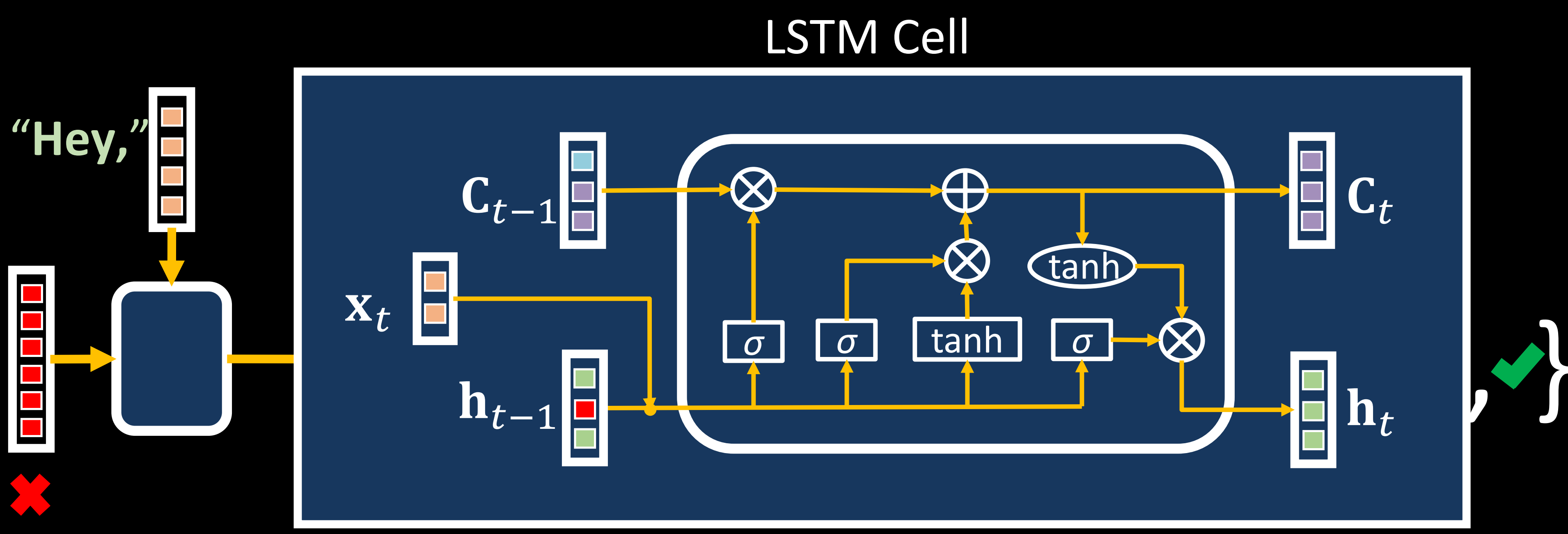
- Unitary RNNs force all the eigenvalues of $\mathbf{U}$ to be ≈ 1
- Unfortunately, they are expensive to train & lack accuracy



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

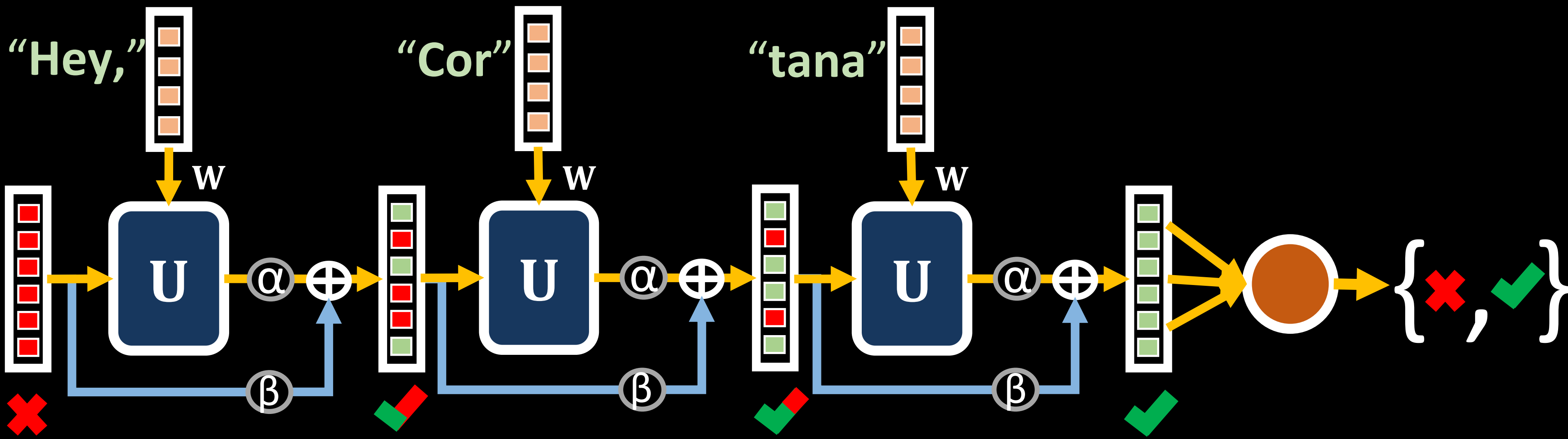
Gated RNNs – LSTM, GRU, …

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



FastRNN

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



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

RNN gradients

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

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

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

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

FastRNN gradients

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

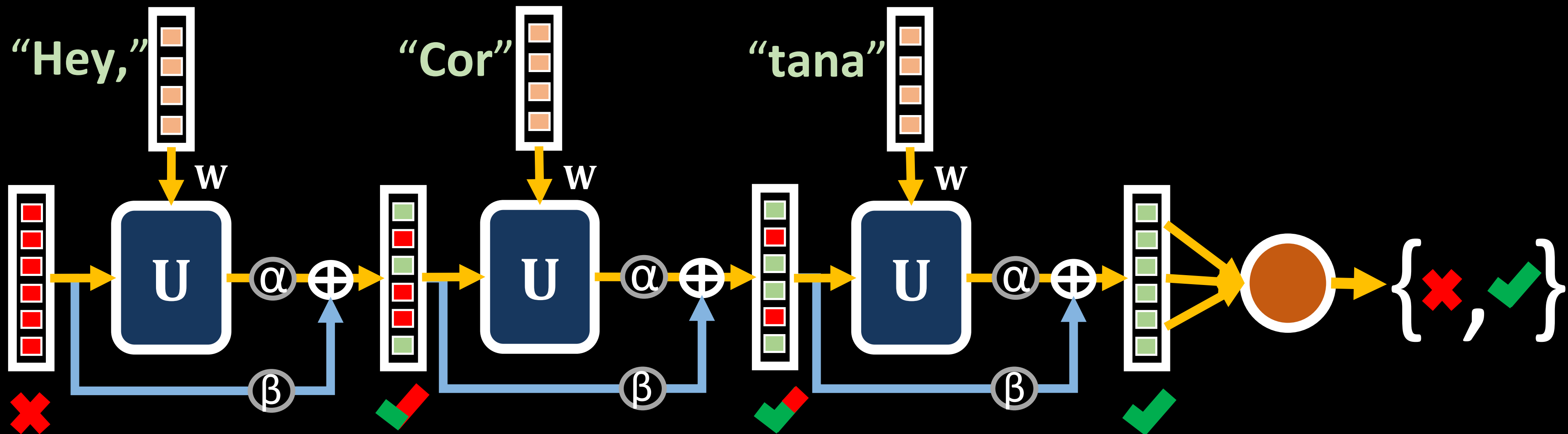
$$\frac{\partial L}{\partial \mathbf{U}} = \alpha \sum_{t=0}^{T} \mathbf{D}_t \left(\prod_{k=t}^{T-1} (\alpha \mathbf{U}^\top \mathbf{D}_{k+1} + \beta \mathbf{I}) \right) (\nabla_{\mathbf{h}_T} L) \mathbf{h}_{t-1}^\top$$

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

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

FastRNN

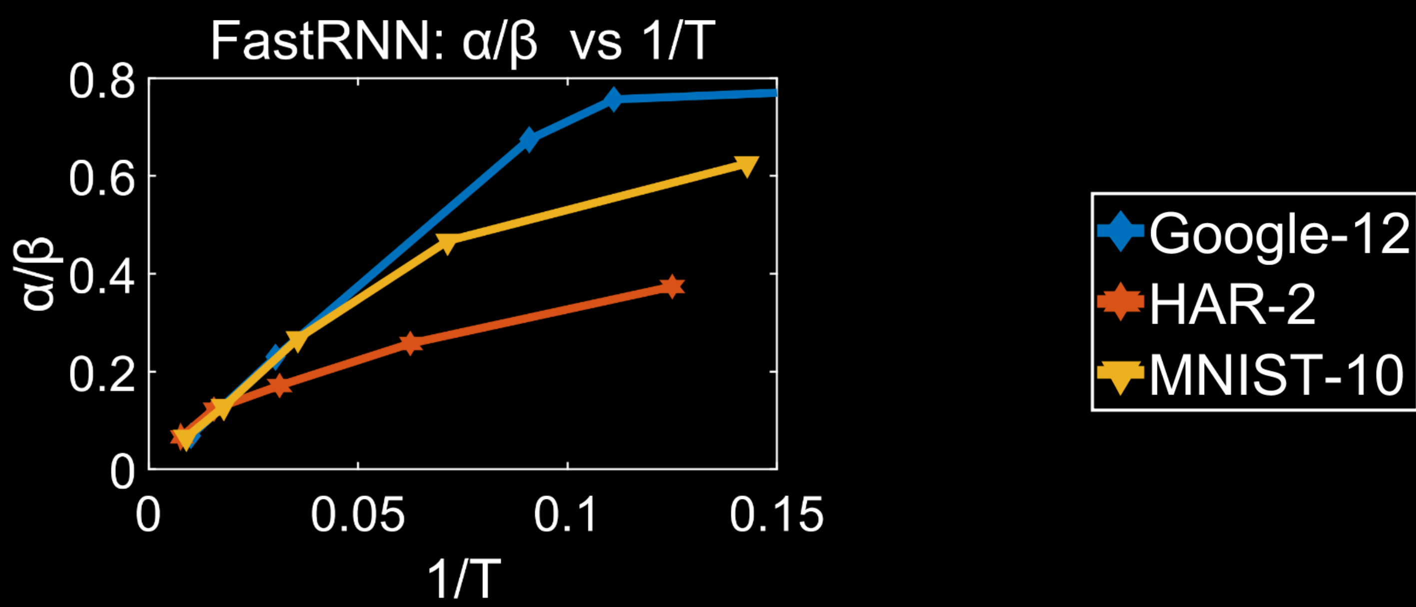
- Provably stable training with 2 additional scalars
- Accuracy: RNN ≪ Unitary RNNs < FastRNN < Gated RNNs



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

Theoretical Analysis of FastRNN

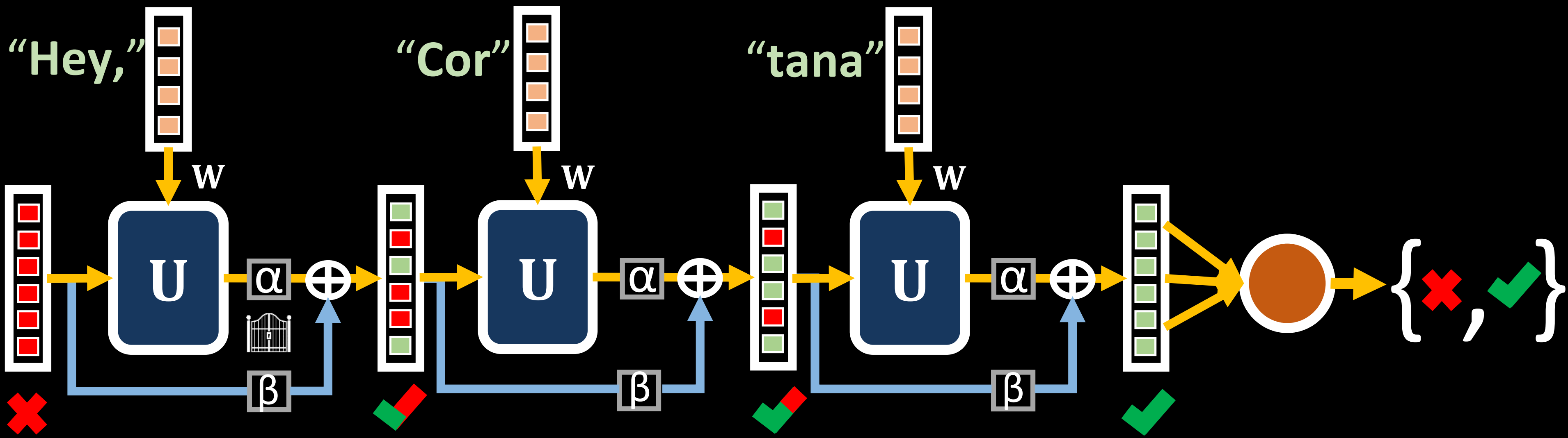
- Convergence & generalization error bounds
  - FastRNN is independent of T when $\alpha/\beta \approx O(1/T)$
  - A similar analysis reveals that an RNN is exponential in T



- FastRNN converges to within ϵ of a stationary point in $O(1/_{\epsilon^2})$ SGD iterations and its generalization error is bounded by $O(\frac{1}{\sqrt{n}})$

FastGRNN

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

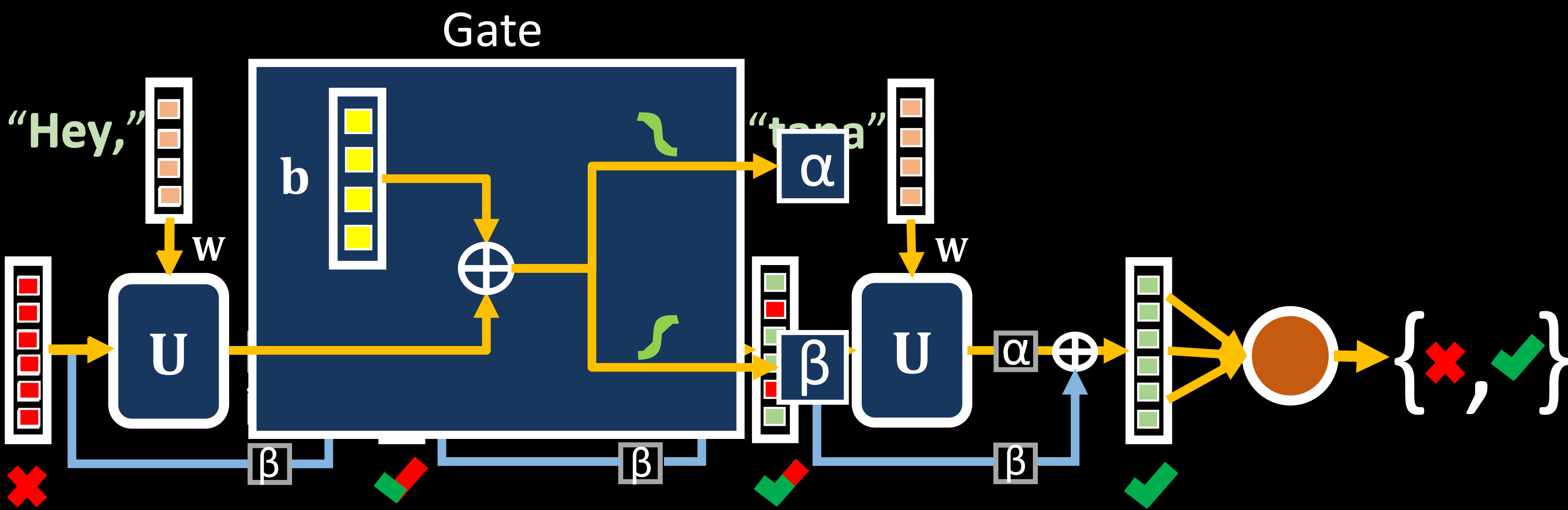


$$\boldsymbol{\beta}_t = \sigma_\beta(\mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1} + \mathbf{b}_\beta); \widetilde{\mathbf{h}_t} = \sigma_h(\mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1} + \mathbf{b}_h)$$

$$\boldsymbol{\alpha}_t \approx \mathbf{1} - \boldsymbol{\beta}_t; \qquad \mathbf{h}_t = \boldsymbol{\alpha}_t \odot \widetilde{\mathbf{h}_t} + \boldsymbol{\beta}_t \odot \mathbf{h}_{t-1}$$

FastGRNN

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

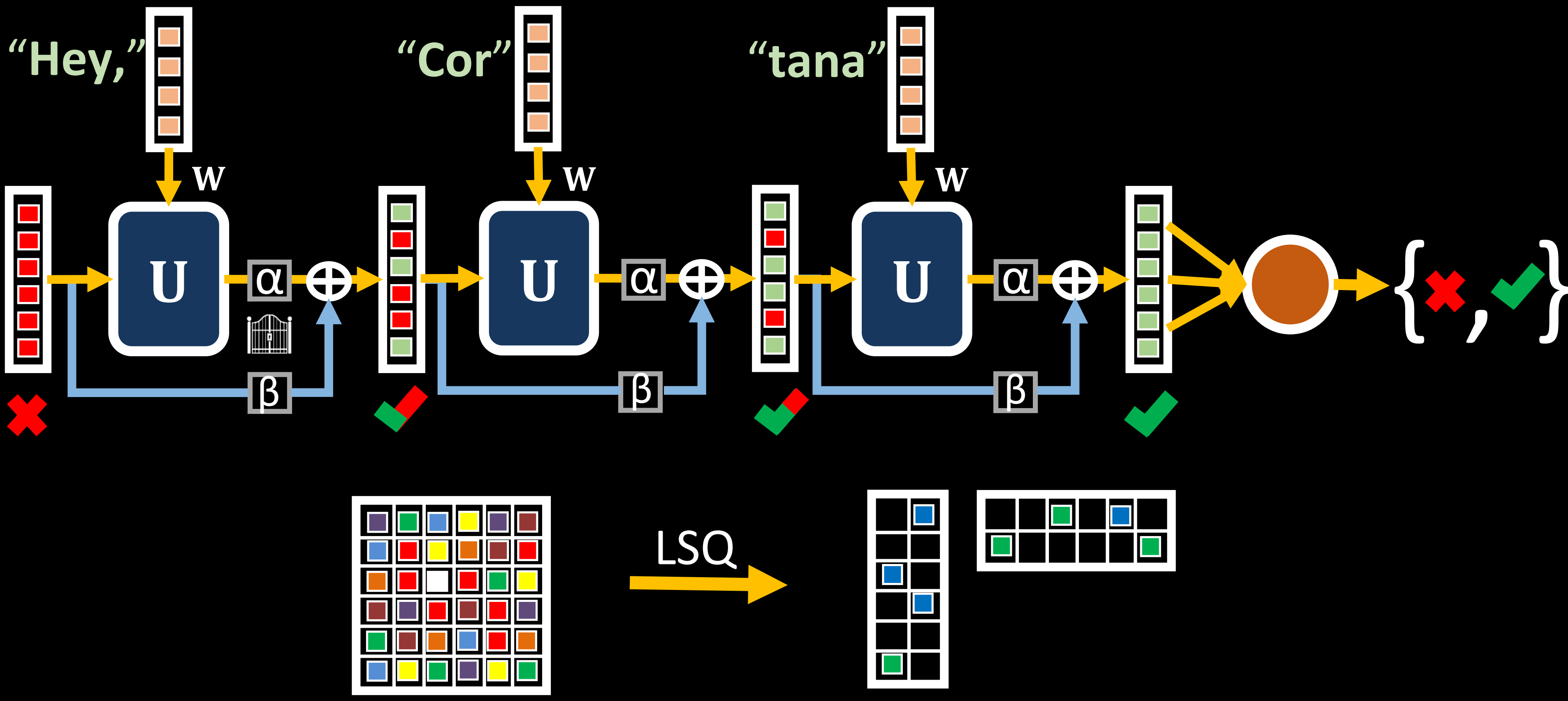


$$\boldsymbol{\beta}_t = \sigma_\beta(\mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1} + \mathbf{b}_\beta); \widetilde{\mathbf{h}_t} = \sigma_h(\mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1} + \mathbf{b}_h)$$

$$\boldsymbol{\alpha}_t = \zeta(1 - \boldsymbol{\beta}_t) + \nu; \qquad \mathbf{h}_t = \boldsymbol{\alpha}_t \odot \widetilde{\mathbf{h}_t} + \boldsymbol{\beta}_t \odot \mathbf{h}_{t-1}$$

FastGRNN

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

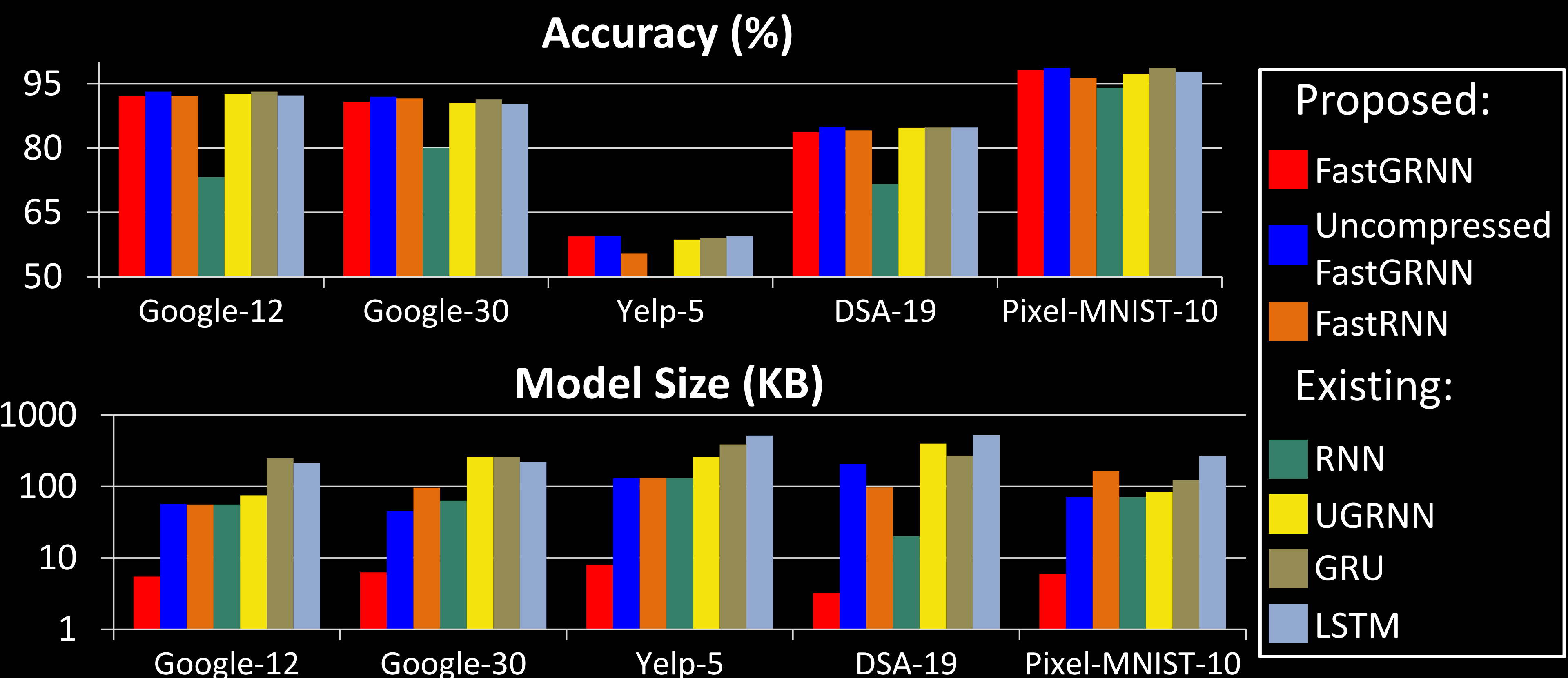


Dataset Statistics

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

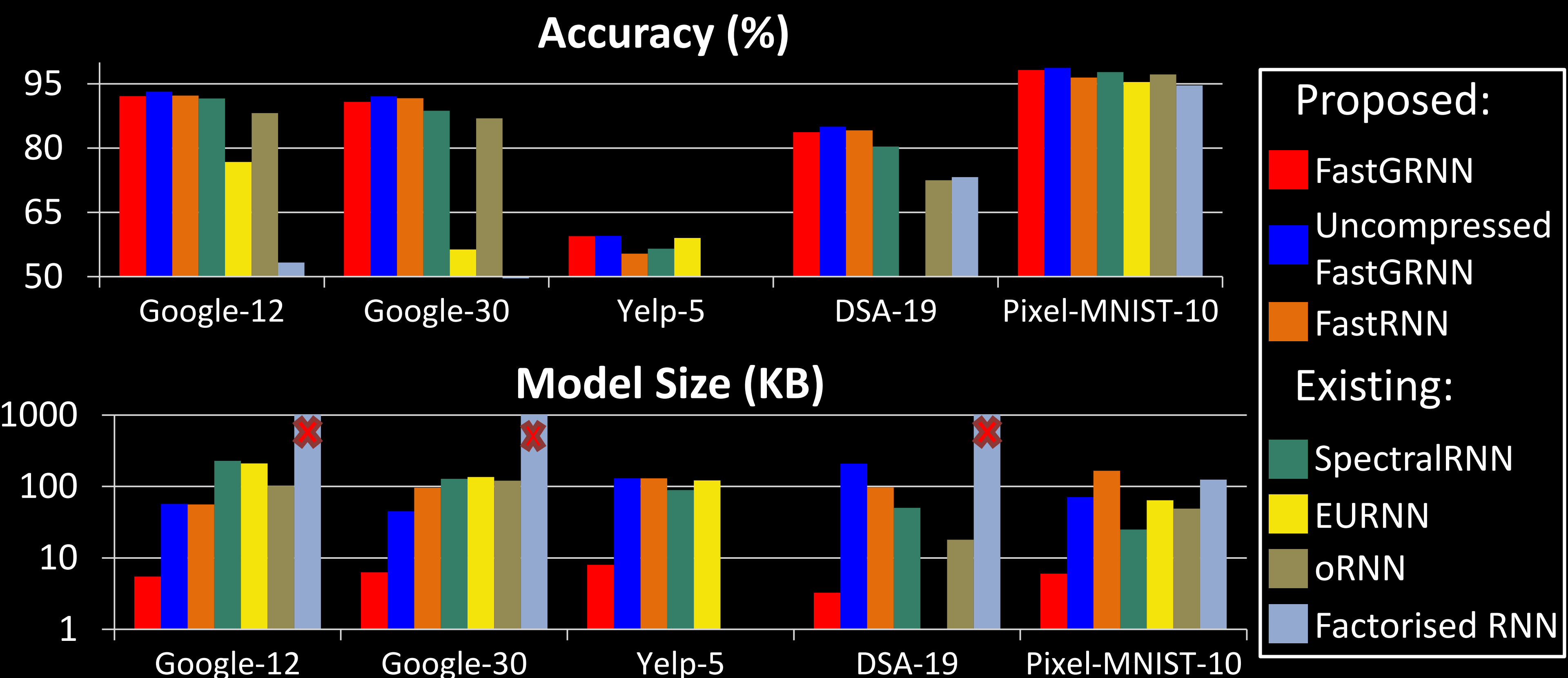
Comparison to Gated Architectures

- Uncompressed FastGRNN is as accurate as a GRU/LSTM
- FastGRNN is almost as accurate as a GRU/LSTM (within 1%)
- FastGRNN is 20-80x smaller than a GRU/LSTM



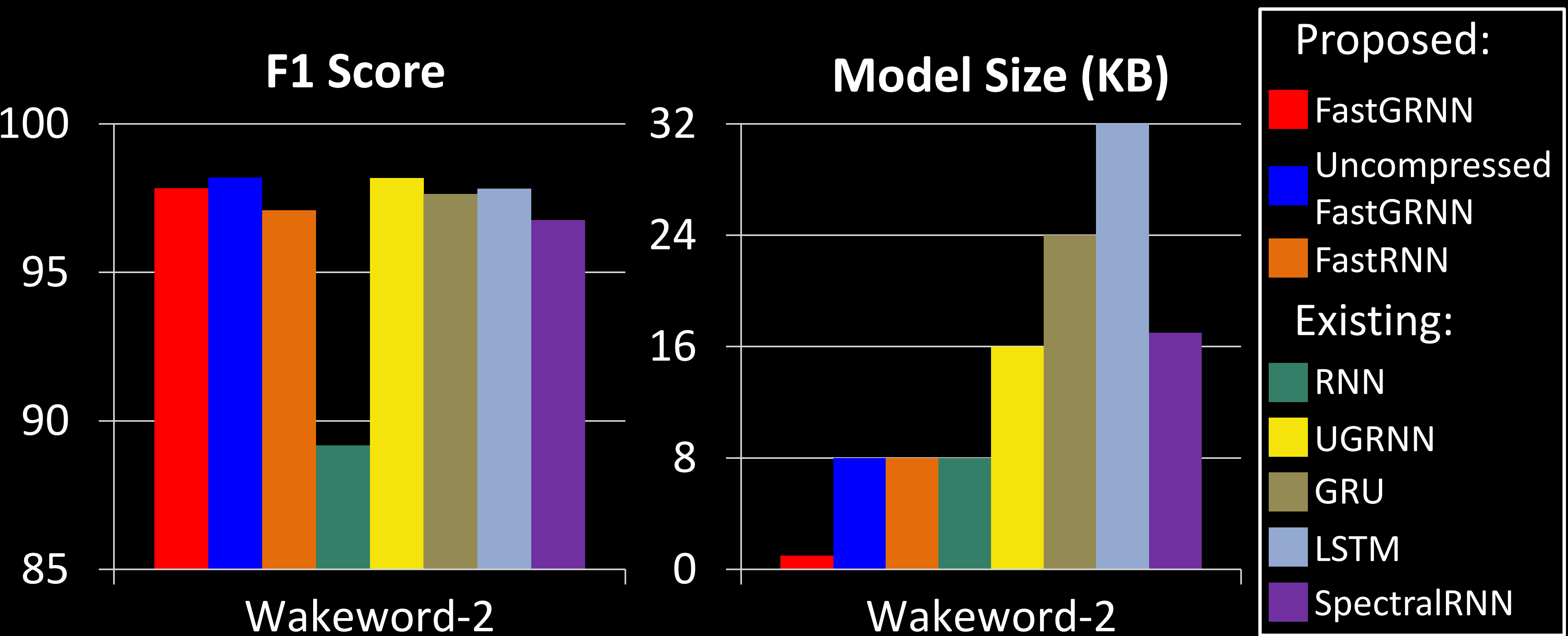
Comparison to Unitary Architectures

- FastRNN outperforms all unitary RNNs on most datasets
- FastGRNN can be 3-5% more accurate
- FastGRNN can be 45x-200x smaller



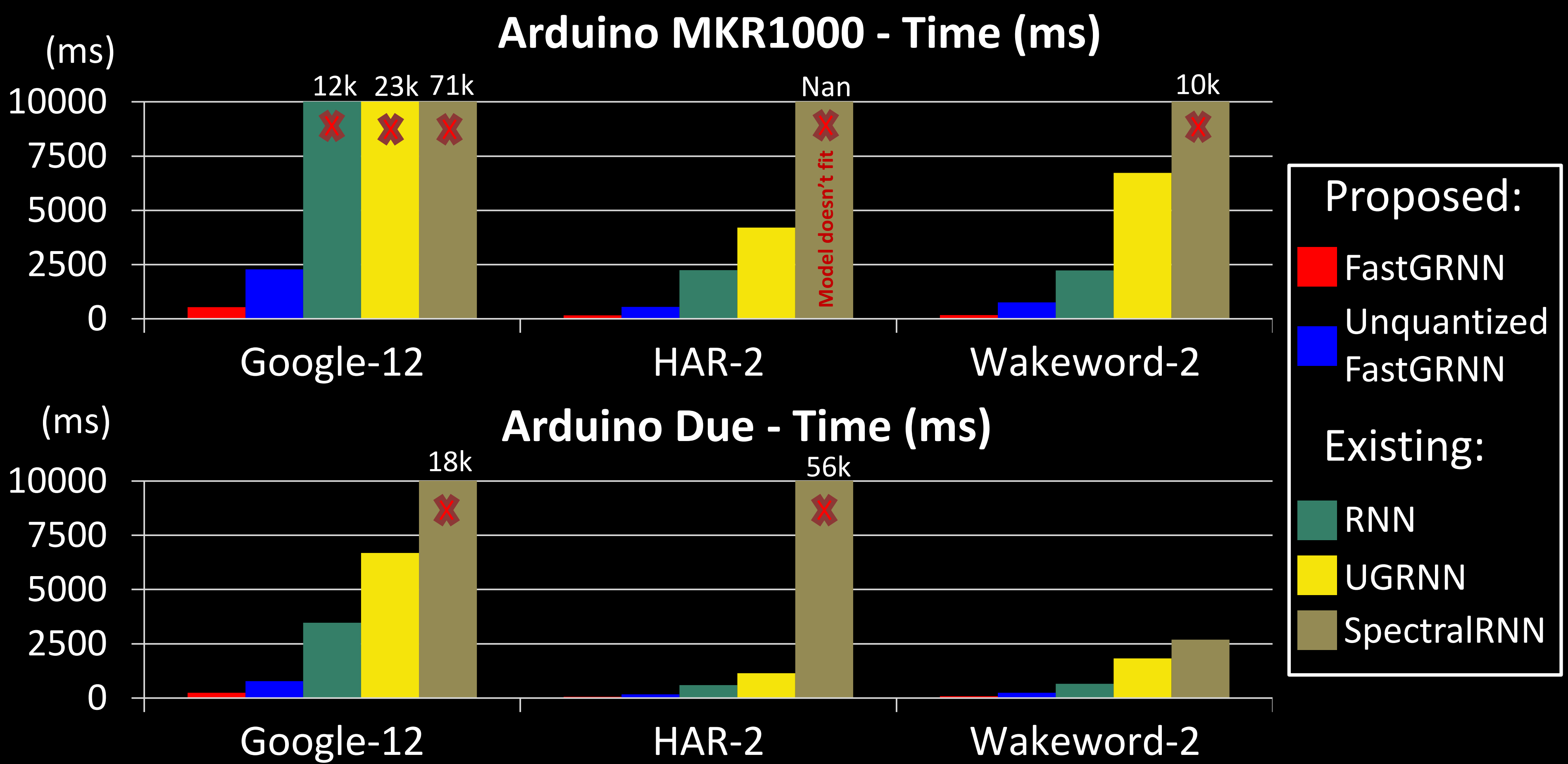
Recognizing "Hey, Cortana" in 1 KB

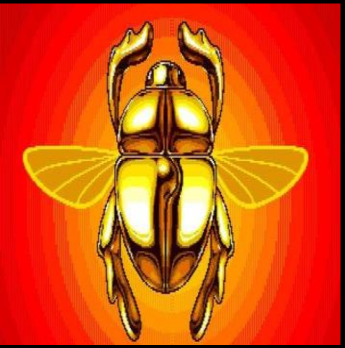
- Uncompressed FastGRNN outperforms state-of-the-art RNNs
- FastGRNN matches state-of-the-art RNN accuracies



Prediction on Edge Devices

- None of the other RNNs fit on an Arduino Uno
- FastGRNN can be 25-132x faster at prediction on the MKR1K



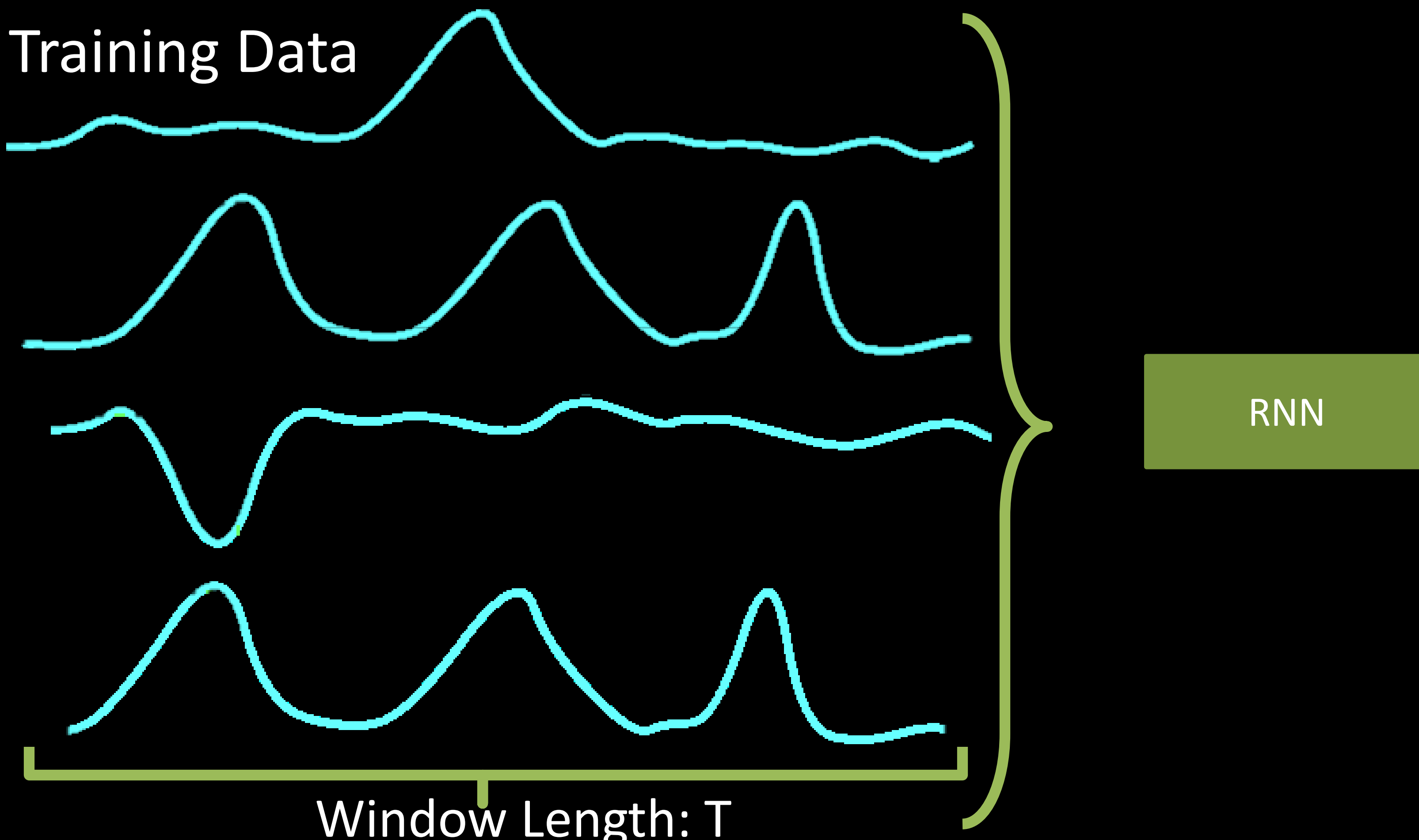



EMIRNN

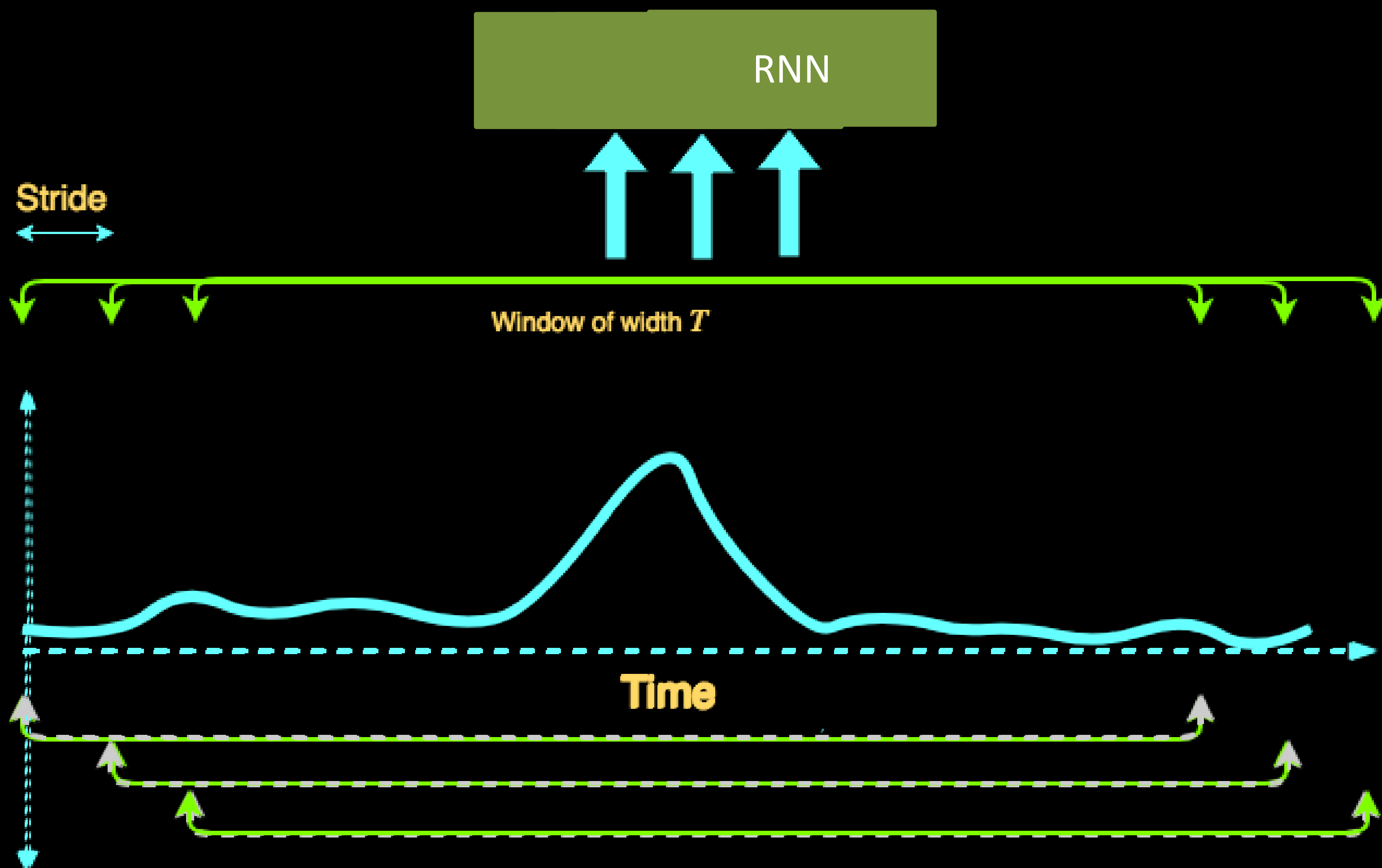
Simultaneous segmentation and early prediction in RNNs

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

Time-series Classification: Training



Time-series Analysis: Sliding Windows

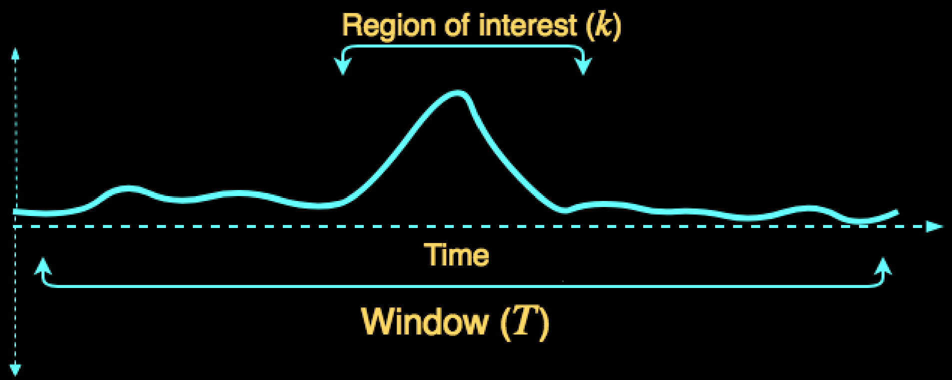


Prediction Cost per Window: O(Window Size)

Time-series Analysis: Sliding Windows

- Complicated RNN cell updates
- Running time O(T)
- Information reuse across windows

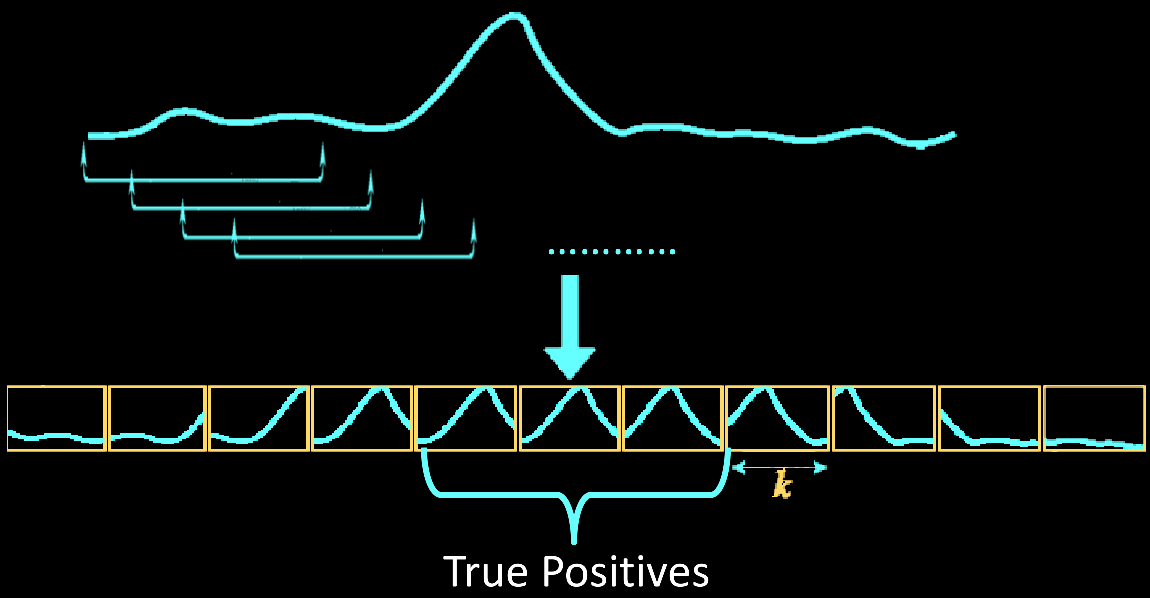
Coarse Training Data



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.

Smaller Windows?

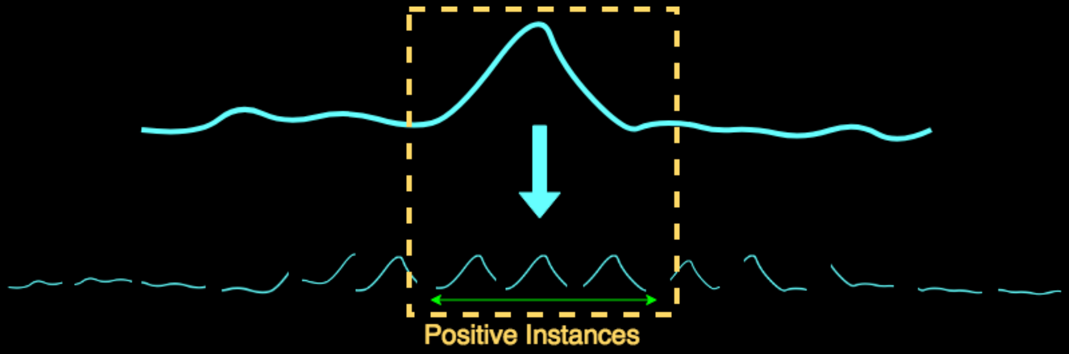


Single Training Datapoint
With label +

Multiple Instances
With label +

- Only a few true positives: several false positives!
- Issue: apriori location of true positives unknown

EMI-RNN: Approach

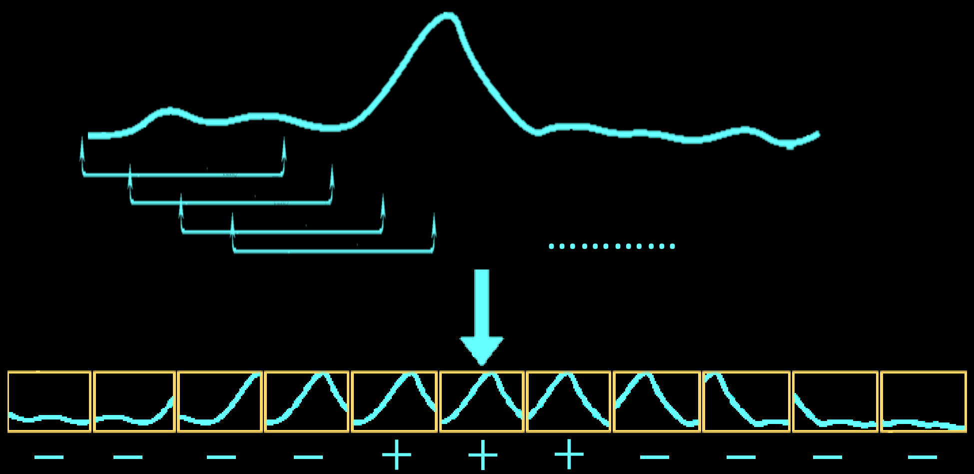


Exploit temporal locality with MIL/Robust learning techniques

Property 1: Positive instances are clustered together.

Property 2: Number of positive instances can be estimated.

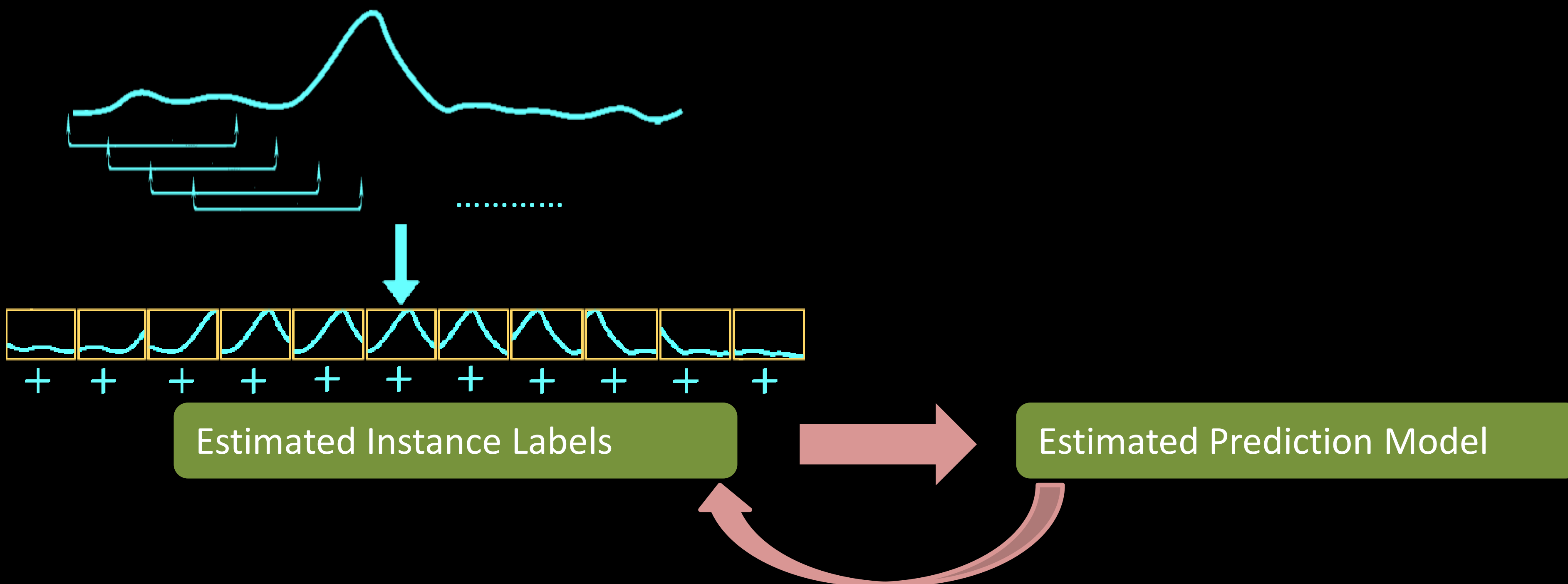
EMI-RNN: Chicken & Egg Problem



Correct Instance Labels ⟷ Accurate Prediction Model

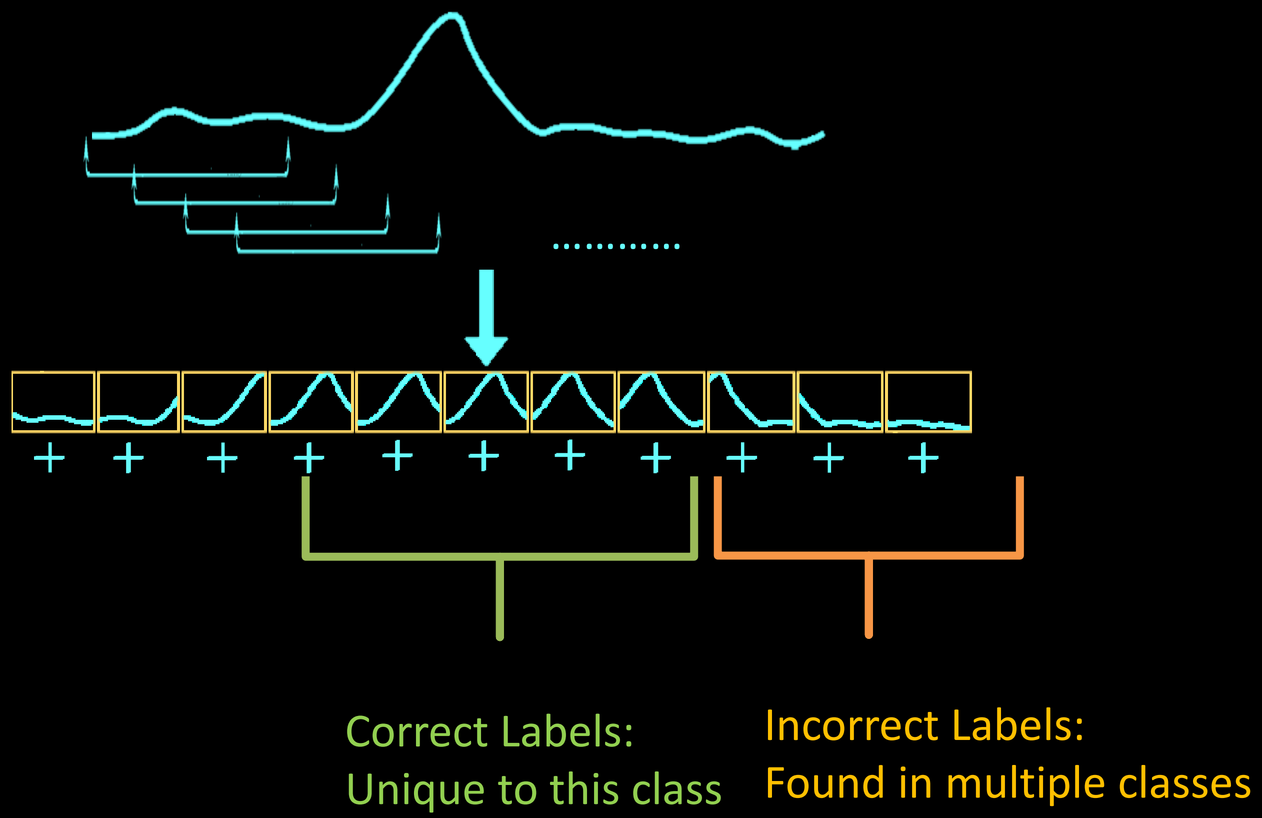
Our Approach: be optimistic and iterate!

EMI-RNN: Algorithm



Iterate till convergence

EMI-RNN: Algorithm



EMI-RNN: Analysis?

- Optimizing:

$$\min_{W,\widehat{y_{ij}}} \sum_{i=1}^{n} \sum_{j=1}^{m} loss(\widehat{y_{ij}}, f(z_{ij}, W))$$

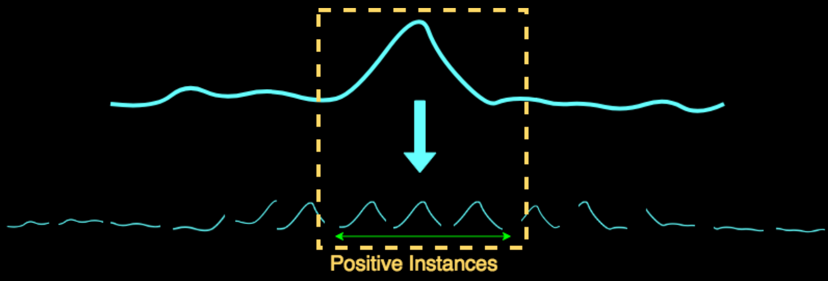
$$s.t. \sum_{j} \hat{y}_{ij} = k, \qquad i: positive$$

$$\hat{y}_{ij}\ satisfies\ temporal\ locality$$

$$\hat{y}_{ij} \in \{0,1\}, i: positive, \qquad \hat{y}_{ij} = 0, i: negative$$

- f: prediction function (LSTM/FastGRNN)
- Algorithm: alternating between $\widehat{y_{ij}}$ and W

EMI-RNN: Analysis?



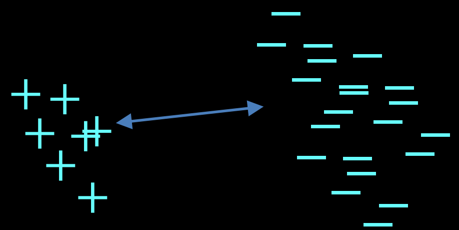
Alternating minimization for Non-convex optimization problem

- Need not converge in general!

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

$$S_* = \{(i,j), \hat{y}_{ij} = +1\}$$

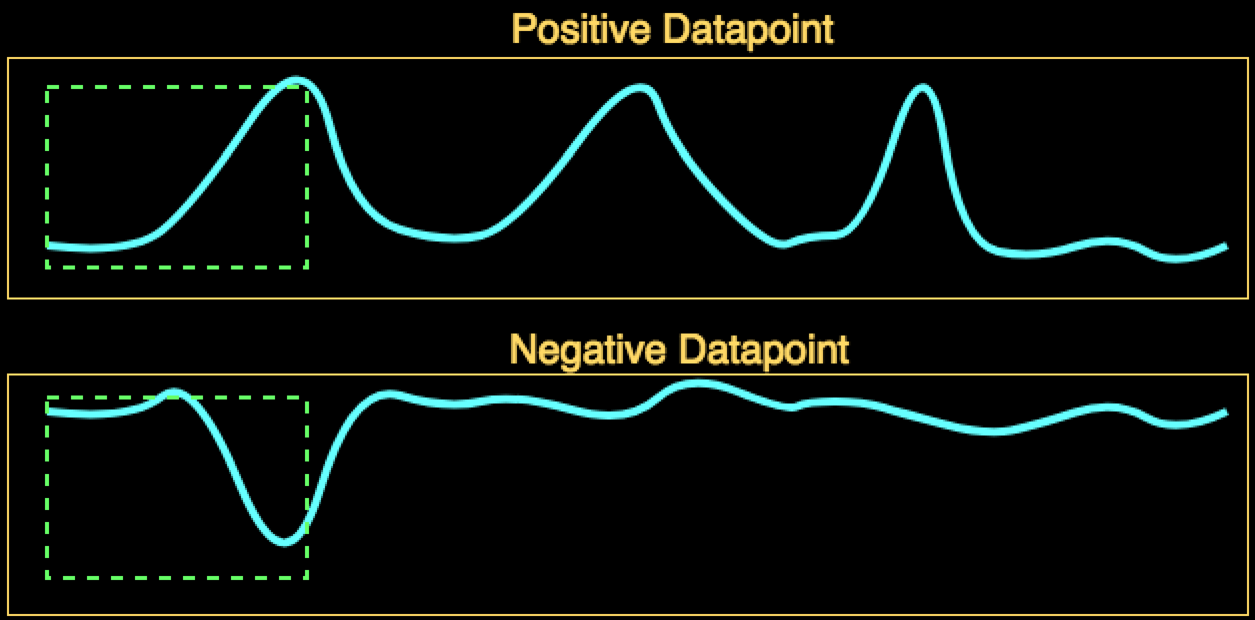
will be recovered exactly, with high probability.



Positives need not be i.i.d.

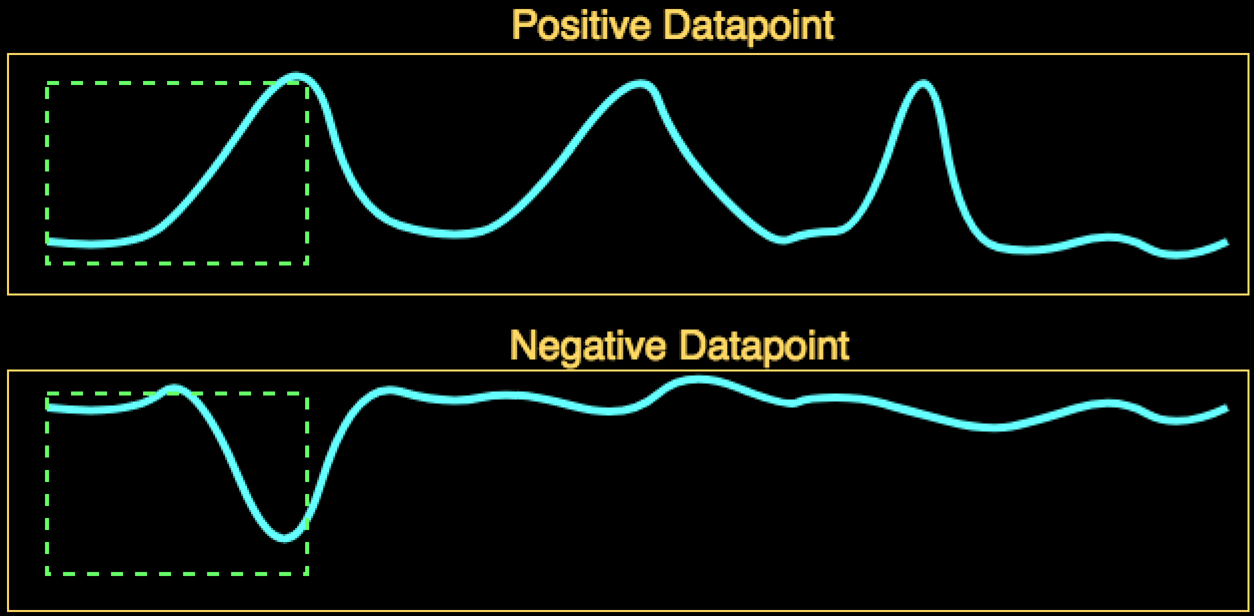
Non-homogenous setting: first such result in literature

Early Prediction?



- Existing work:
  - Assumes pretrained classifier and uses secondary classifiers
  - Template matching approaches
  - Separate policy for early classification

Early Prediction?



Our Approach

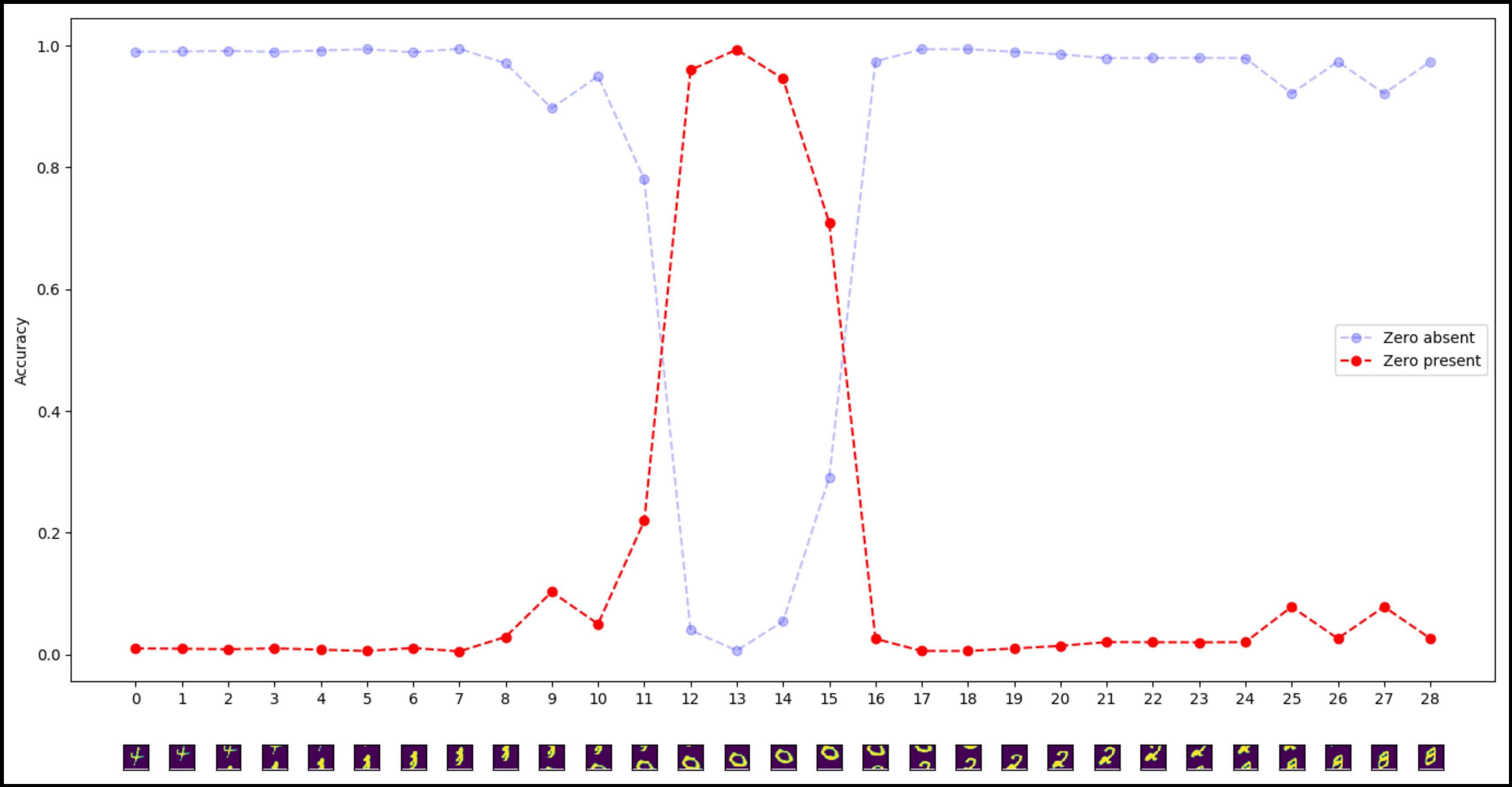
Inference: Predict at each step – stop as soon as prediction confidence is high.

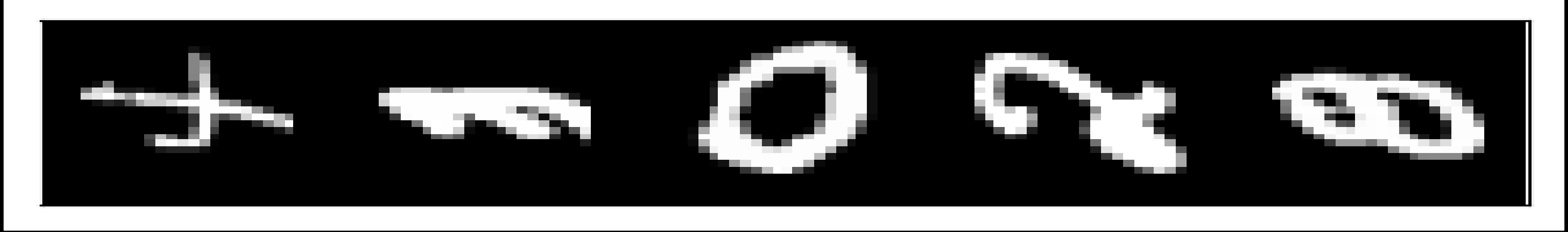
Training: Incentivize early prediction by rewarding correct and early detections.

Advantages

- Obvious: Speed
- Prediction accuracy
- Early prediction

EMI-RNN: Empirical Results





EMI-RNN: Empirical Results

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

EMI-RNN: Empirical Results

Dataset	Accuracy Gain (over Baseline LSTMs)	Prediction Time Reduction	Memory Savings
HAR	0.7%	11x	4x
Sports	2.0%	12x	4x
Google	1.5%	8x	4x
Interactive Cane	0.9%	72x	36x

Time-series Analysis: Conclusions

- Complicated RNN cell updates: FastGRNN
- Running time O(T): EMI-FastGRNN
- Information reuse across windows: Shallow Recurrent Networks

EdgeML repository

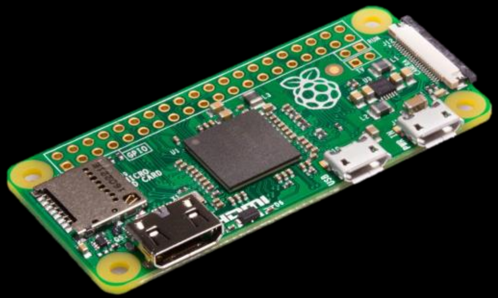
- Code release
  - EdgeML: 70K page views, 1017 clones, 569 stars
  - Bonsai and ProtoNN released as TLC Beta

- Used extensively in MSR India IoT Summer School
  - Automated voice feedback system
  - Radar-based poacher detection and SONYC
  - Predictive maintenance for solar panels

- https://github.com/Microsoft/EdgeML

Thank You

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)

Appendix

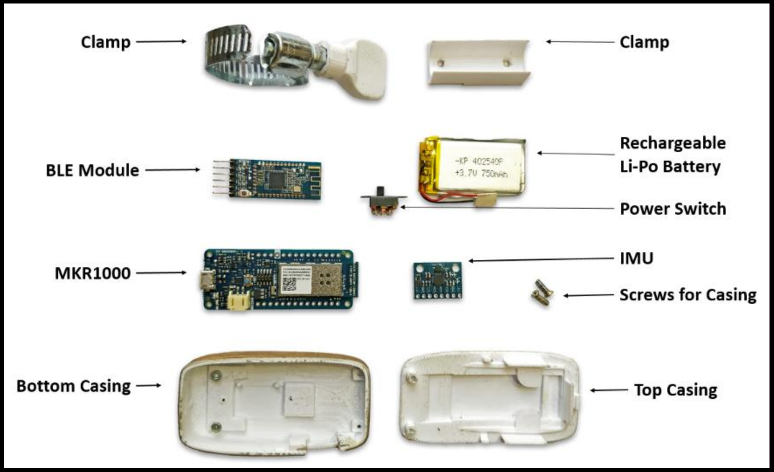
Demos: Algorithms in real-time

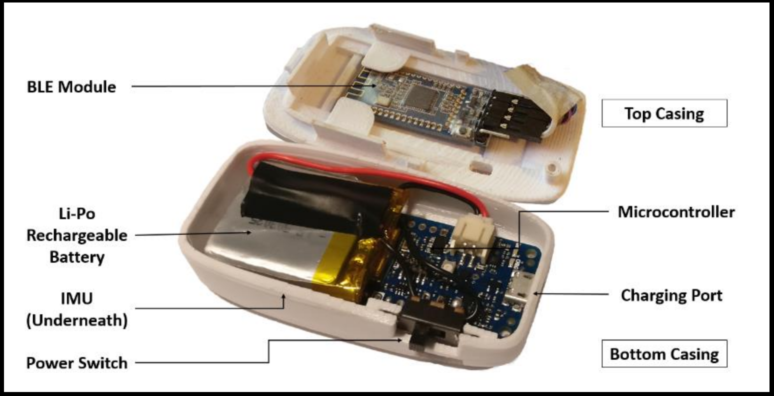
Interactive Cane / GesturePod:

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

Wakeword detection

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



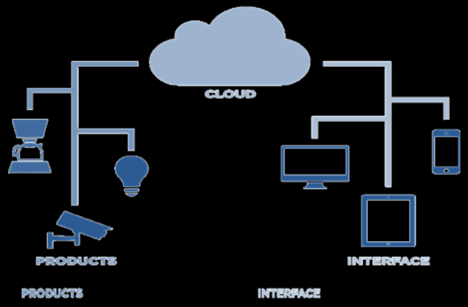


Conclusions

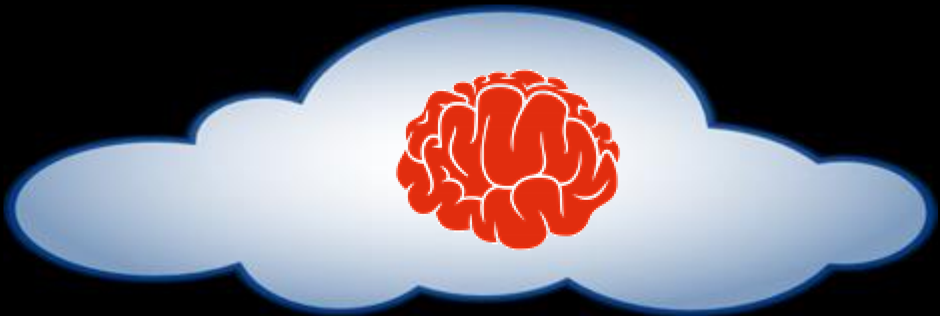
- ML for IoT devices provides many high-impact opportunities
- Microsoft's Edge Machine Learning Library
  - https://github.com/Microsoft/EdgeML
- Bonsai, ProtoNN, FastGRNN & EMI-RNN
  - Fit into a few Kilobytes of memory
  - Make predictions in milliseconds
  - Are energy efficient & extend battery life
  - Have state-of-the-art prediction accuracies

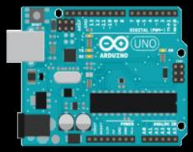
The Internet of Things

Smart City



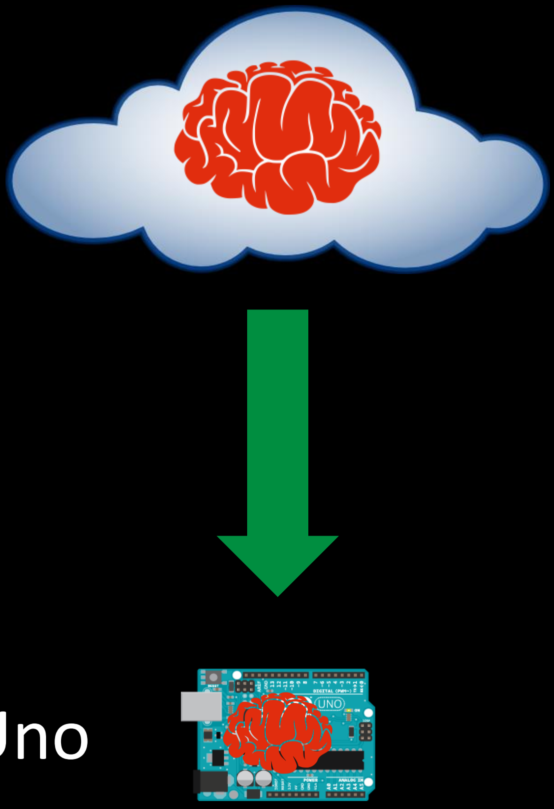
Smart Appliances





Edge Machine Learning – Objectives

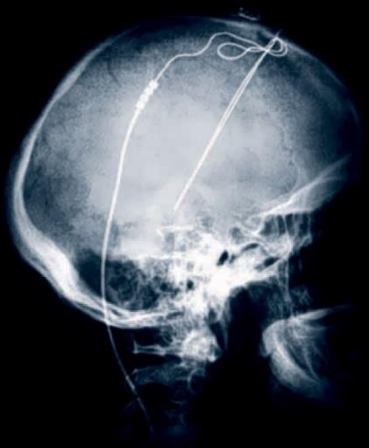
- To build a library of machine learning algorithms
  - Which can be trained on the cloud
  - But which will run on tiny IoT devices



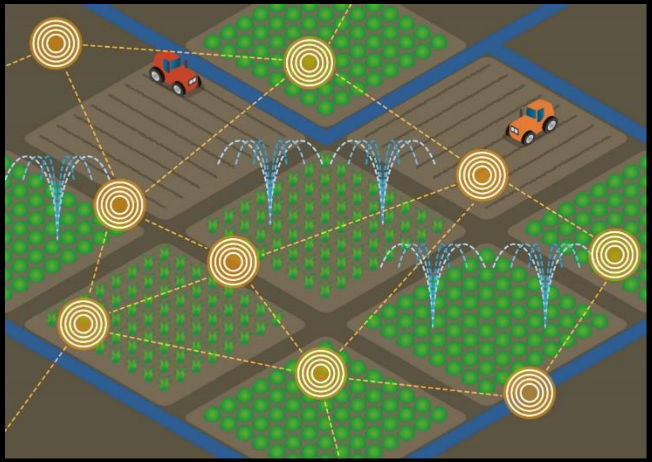
Arduino Uno

Intelligent IoT Devices

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



Low latency brain implants

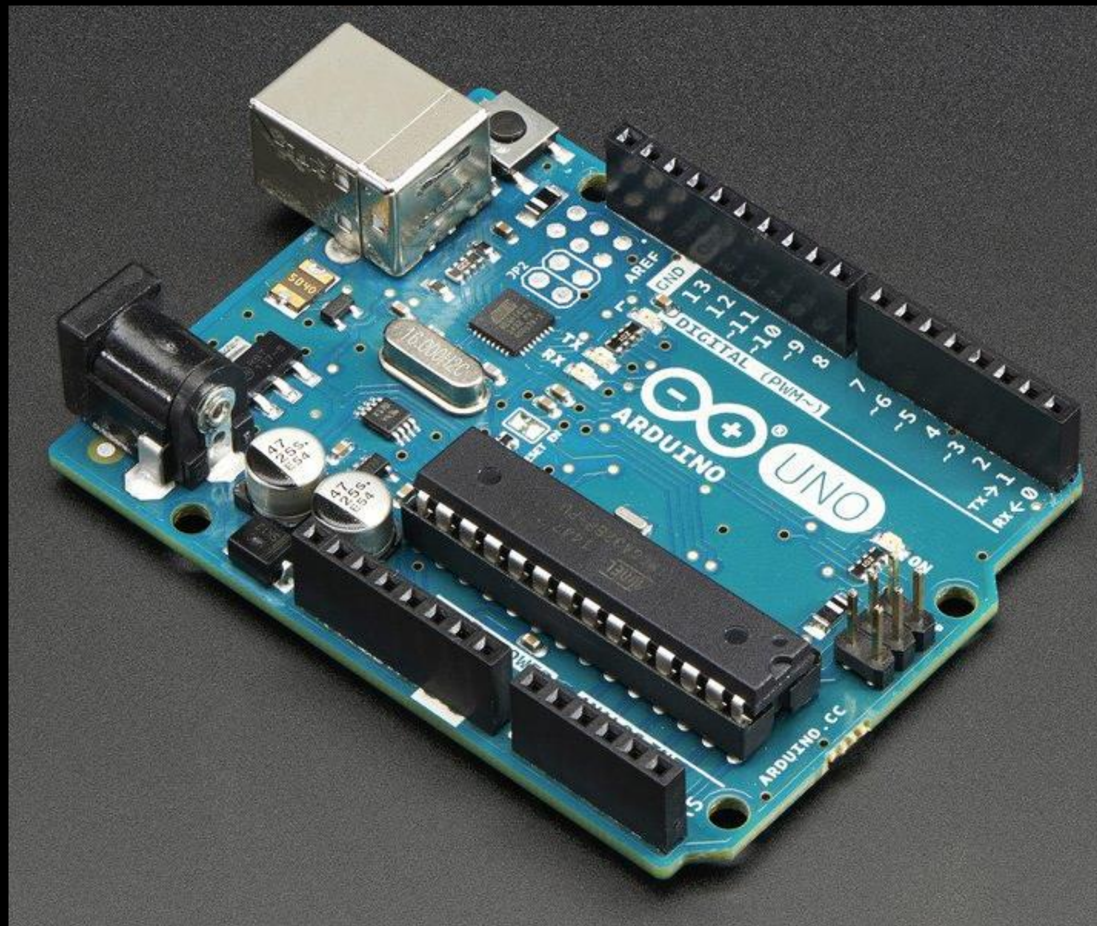


Smart agriculture for disconnected farms

Privacy preserving smart glasses

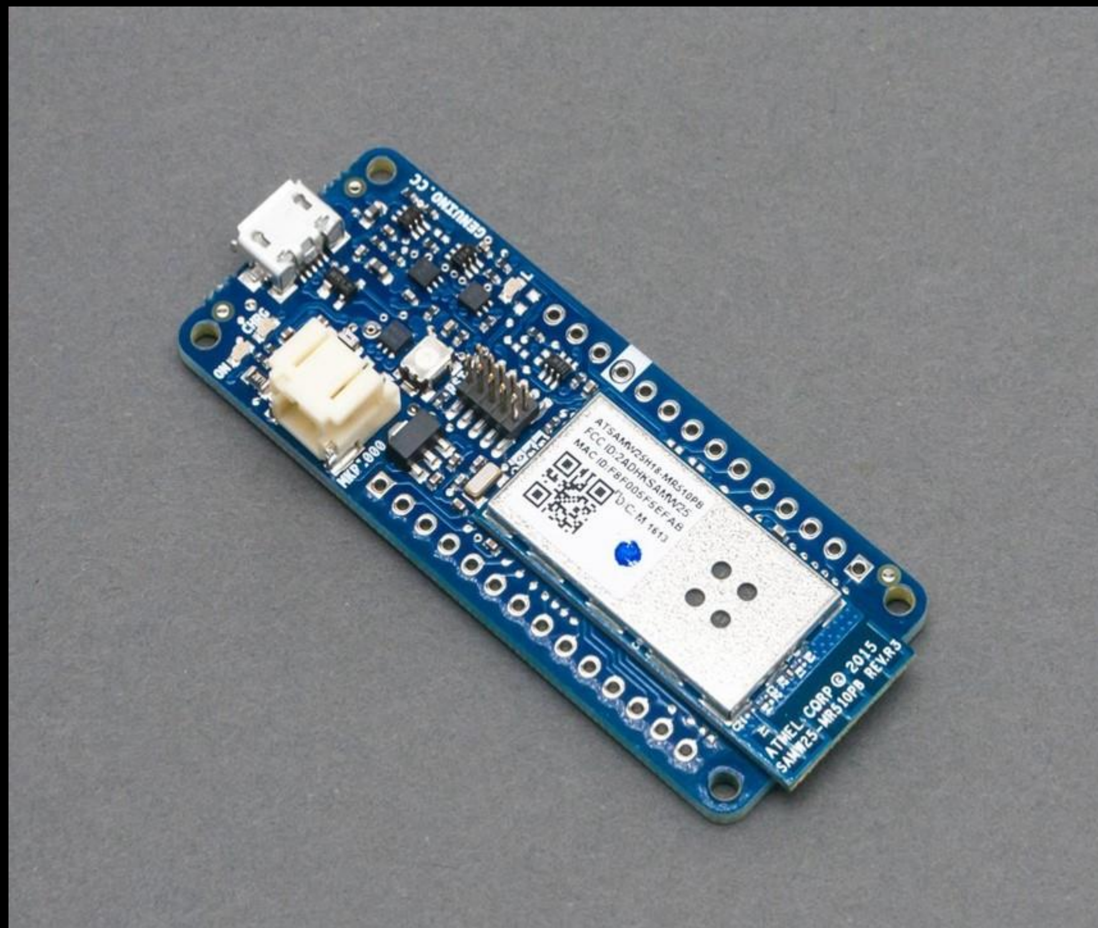
Energy efficient smart forks

The Arduino Uno



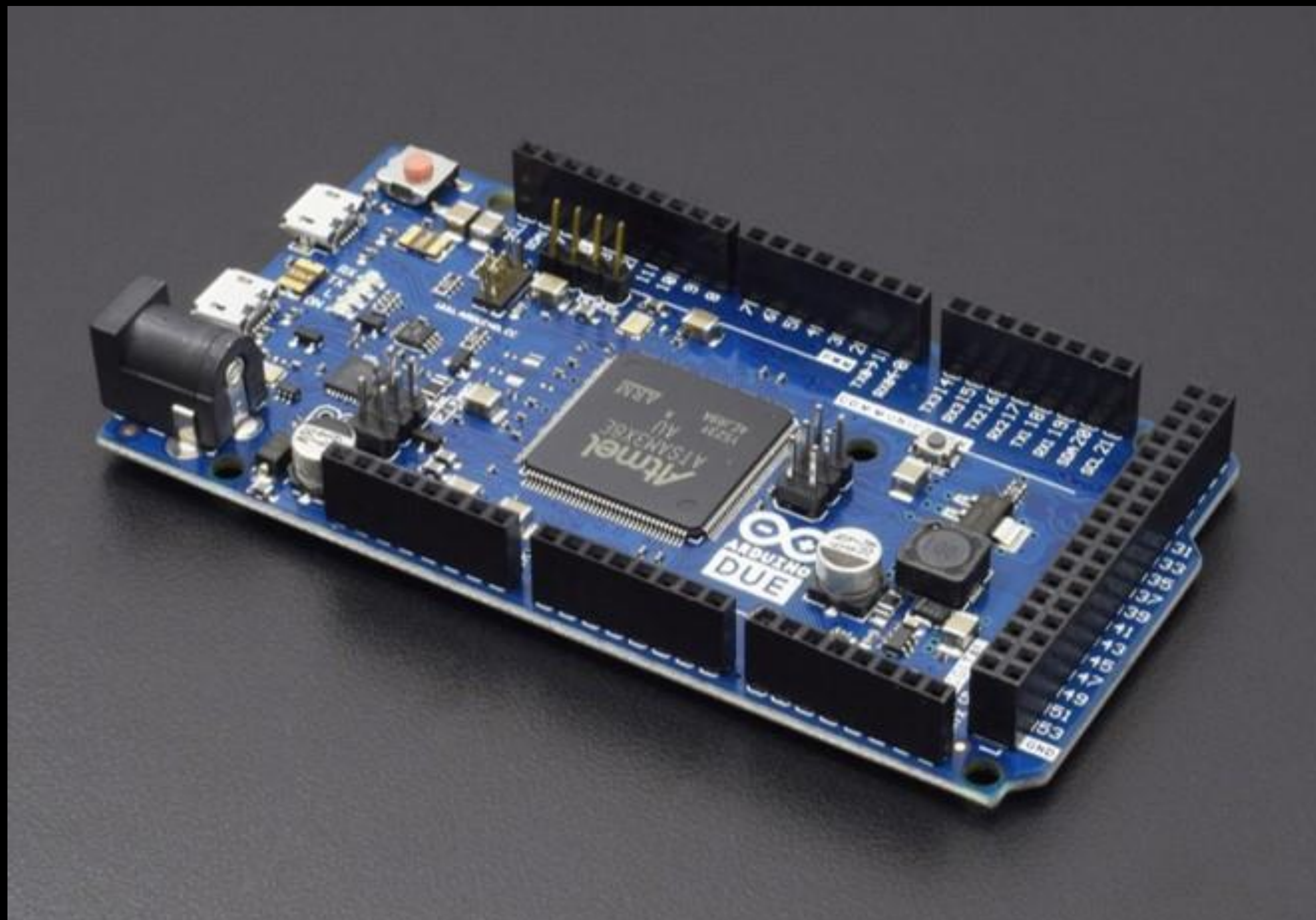
8 bit ATmega328P Processor at 16 MHz
with 2 KB RAM & 32 KB read only Flash

The Arduino MKR1000



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

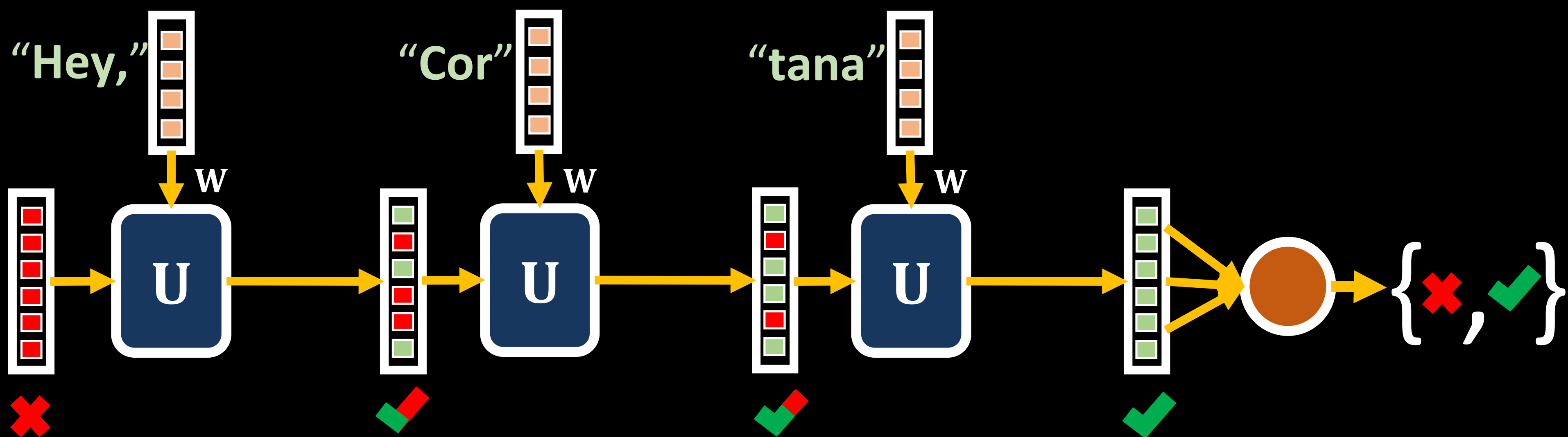
The Arduino Due



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

Recurrent Neural Networks (RNNs)

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



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

Theorems

Convergence Bound:

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

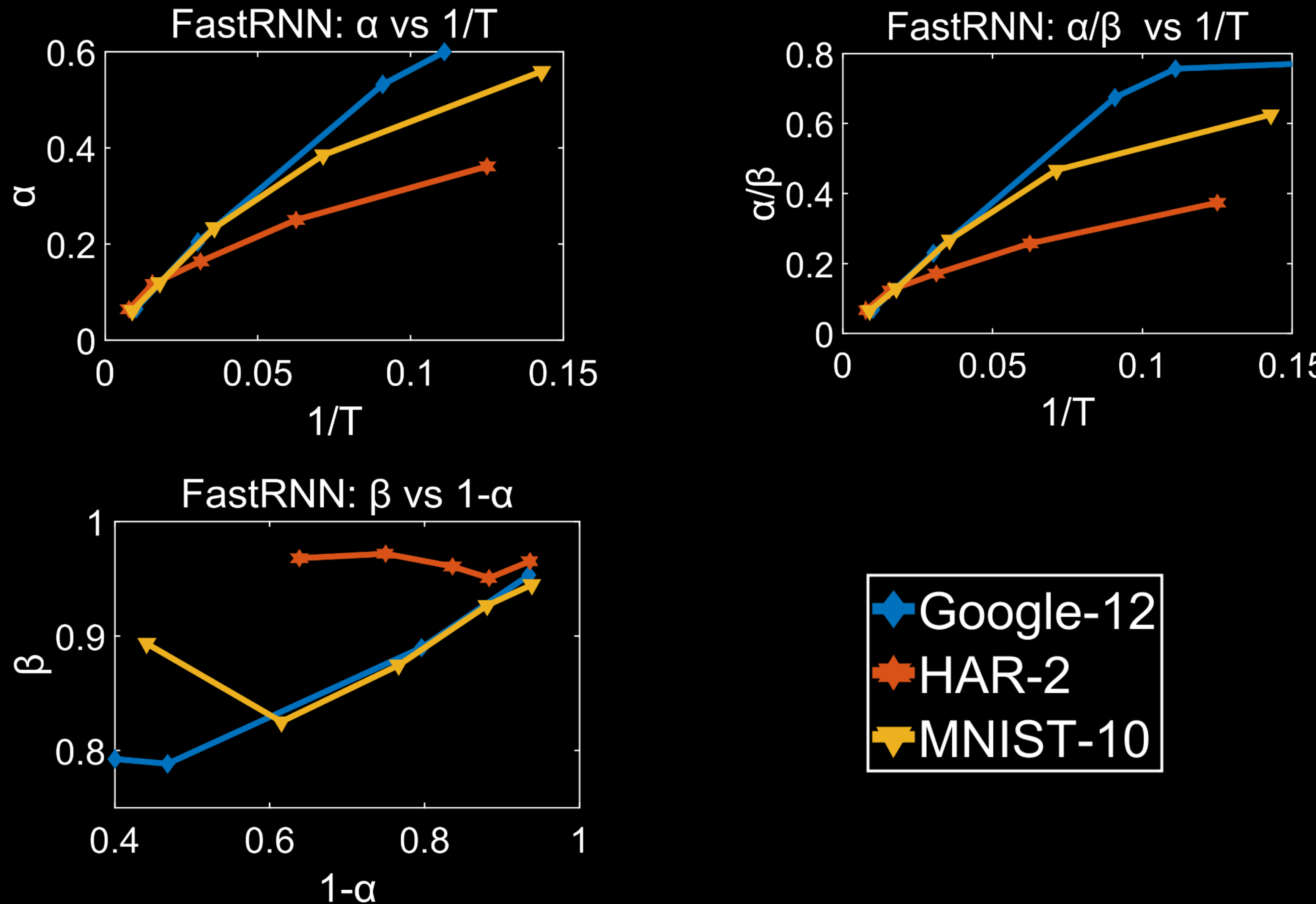
Generalization Error Bound:

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

$$\alpha \approx O(1/T), \beta = 1 - \alpha$$

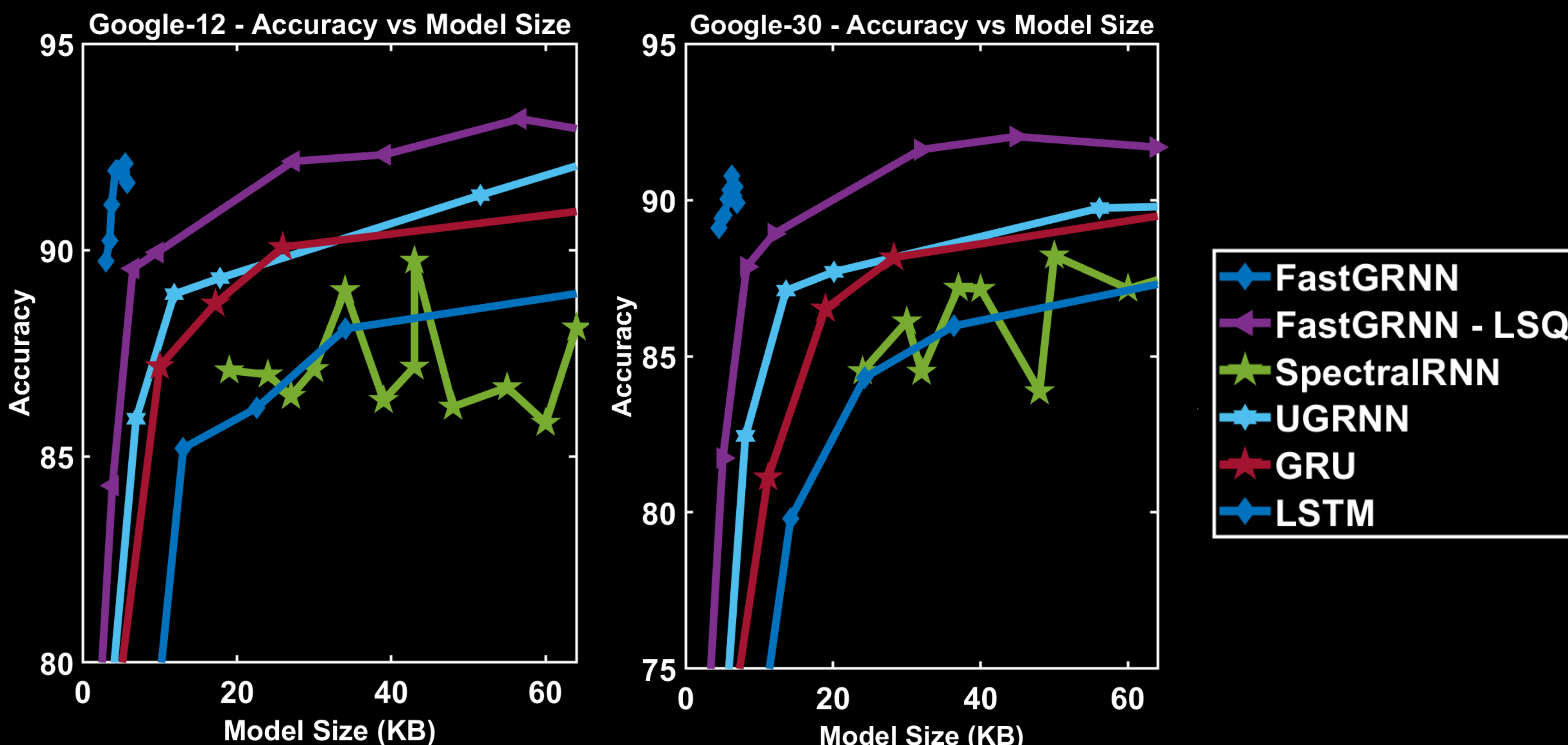
L is any 1-Lipschitz loss function bounded by [0, B].
N is # SGD iterations and n is # datapoints.
$\bar{D} \geq 0$ helps in choosing right learning rate. $R_{\mathbf{X}} = max_{\mathbf{X}} \|\mathbf{X}\|_{\mathrm{F}}$.

FastRNN: α & β vs T

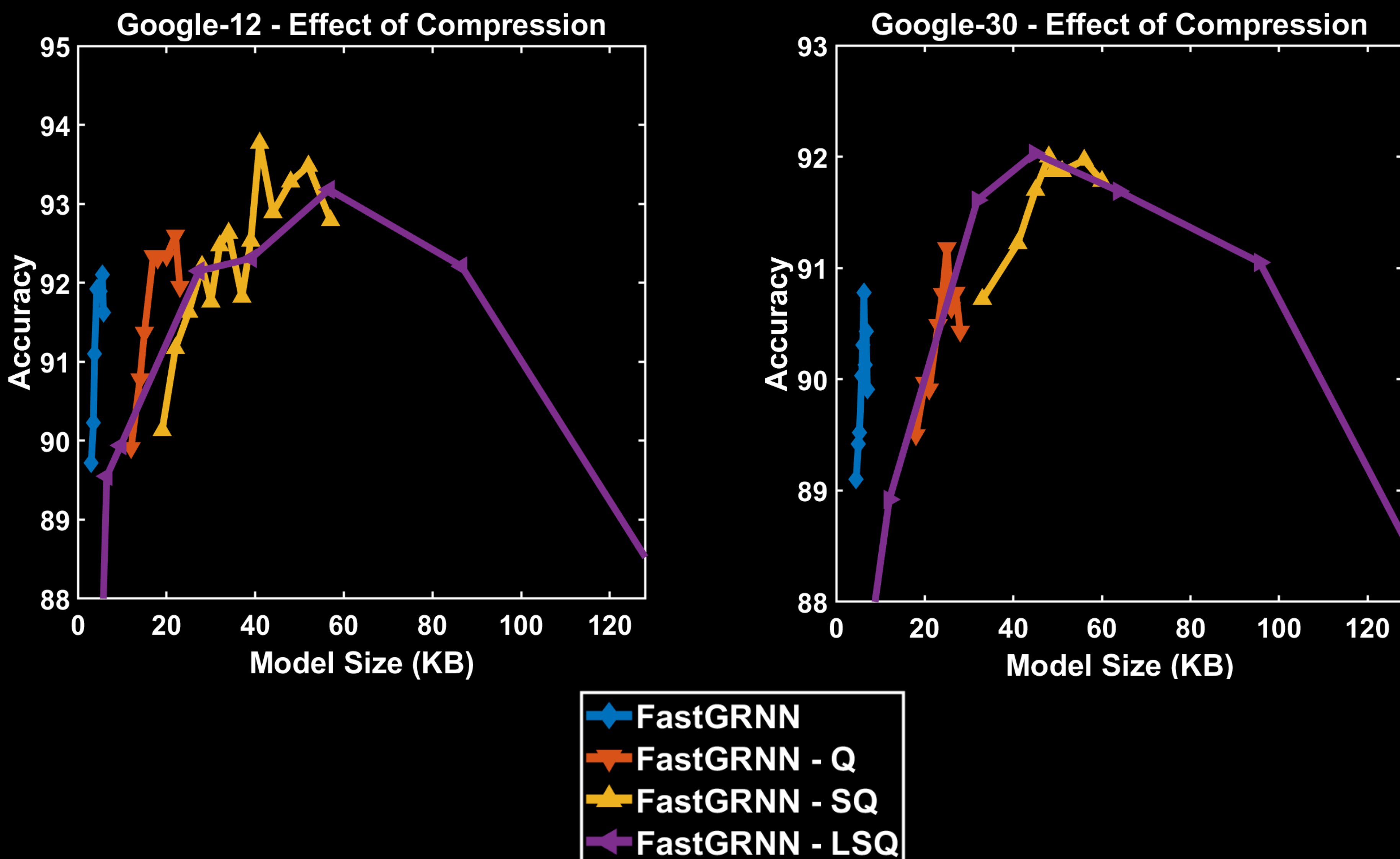


Prediction Accuracy vs Model Size

- FastGRNN-LSQ & FastGRNN dominate all other RNNs in 0-64 KB



Effects of Compression (LSQ)



EMI-RNN: Empirical Results

Dataset	Accuracy Gain (over Baseline LSTMs)	Prediction Time Reduction
HAR	-1.0%	42x
Sports	2.0%	8x
Google	-1.0%	32x
Interactive Cane	0.9%	72x

Engagements

- MSRI IoT Summer School (11th June – 6th July)

Team	Project	Outcome
Gaia (Startup)	Automated voice feedback	FastGRNN deployed on device Product integration in progress
DataGlen (Startup)	Solar panel maintenance	Working on a paper Possible tech-transfer
OSU	Poacher detection & SONYC	Targeting UBICOMP 2019
Amrita University	IoT compiler	Exploring integration with Azure Sphere + Code for cane