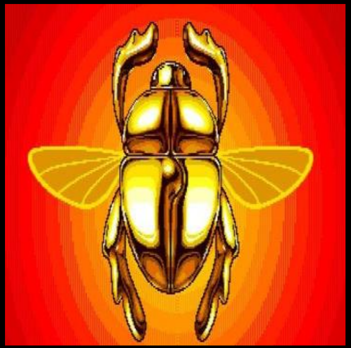


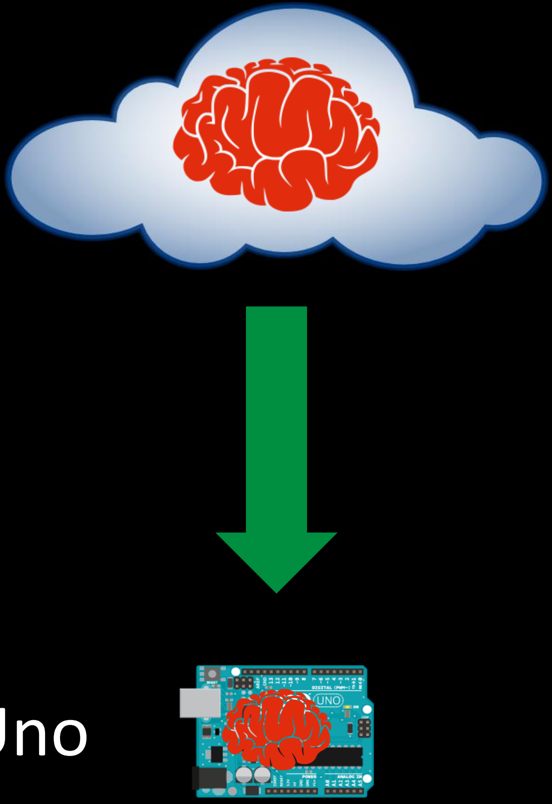
The Edge of Machine Learning

Multiple Instance Learning for Fast, Stable and Early RNN Predictions

Don Dennis,
Microsoft Research India,
Joint work with Chirag P., Harsha and Prateek
Accepted to NIPS ’18

Algorithms for the IDE - EdgeML

- A library of machine learning algorithms
  - Trained on the cloud
  - Ability to run on tiniest of IoT devices



Arduino Uno

Previous Work: EdgeML Classifiers

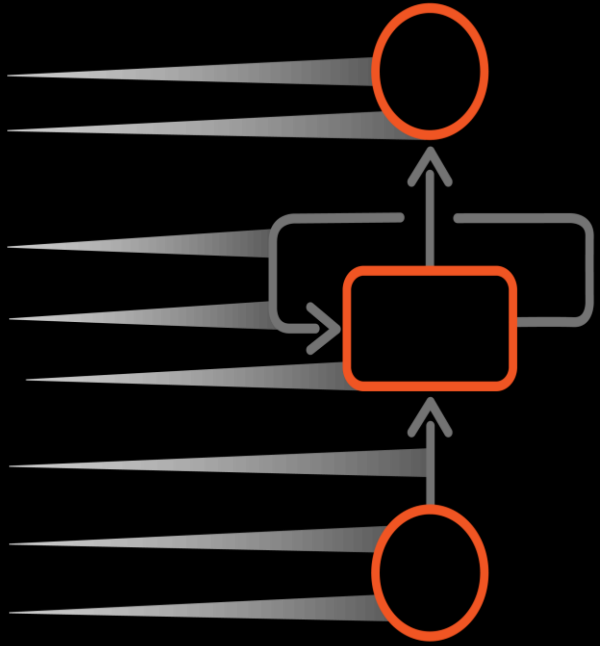
ProtoNN

Gupta et al., ICML ’17

Bonsai

Kumar et al., ICML ’17

Fast(G)RNN

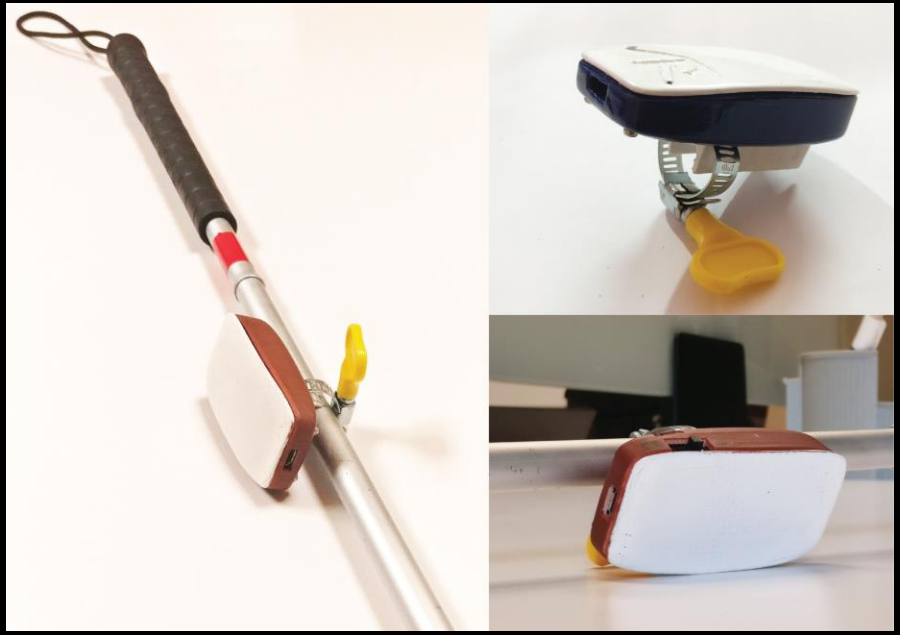


Kusupati et al., NIPS ’18

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

Previous Work: EdgeML Applications

GesturePod



Patil et al.,
(to be submitted)

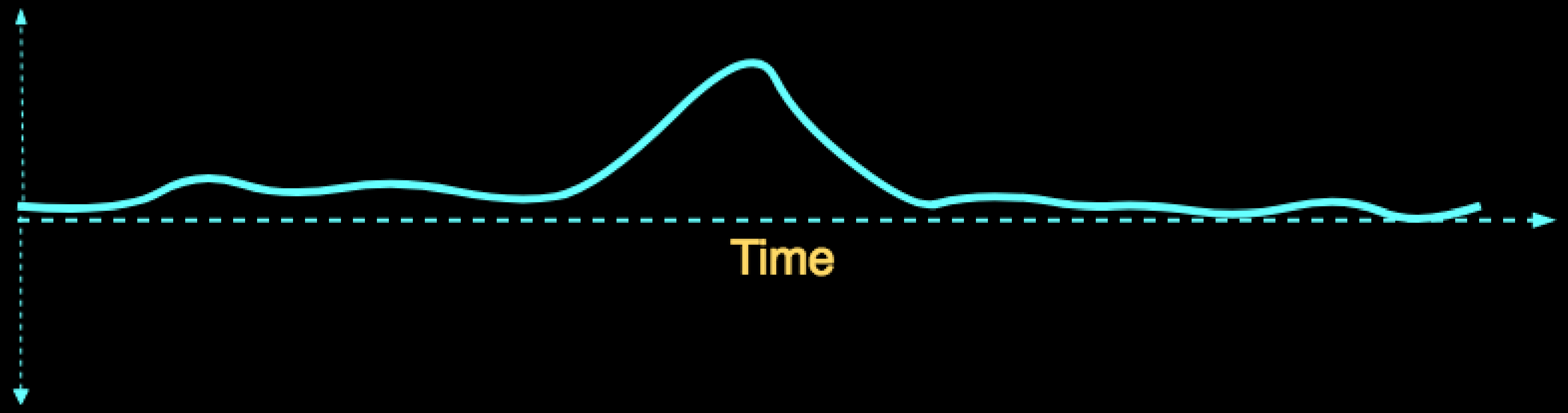
Wake Word

(work in progress)

Code: En route

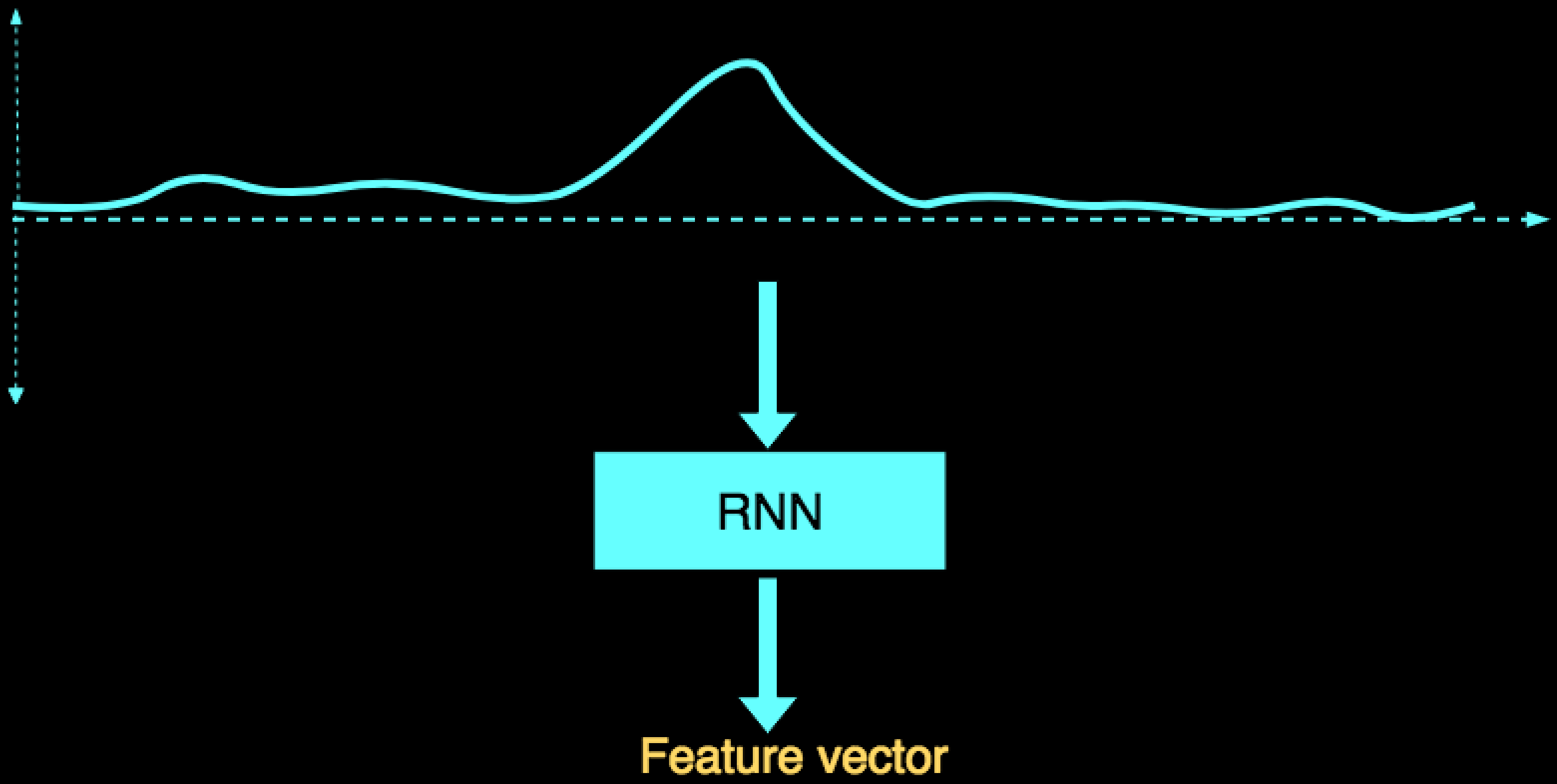
Problem

Problem

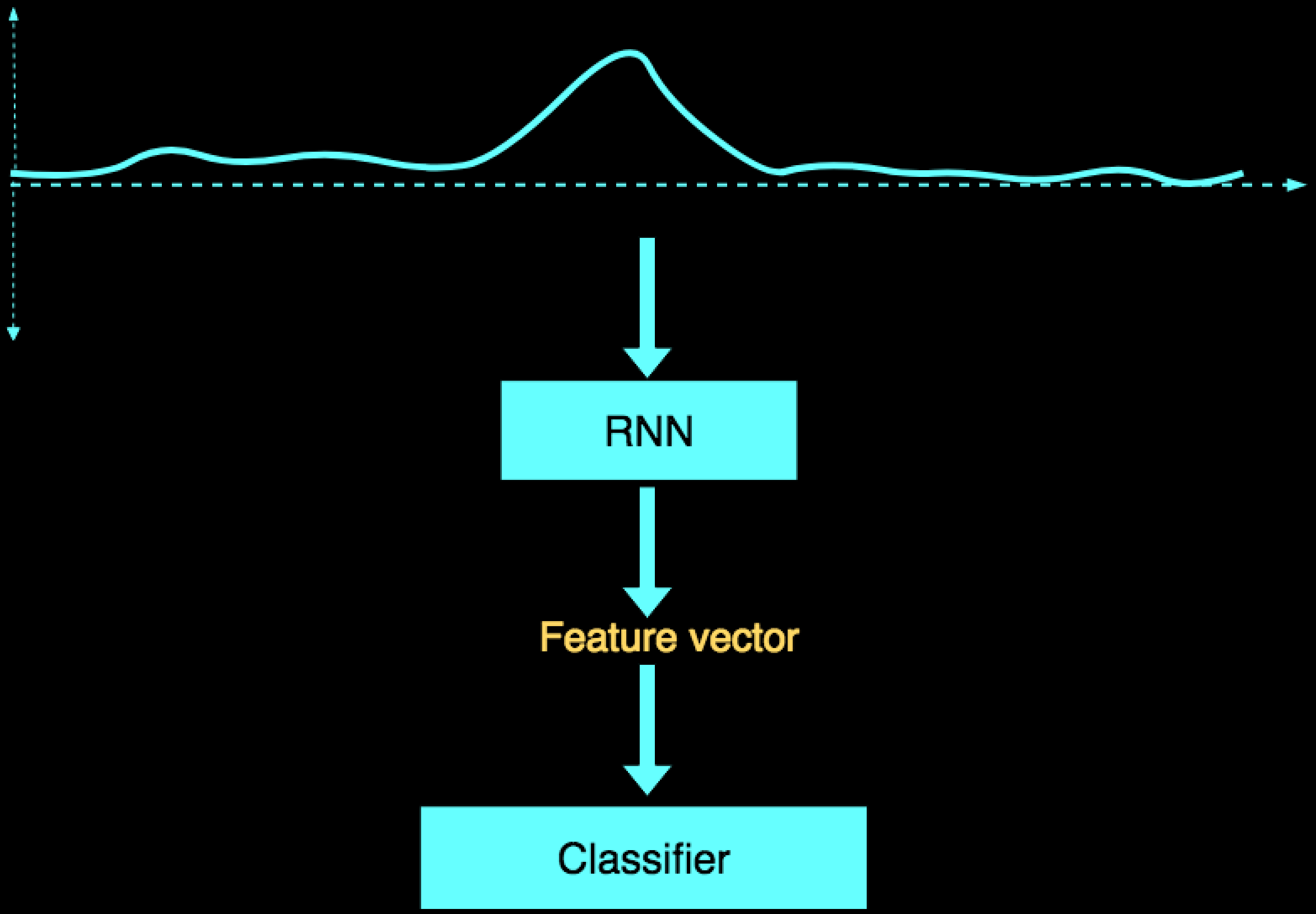


- Given time series data point, classify it as a certain class.
- GesturePod:
  - Data: Accelerometer and gyroscope information
  - Task: Detect if gesture was performed

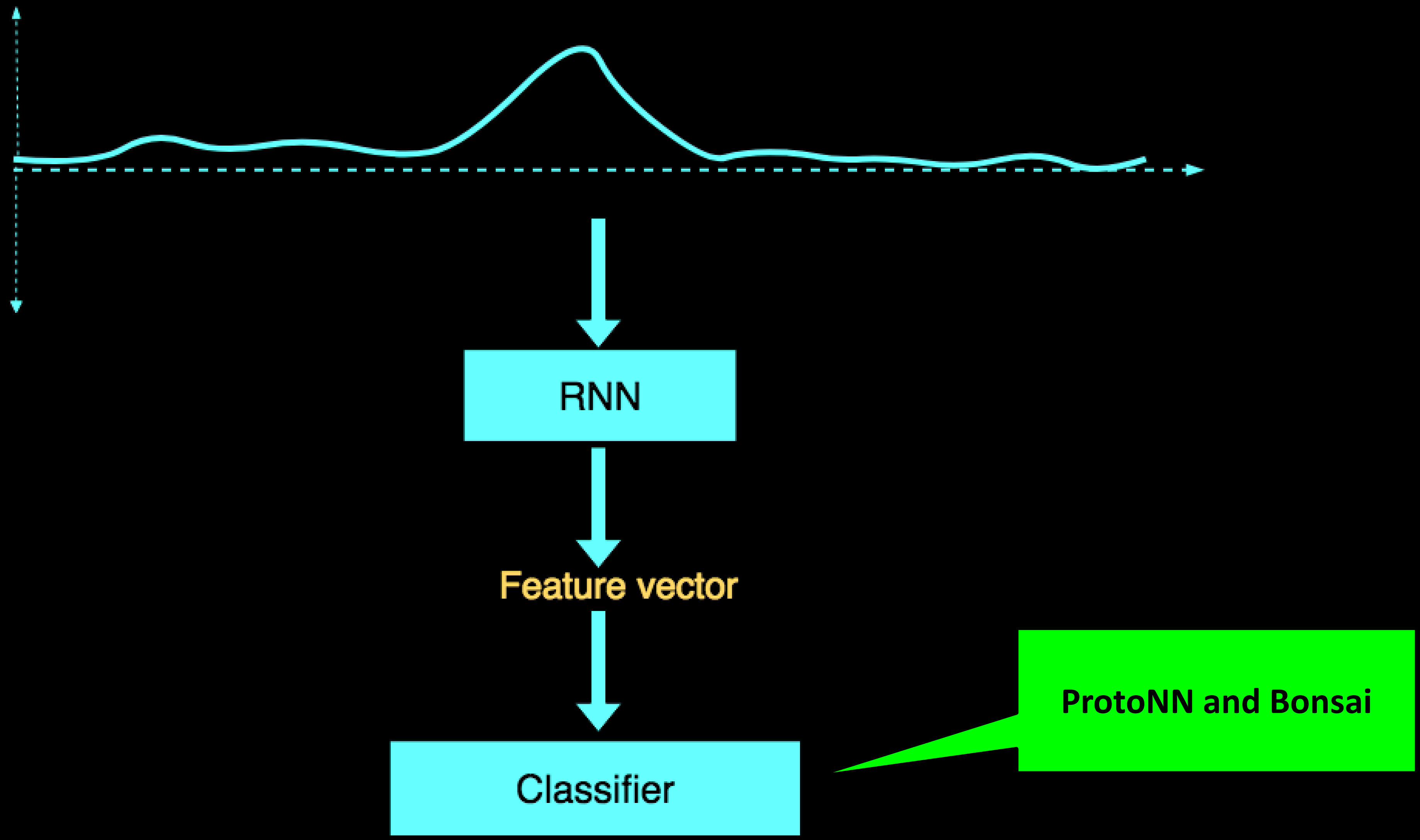
Problem



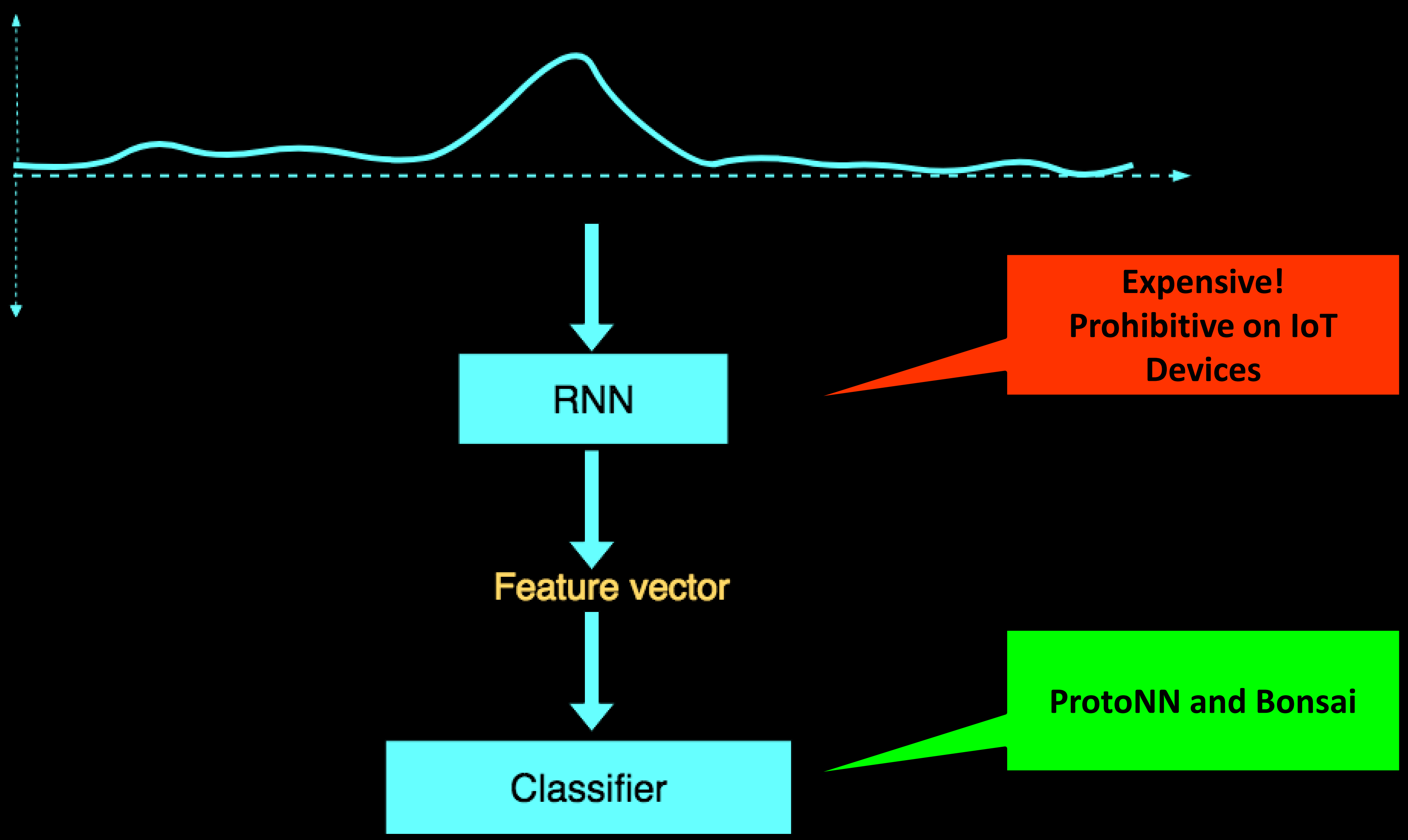
Problem



Problem



Problem


RNN
Feature vector
Classifier
Expensive!
Prohibitive on IoT
Devices
ProtoNN and Bonsai

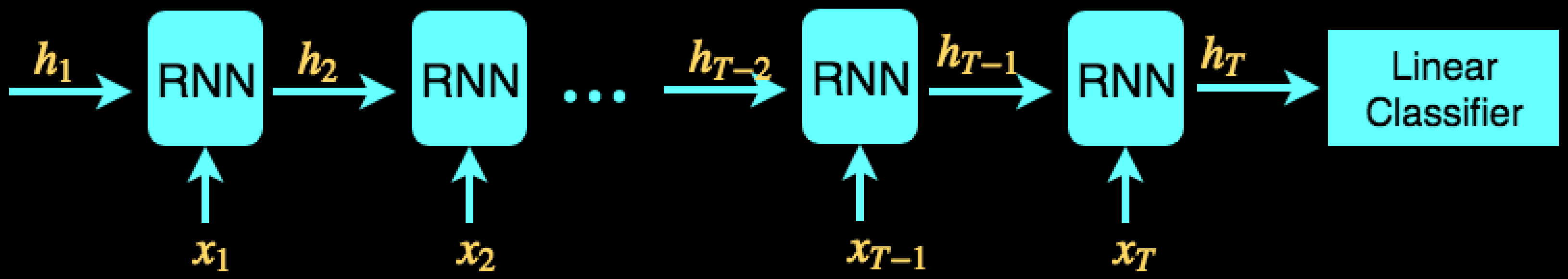
RNNs are Expensive

- For time series data: $X = [x_1, x_2, x_3, \ldots, x_T] \quad x \in \mathbb{R}^d$
- *T* RNN updates are performed:

$$h_t = \sigma(\mathbf{w}x_t + \mathbf{u}h_{t-1} + b)$$

- *T* is determined by the data labelling process. Example *GesturePod* – 2 seconds.

RNNs are Expensive

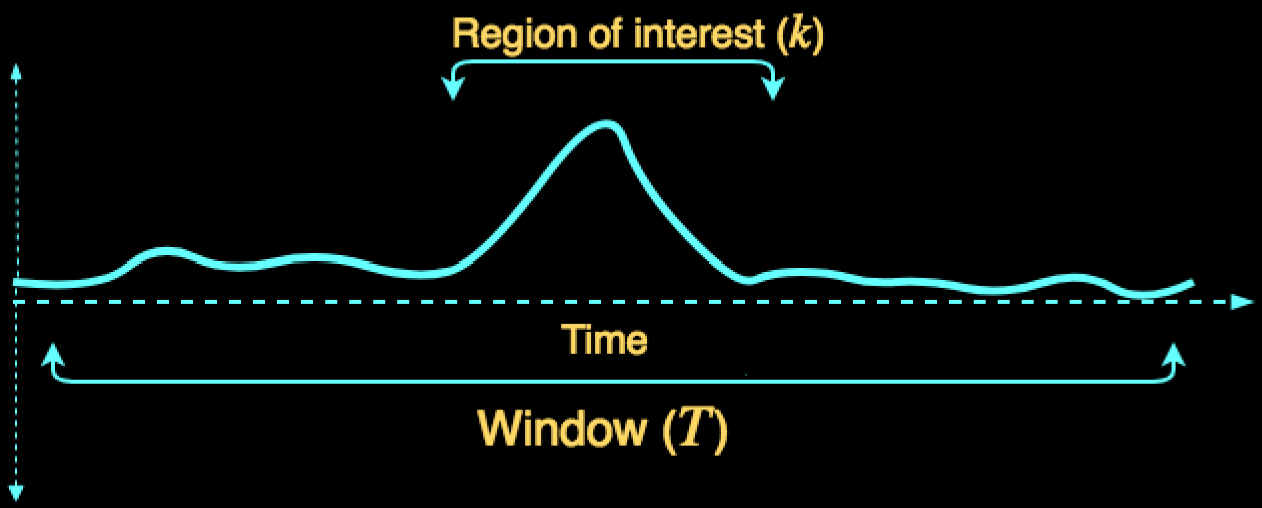


- For time series data: $X = [x_1, x_2, x_3, \ldots, x_T] \quad x \in \mathbb{R}^d$
- *T* RNN updates are performed:

$$h_t = \sigma(\mathbf{w}x_t + \mathbf{u}h_{t-1} + b)$$

- *T* is determined by the data labelling process. Example *GesturePod* – 2 seconds.

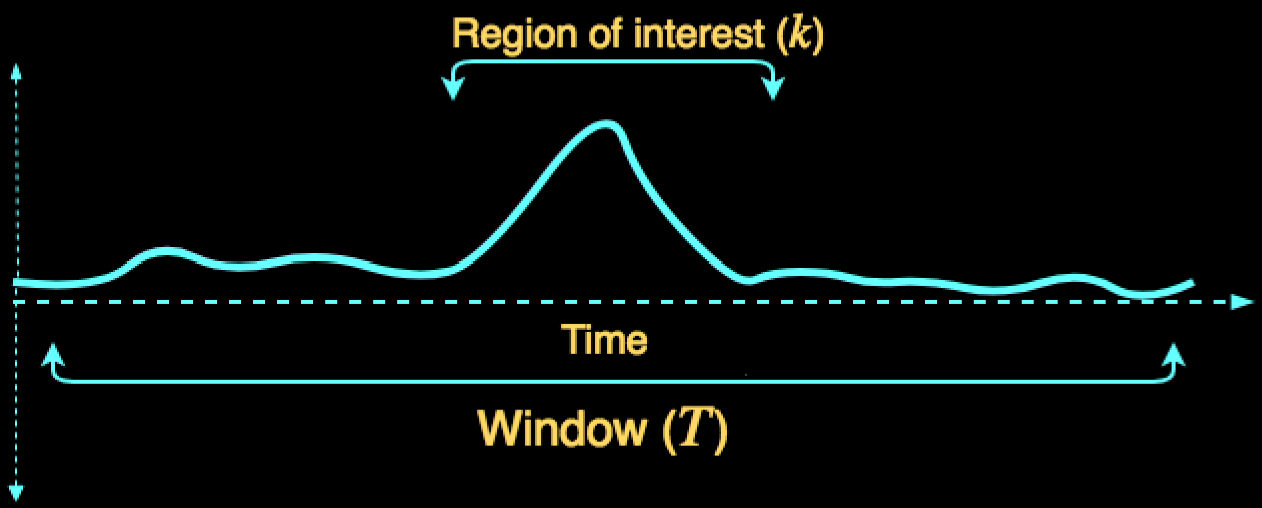
RNNs are Expensive



Observe how $k << T$.

- RNN runs over longer data point – *unnecessarily large T* and prediction time.
- Predictors must recognize signatures with different offsets - *requires larger* predictors.
- Sequential compute.
- Also lag.

RNNs are Expensive



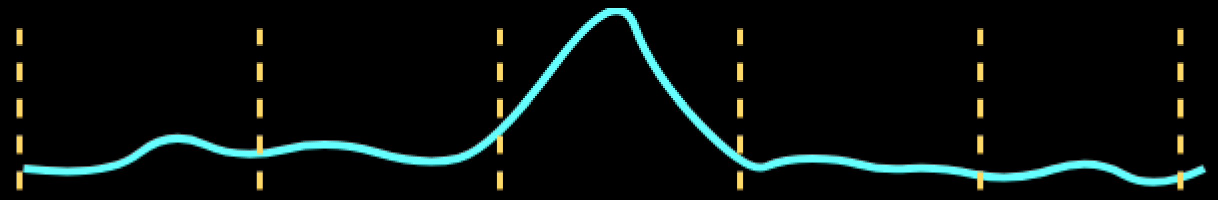
Solution ?

Approach 1 of 2 : Exploit the fact that $k << T$ and learn a smaller classifier.

How?

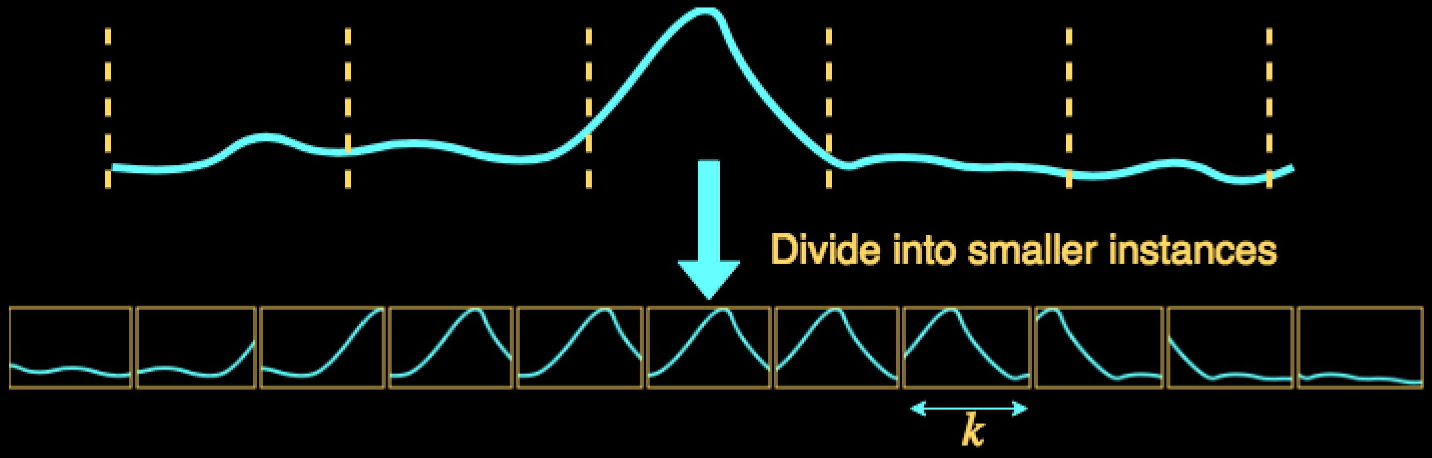
How ?

- STEP 1: Divide *X* into smaller instances.



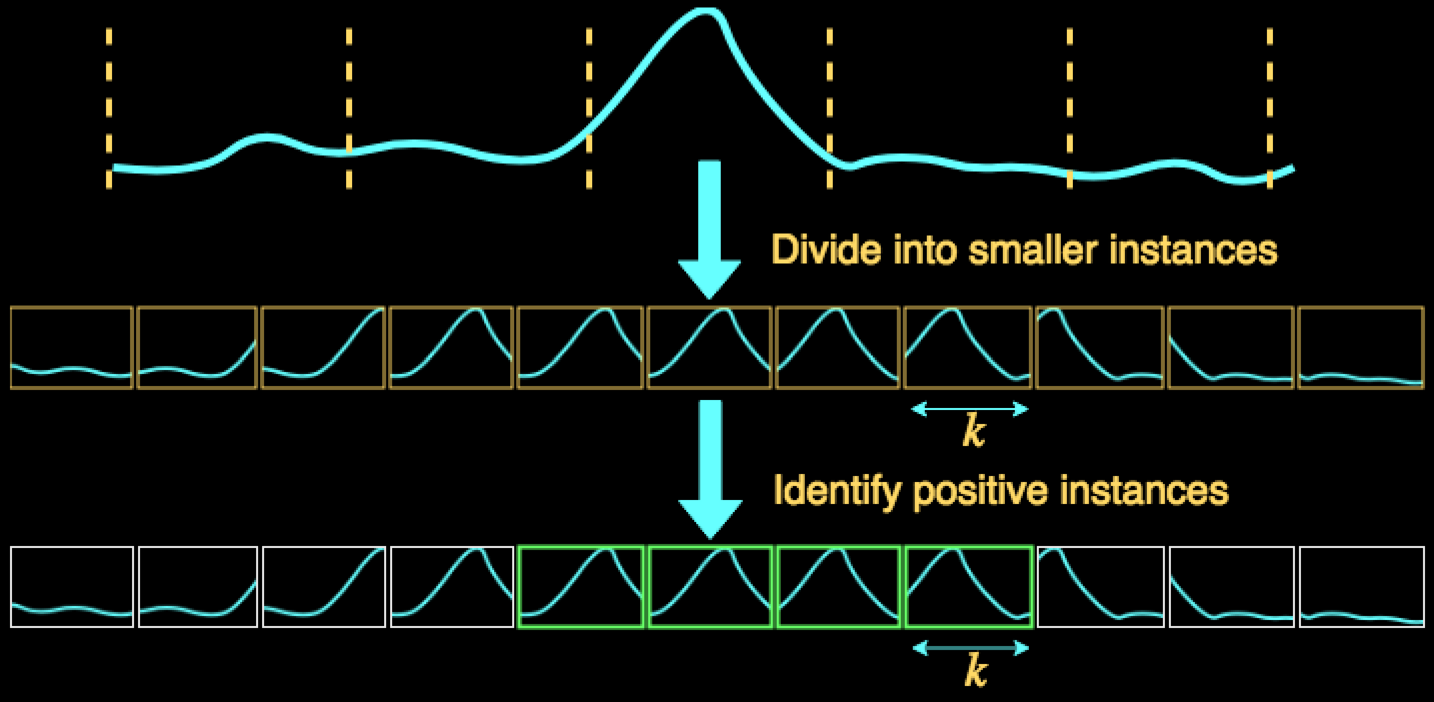
How ?

- STEP 1: Divide *X* into smaller instances.



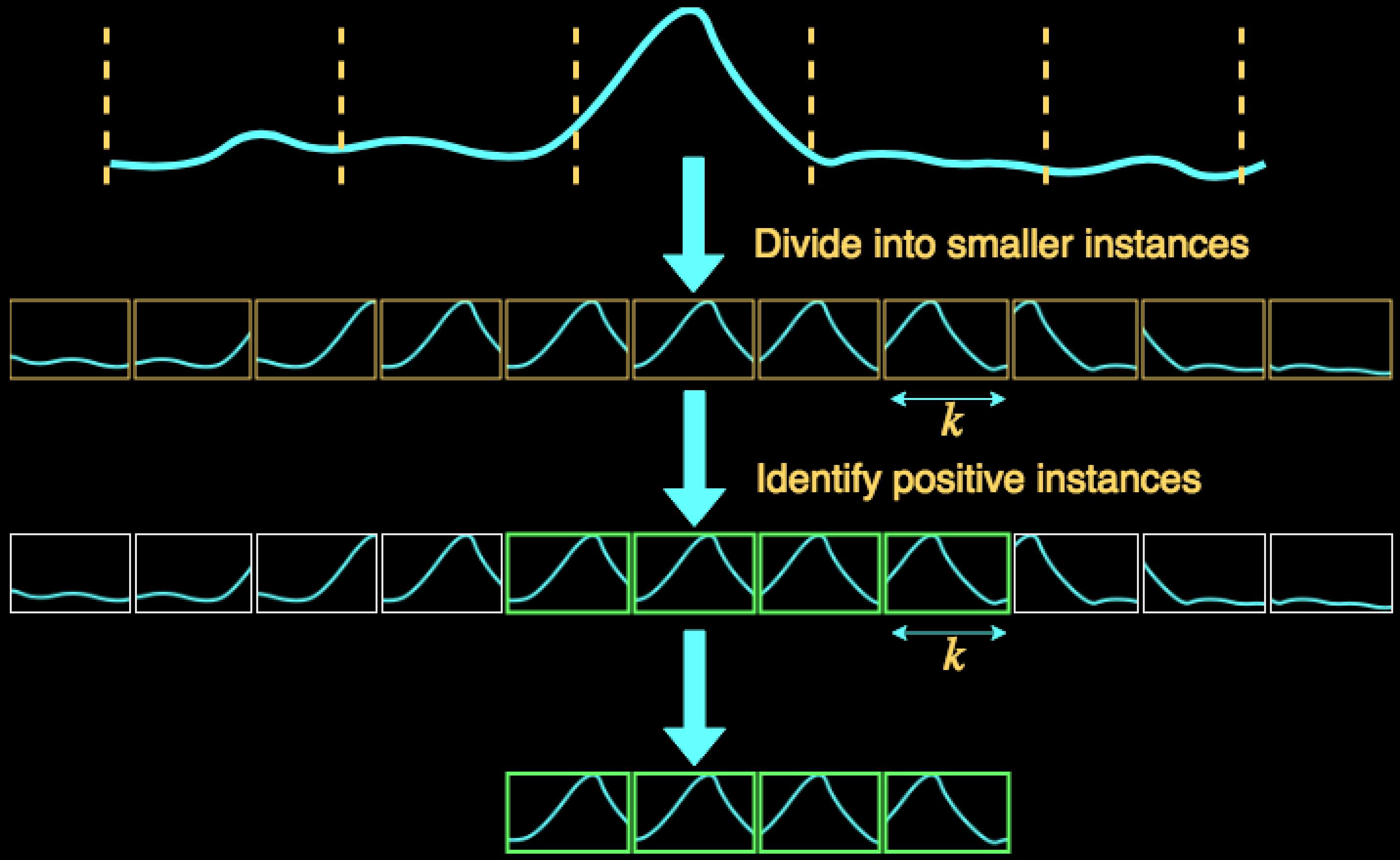
How ?

- STEP 1: Divide *X* into smaller instances.
- STEP 2: Identify positive instances. Discard negative (noise) instances.



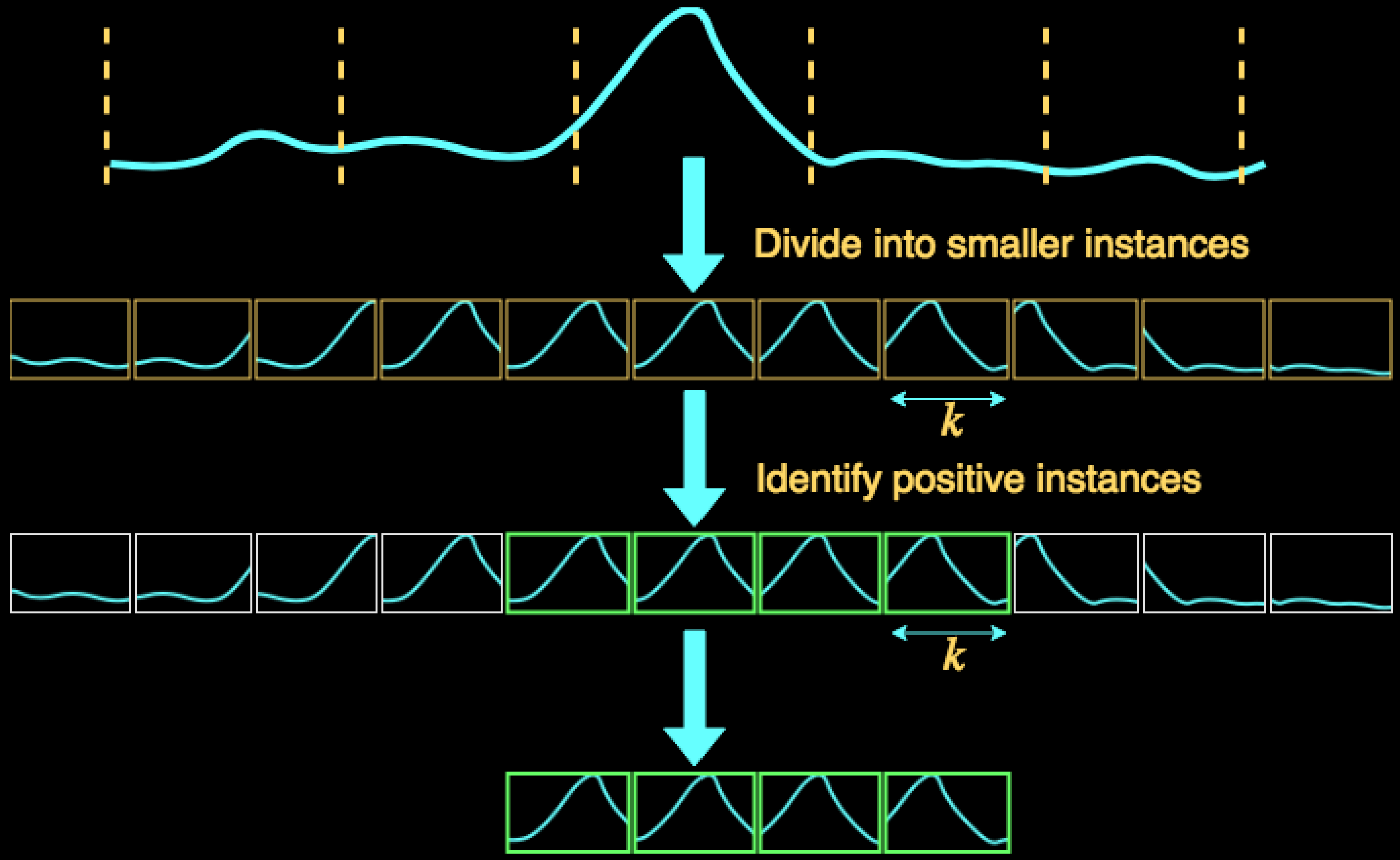
How ?

- STEP 1: Divide *X* into smaller instances.
- STEP 2: Identify positive instances. Discard negative (noise) instances.



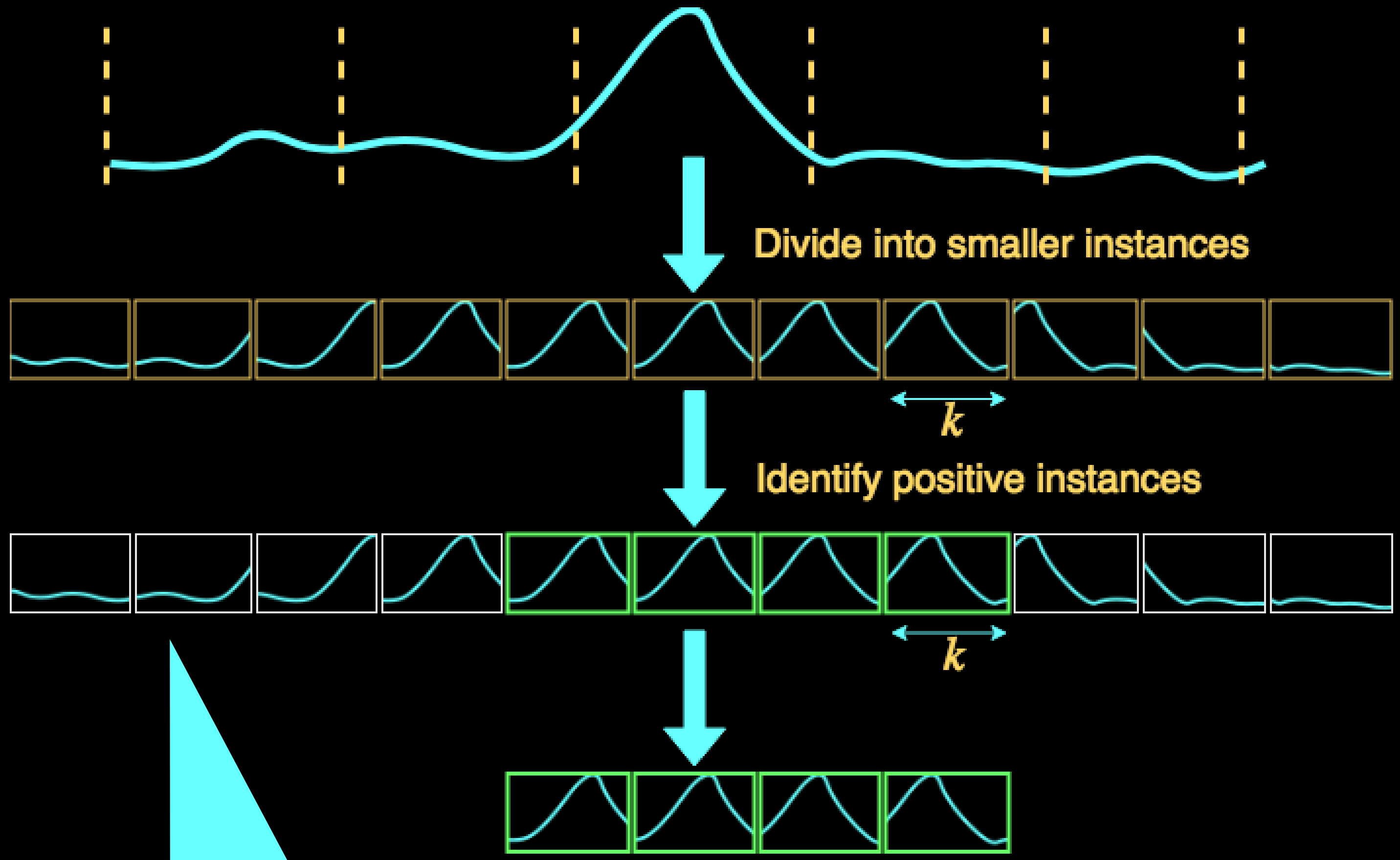
How ?

- 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.



How ?

- 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.



Note! Most of the instances are just *noise.*

How ?

- 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.

How ?

- 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.

Robust Learning

How ?

- 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.

~~Robust Learning~~

Standard techniques don't apply.

- Too much noise.
- Ignores temporal structure of the data.

How ?

- 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.

~~Robust Learning~~

Standard techniques don't apply.

- Too much noise.
- Ignores temporal structure of the data

Traditional Multi Instance Learning (MIL)

How ?

- 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.

~~Robust Learning~~

Standard techniques don't apply.

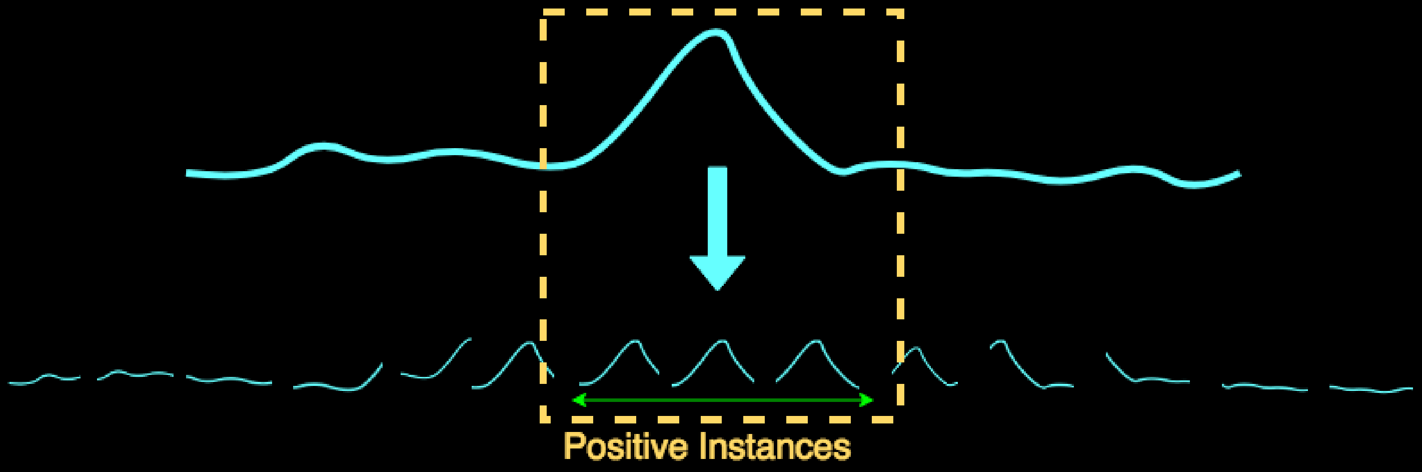
- Too much noise.
- Ignores temporal structure of the data

~~Traditional Multi Instance Learning (MIL)~~

Standard techniques don't apply.

- Heterogenous.
- Ignores temporal structure of the data.

How ?



Exploit temporal locality with MIL/Robust learning techniques

Property 1: Positive instances are clustered together.

Property 2: Number of positive instances can be estimated.

Algorithm: MI-RNN

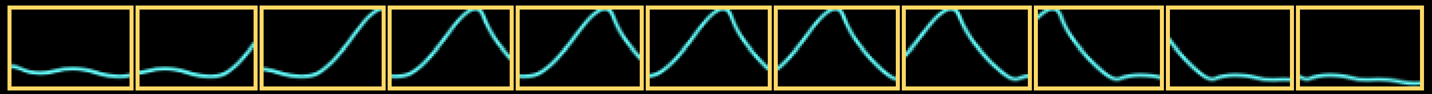
Two phase algorithm – alternates between identifying positive instances and training on the positive instances.

Algorithm: MI-RNN

- Step 1:
  Assign labels

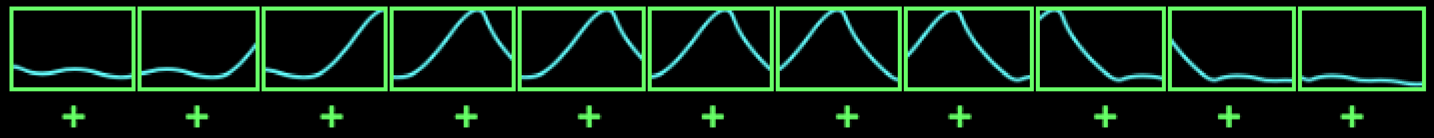
  Instance = source data

Algorithm: MI-RNN



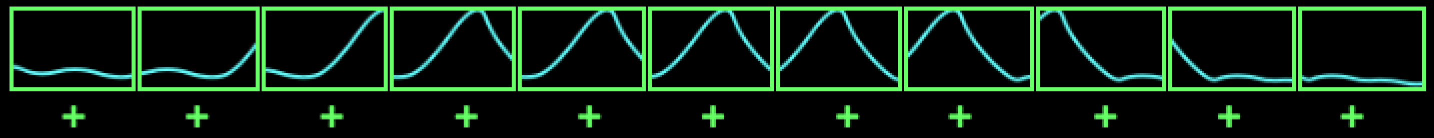
- Step 1:
  Assign labels
  Instance = source data

Algorithm: MI-RNN



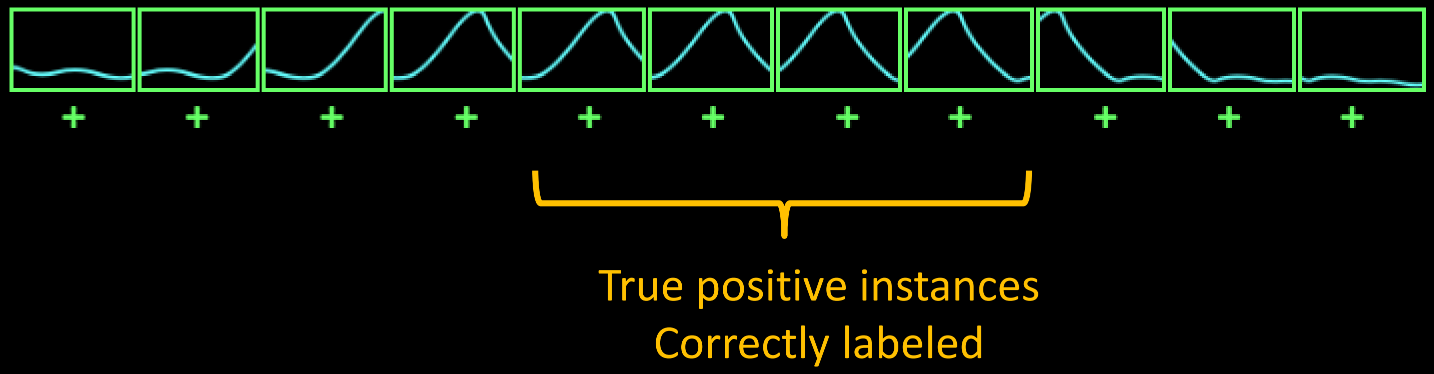
- Step 1:
  Assign labels
  Instance = source data

Algorithm: MI-RNN



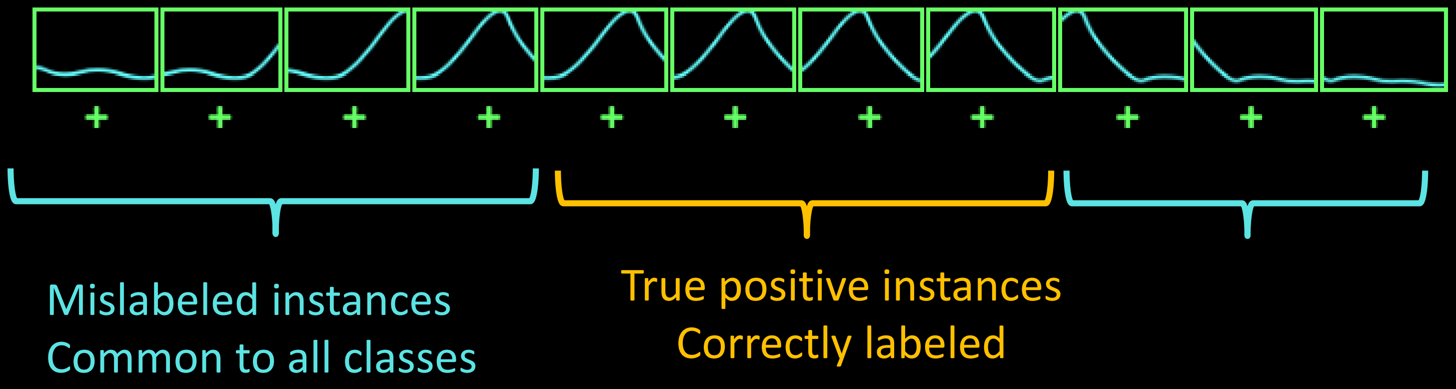
- Step 2:
  Train classifier on this data

Algorithm: MI-RNN



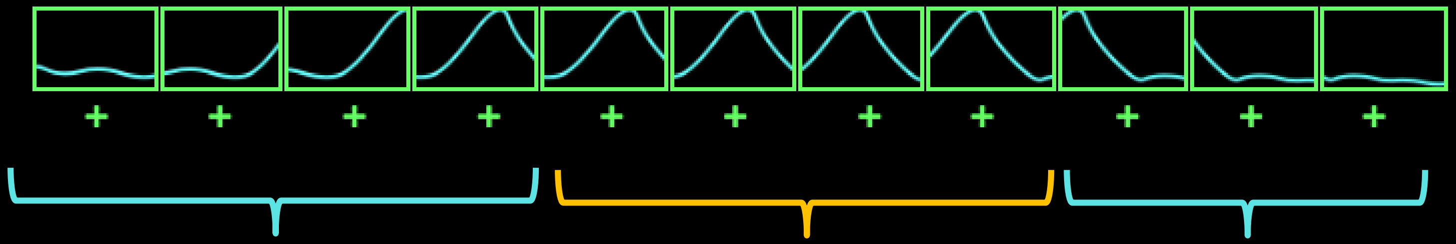
- Step 2:
  Train classifier on this data

Algorithm: MI-RNN



- Step 2:
  Train classifier on this data

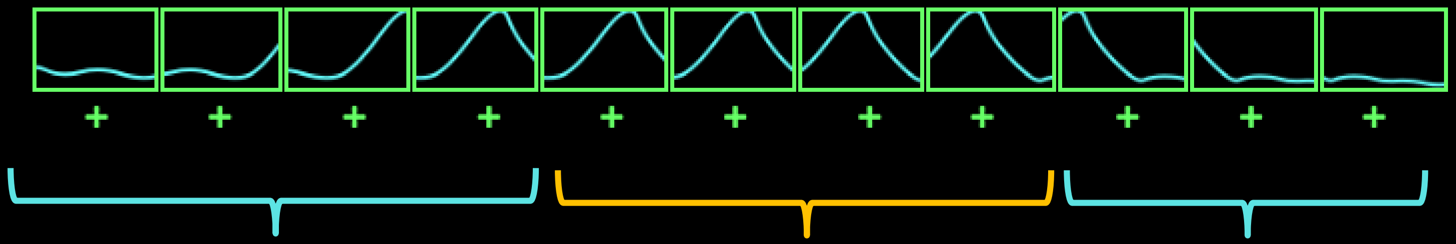
Algorithm: MI-RNN



Common to all classes

- Step 2:
  Train classifier on this data

Algorithm: MI-RNN

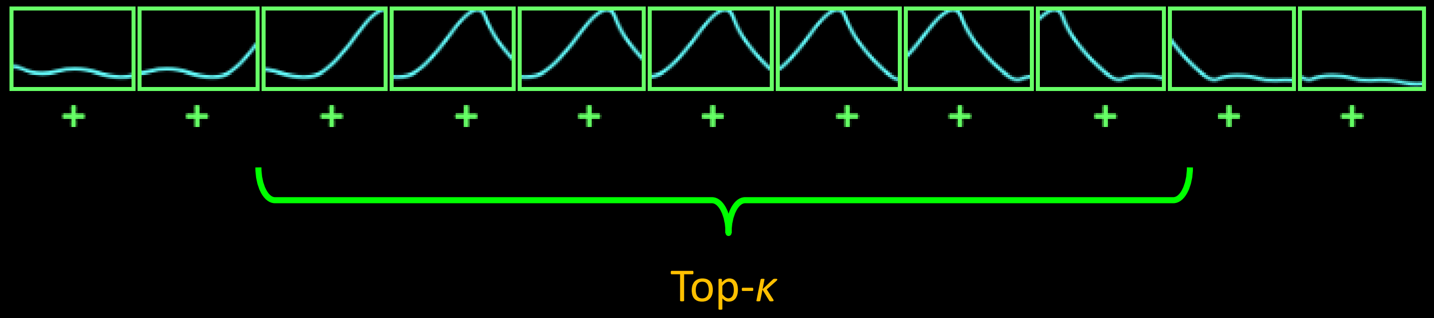


Common to all classes

- Step 2:
  Train classifier on this data

Classifier will be confused.
Low prediction confidence.

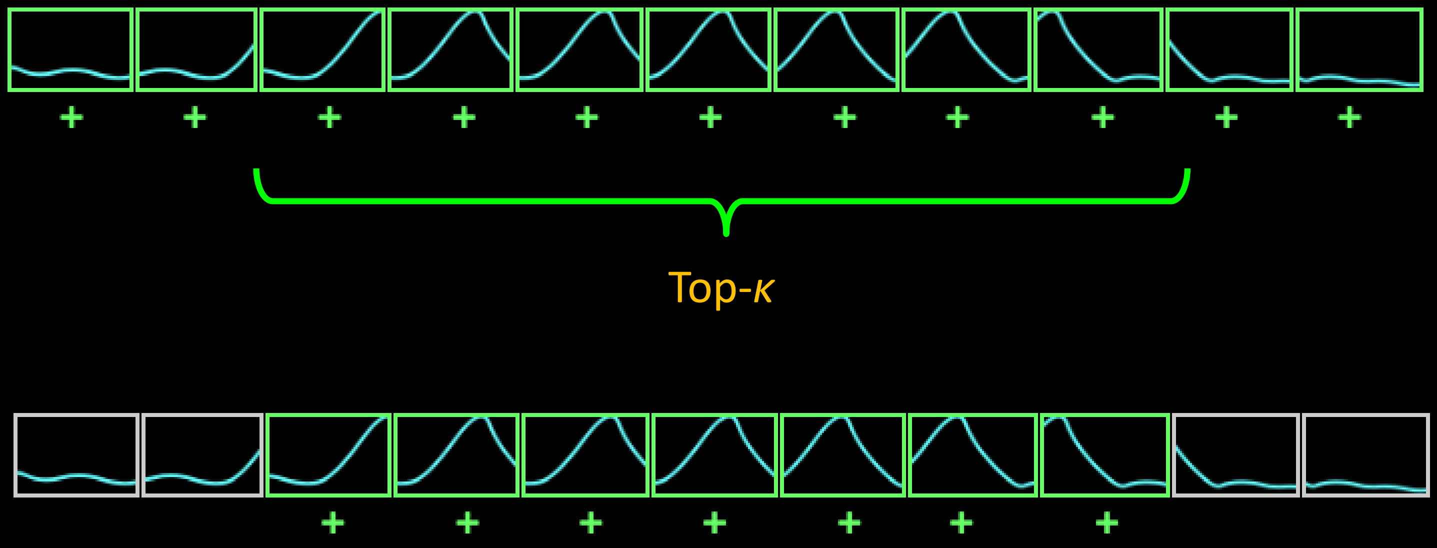
Algorithm: MI-RNN



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

  Should satisfy property 1 and property 2

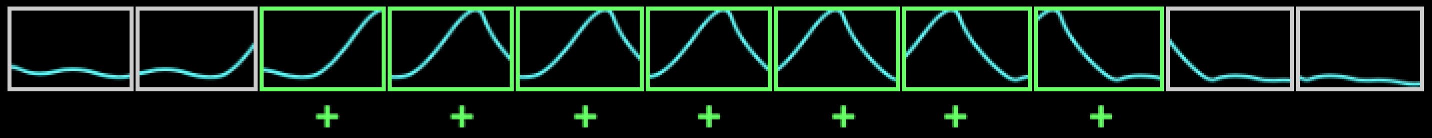
Algorithm: MI-RNN



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

  Should satisfy property 1 and property 2

Algorithm: MI-RNN



- Step 4:
  Repeat with new labels

MI-RNN: Does It Work?

MI-RNN: Does It Work?

- Of course!

MI-RNN: Does It Work?

- Of course!
- Theoretical analysis:

  Convergence to global optima in linear time for *nice* data

MI-RNN: Does It Work?

- Of course!
- Theoretical analysis:

  Convergence to global optima in linear time for *nice* data
- Experiments:

  Significantly improve accuracy while saving computation
  - Various tasks: activity recognition, audio keyword detection, gesture recognition

MI-RNN: Does It Work?

Dataset	Hidden Dim	LSTM	MI-RNN	Savings %
HAR-6 *(Activity detection)*	8	89.54	91.92	62.5
	16	92.90	93.89	
	32	93.04	91.78	
Google-13 *(Audio)*	16	86.99	89.78	50.5
	32	89.84	92.61	
	64	91.13	93.16	
WakeWord-2 (Audio)	8	98.07	98.08	50.0
	16	98.78	99.07	
	32	99.01	98.96	

MI-RNN: Does It Work?

Dataset	Hidden Dim	LSTM	MI-RNN	Savings %
HAR-6 *(Activity detection)*	8	89.54	91.92	62.5
	16	92.90	93.89	
	32	93.04	91.78	
Google-13 *(Audio)*	16	86.99	89.78	50.5
	32	89.84	92.61	
	64	91.13	93.16	
WakeWord-2 (Audio)	8	98.07	98.08	50.0
	16	98.78	99.07	
	32	99.01	98.96	

MI-RNN better than LSTM almost always

MI-RNN: Does It Work?

MI-RNN better than LSTM almost always

Dataset	Hidden Dim	LSTM	MI-RNN	Savings %
GesturePod-6 *(Gesture detection)*	8	-	98.00	50
	32	94.04	99.13	
	48	97.13	98.43	
DSA-19 *(Activity detection)*	32	84.56	87.01	28
	48	85.35	89.60	
	64	85.17	88.11	

MI-RNN: Savings?

Dataset	Hidden Dim	LSTM	Hidden Dim	MI-RNN	Savings	Savings at ~1% drop
HAR-6	32	93.04	16	93.89	10.5x	42x
Google-13	64	91.13	32	92.61	8x	32x
WakeWord-2	32	99.01	16	99.07	8x	32x
GesturePod-6	48	97.13	8	98.00	72x	-
DSA-19	64	85.17	32	87.01	5.5x	-

MI-RNN: Savings?

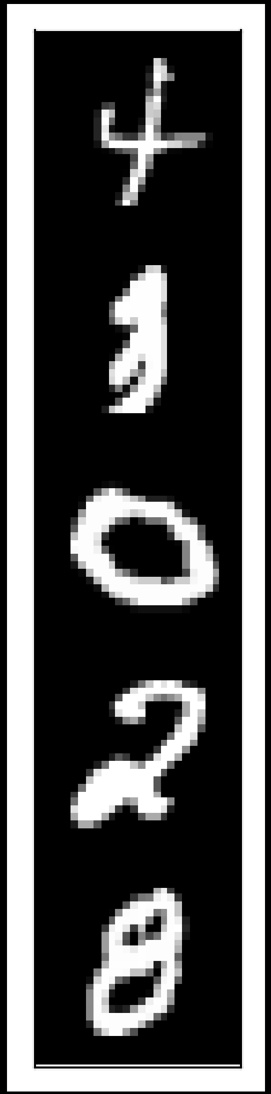
Dataset	Hidden Dim	LSTM	Hidden Dim	MI-RNN	Savings	Savings at ~1% drop
HAR-6	32	93.04	16	93.89	10.5x	42x
Google-13	64	91.13	32	92.61	8x	32x
WakeWord-2	32	99.01	16	99.07	8x	32x
GesturePod-6	48	97.13	8	98.00	72x	-
DSA-19	64	85.17	32	87.01	5.5x	-

MI-RNN achieves same or better accuracy with ½ or ¼ of LSTM hidden dim.

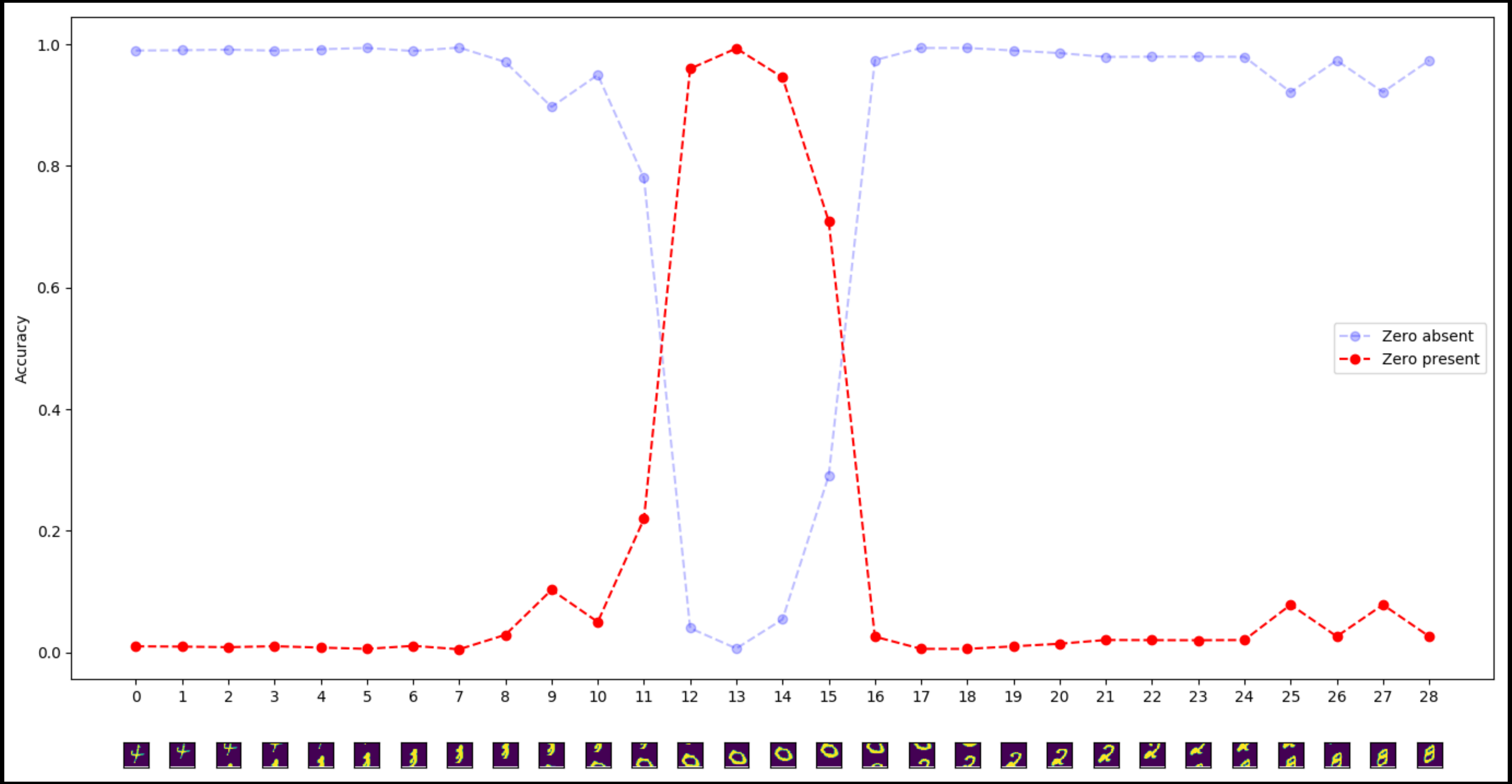
MI-RNN in Action

Synthetic MNIST:

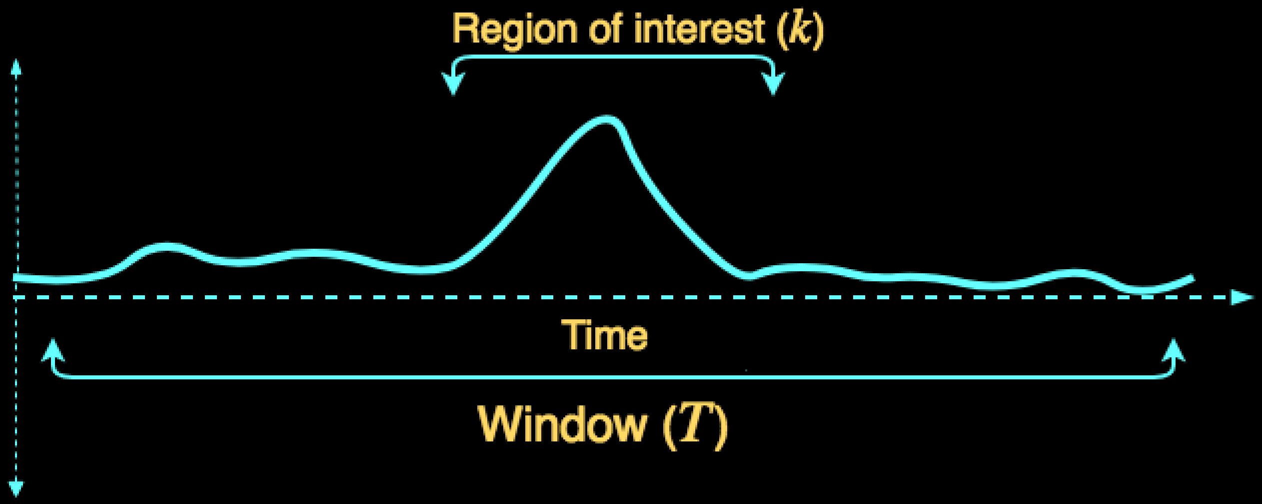
Detecting the presence of Zero.



MI-RNN in Action



RNNs are Expensive



Solution ?

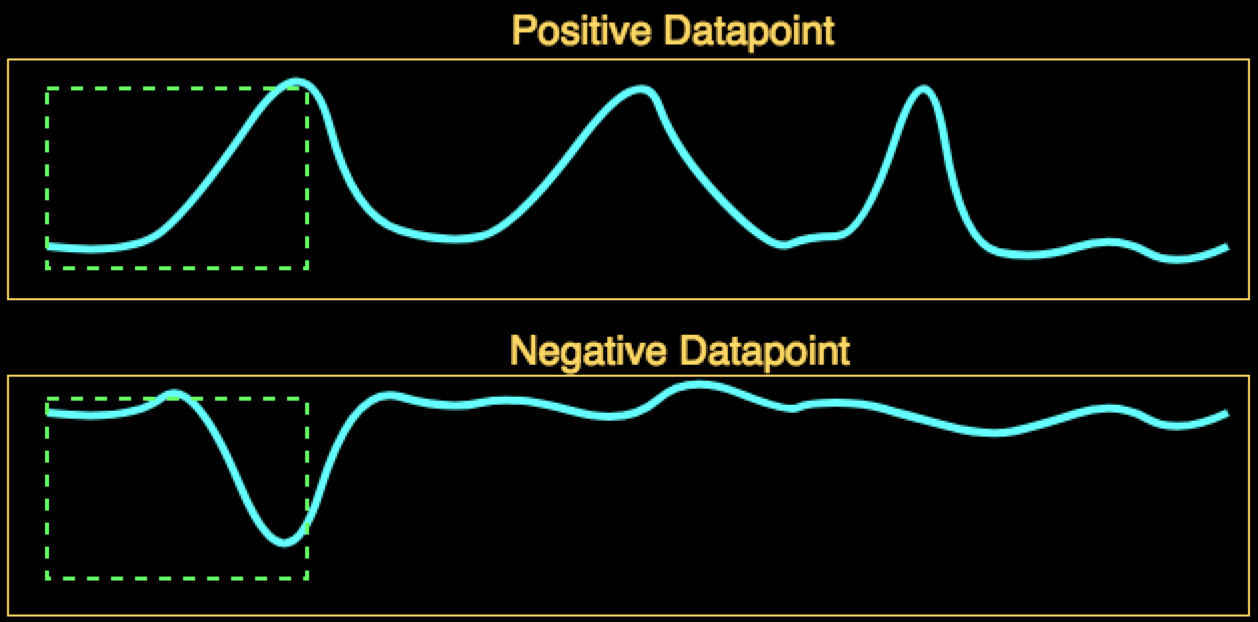
Approach 2 of 2 : Early Prediction

How?

Can we do even better?

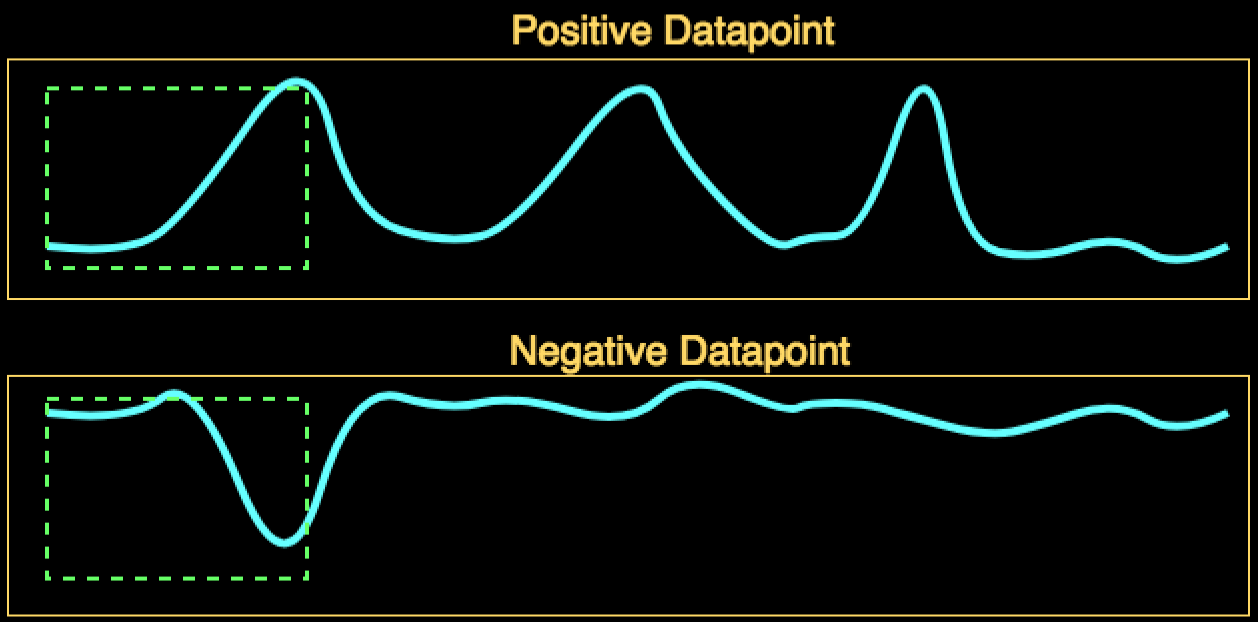
- For a lot of cases, looking only at a small prefix is enough to classify/reject.

Early Prediction



Can we do even better?

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



- Not feasible!

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.

Algorithm: E-RNN

Regular Loss: $$L(X, y) = (W^\top h_T - y)^2$$

Early Loss: $$L_e(X, y) = \sum_{t=1}^{T} (W^\top h_t - y)^2$$

Algorithm: E-RNN

Regular Loss: $$L(X, y) = (W^\top h_T - y)^2$$

Early Loss: $$L_e(X, y) = \sum_{t=1}^{T} (W^\top h_t - y)^2$$

Incentivizes early and consistent prediction.

E-RNN: How well does it work?

E-RNN: How well does it work?

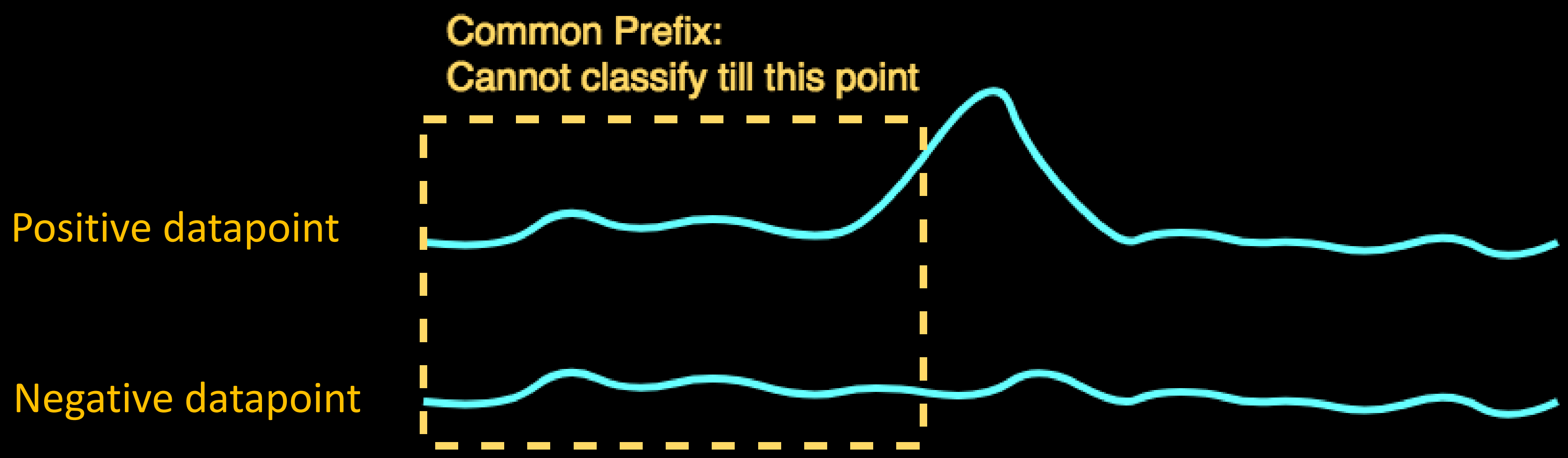
- Abysmally bad ☹

E-RNN: How well does it work?

- Abysmally bad ☹
- In GesturePod-6, we loose 10-12% accuracy attempting to predict early.

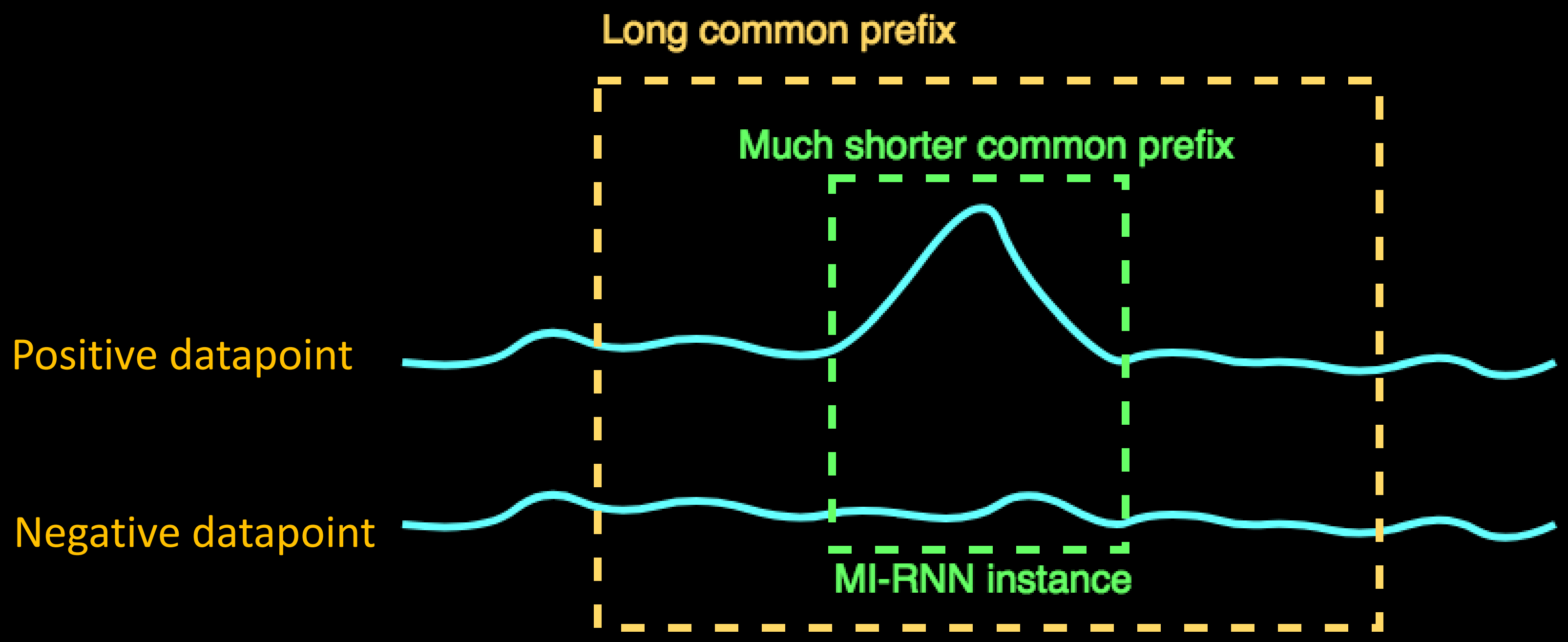
E-RNN: How well does it work?

- Abysmally bad ☹
- In GesturePod-6, we loose 10-12% accuracy attempting to predict early.
- Gets confused easily due to common prefixes!



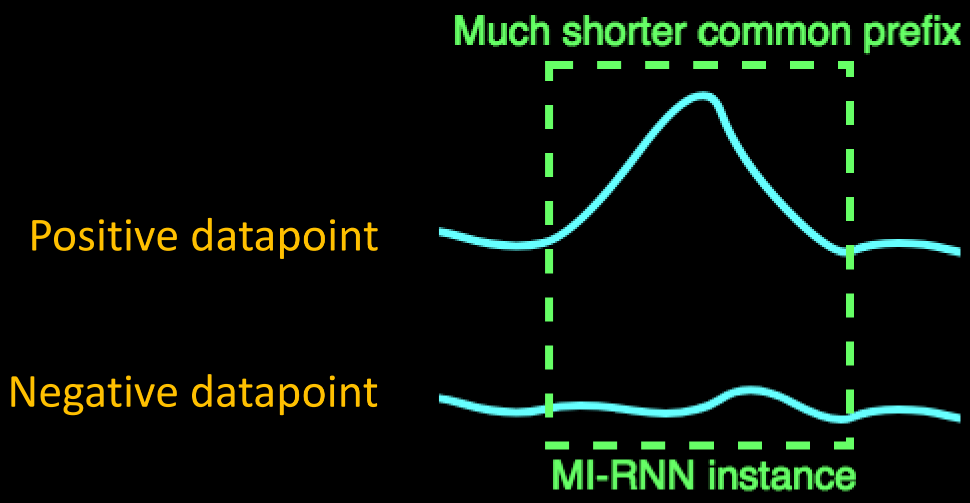
E-RNN: How well does it work?

- MI-RNN can help!
- Instances are very tight around signatures.



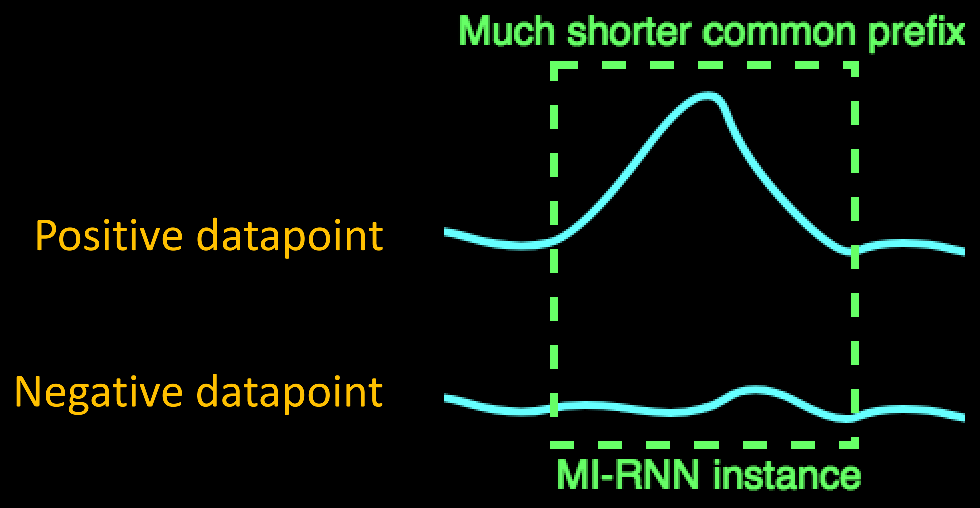
E-RNN: How well does it work?

- MI-RNN can help!
- Instances are very tight around signatures.



E-RNN: How well does it work?

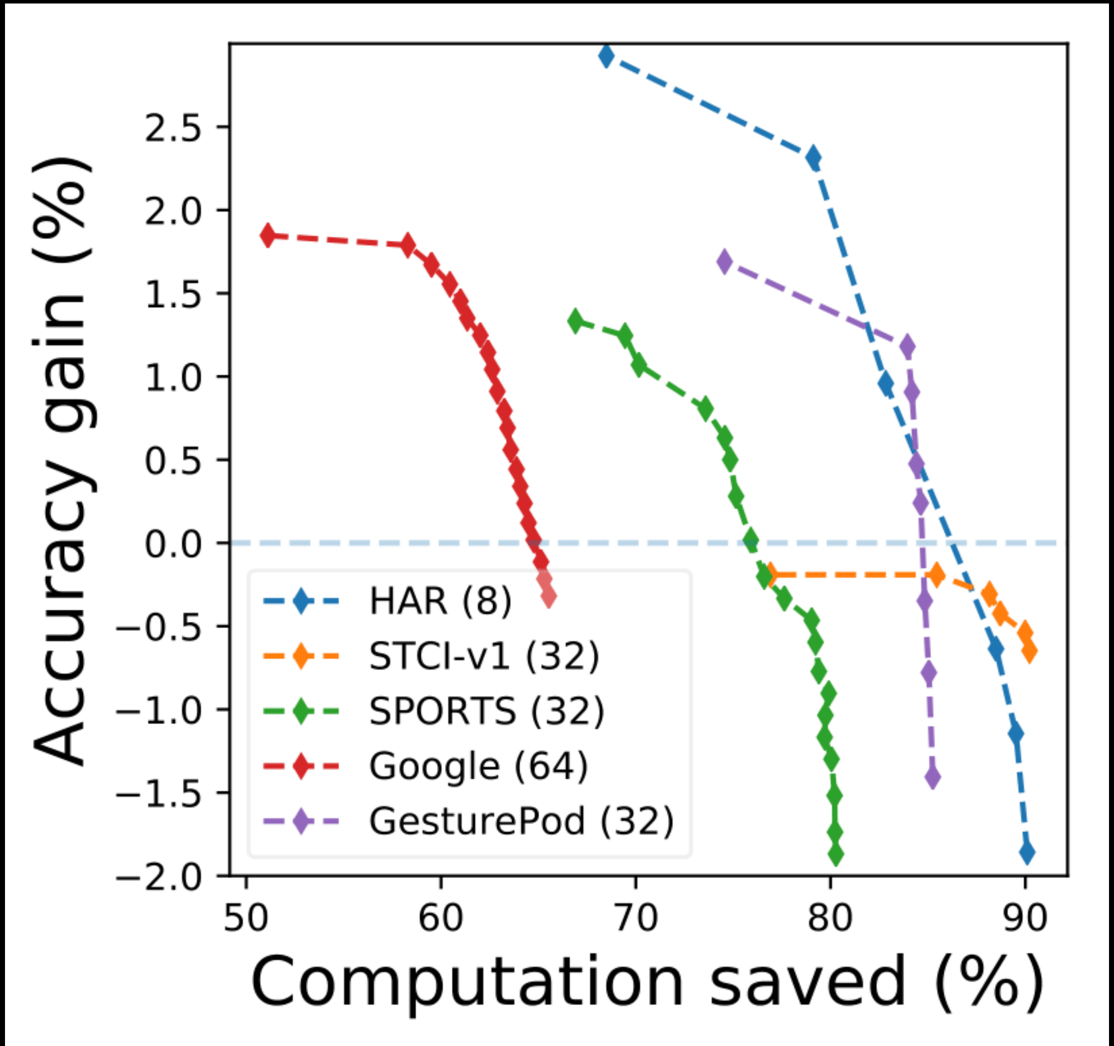
- MI-RNN can help!
- Instances are very tight around signatures.
- Low confusion - common prefixes are small.



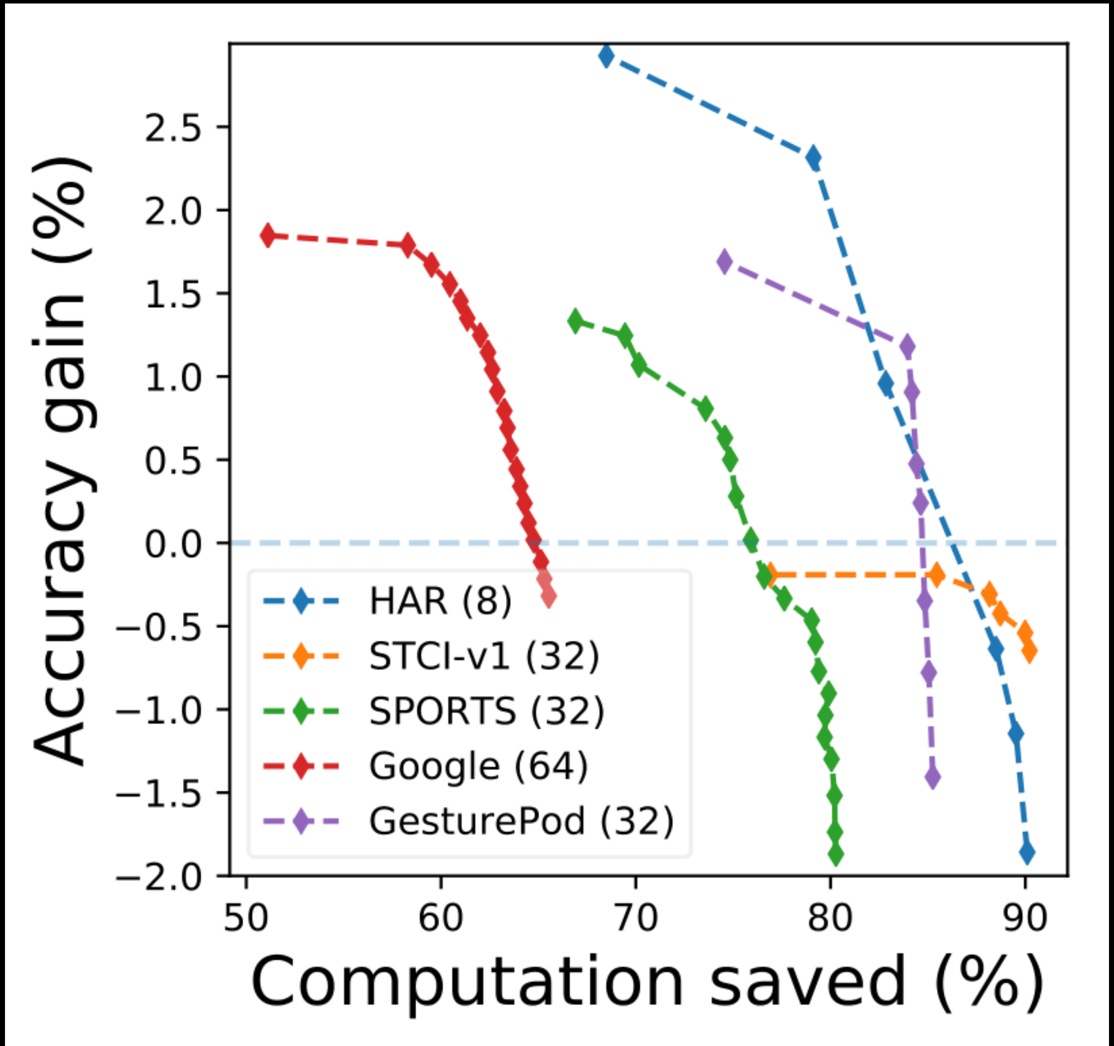
Algorithm: EMI-RNN

- Combine the MI-RNN training routine with E-RNN loss function and train jointly.
- Not only do you predict on smaller windows, but you predict early very often!

EMI-RNN: Results

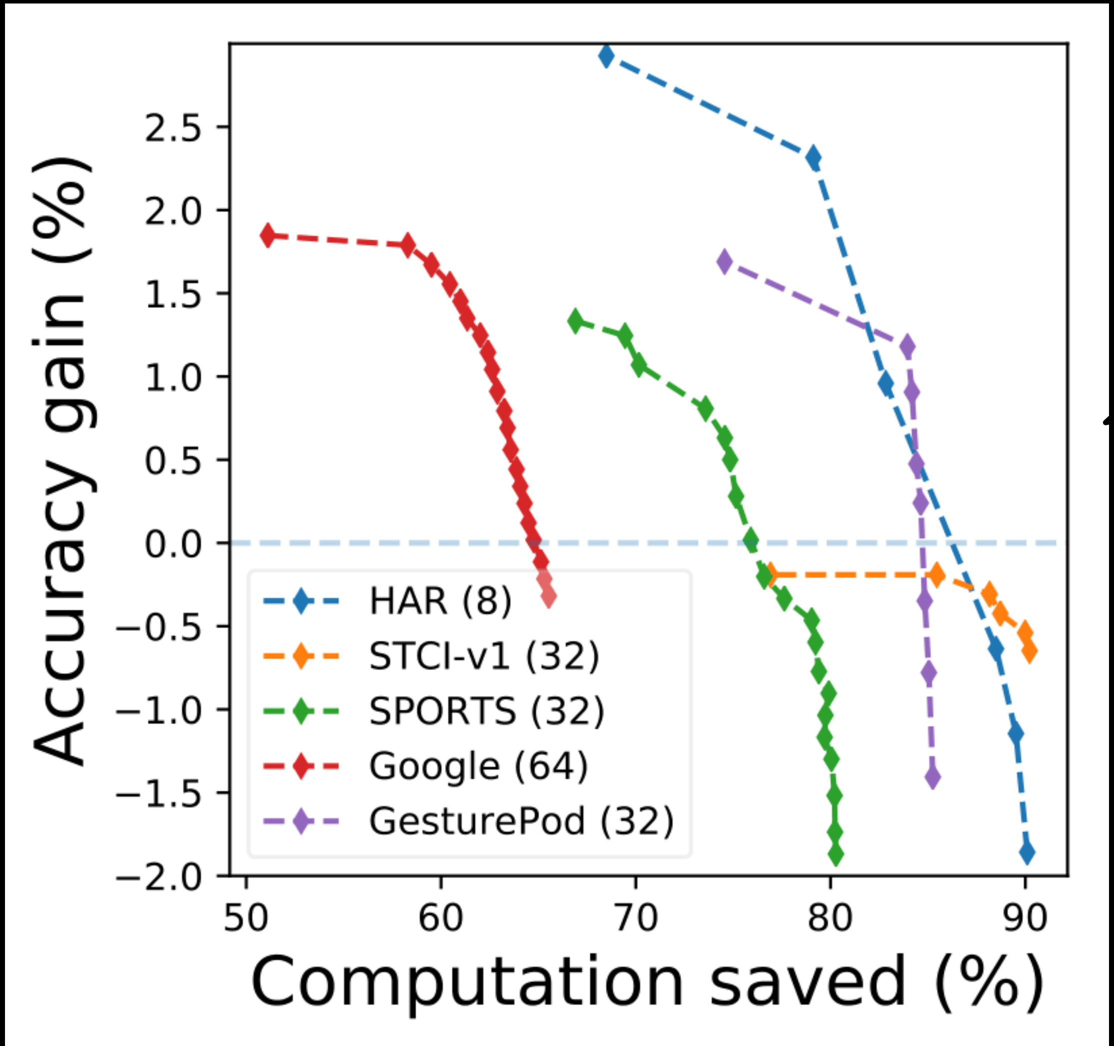


EMI-RNN: Results



For HAR-6, we are 8x faster at 8 hidden size wth better accuracy

EMI-RNN: Results



Comparing across hidden sizes, savings amplify by 4-16x

Raspberry Pi0

Device	Hidden Dim.	LSTM (ms)	MI-RNN (ms)	EMI-RNN (ms)
RPi0 (22.5 ms)	16	28.14	14.06	5.62
	32	74.46	37.41	14.96
	64	226.1	112.6	45.03
RPi3 (26.39 ms)	16	12.76	6.48	2.59
	32	33.10	16.47	6.58
	64	92.09	46.28	18.51

1GHz, Single-core CPU - 512MB RAM

Conclusions and Future Work

- 8x – 72x savings with MI-RNN. Additional savings from early prediction.
- Better or match LSTM performance.
- 10x performance gain away from Arduino class devices:
  - EMI-FastGRNN
  - Rolling LSTM

Thank You!

Next Talk

Support Recovery for Orthogonal Matching Pursuit: Upper and Lower Bounds

Somani et al., NIPS ’18