An Open-World Time-Series Sensing Framework for Embedded Edge Devices

ABDULRAHMAN BUKHARI, SEYEDMEHDI HOSSEINIMOTLAGH AND HYOSEUNG KIM

UNIVERSITY OF CALIFORNIA, RIVERSIDE

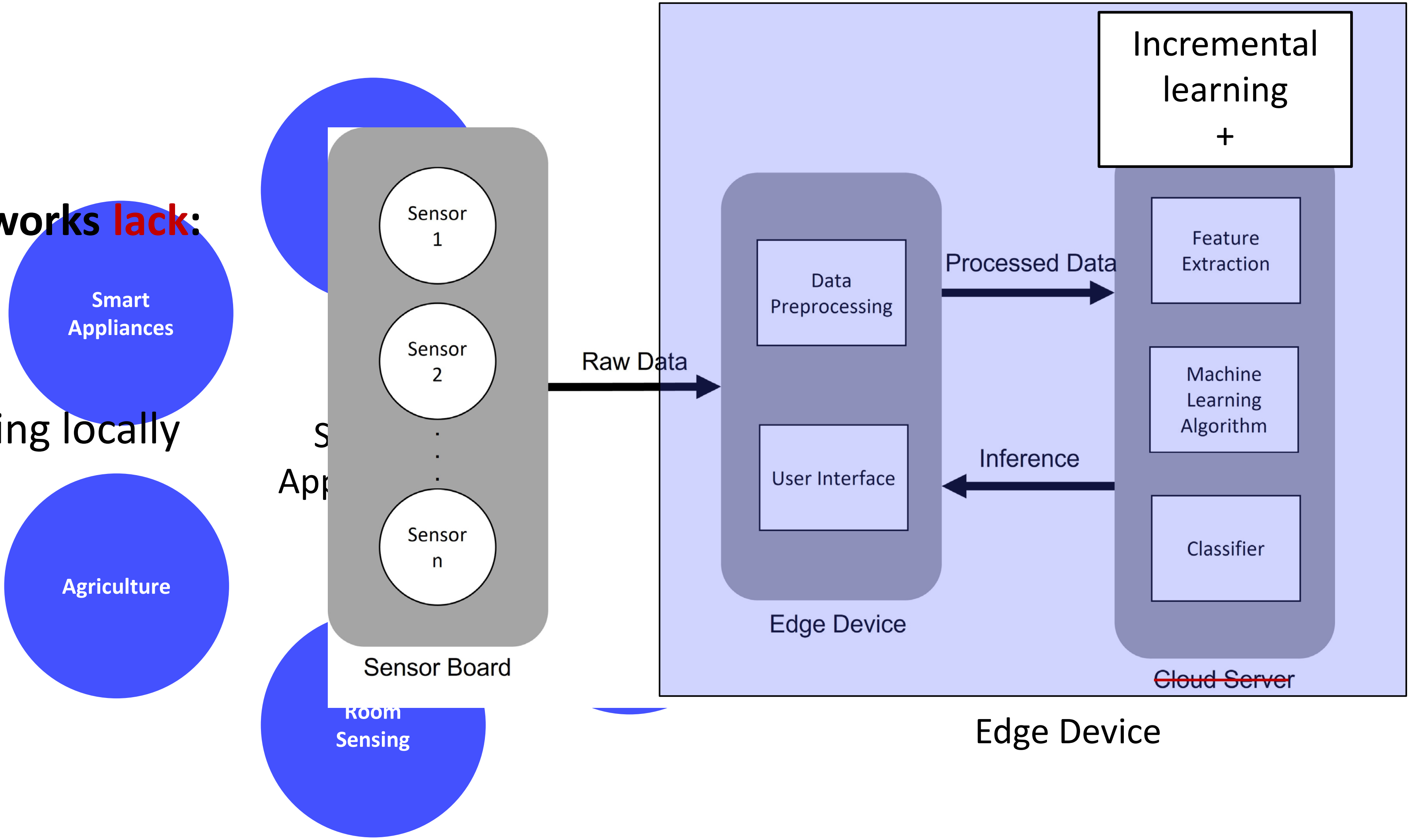
{ABUKH001, SHOSS007, HYOSEUNG}@UCR.EDU

Motivational Example

Current sensing frameworks lack:

Incremental learning

Classification and learning locally



What is Incremental Learning

Supervised vs Open-world

Incremental learning: continuously learning new classes from a stream of data

COST: the classifier will forget old classes → **Catastrophic Forgetting**[1]

This is a **supervised** approach if the new classes are **labelled**

Unlabeled data from unknown classes → **Open-world problem**

Recognize unknown samples and cluster them into new classes

[1] McCloskey et al. Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. (1989)

Classification and Clustering the Unknowns

Classical classifier **cannot** recognize unknown samples

We can do that:

- Adding a threshold to the classifier output
- Use an open-world classifier
  - OpenMax[1]
  - Extreme Value Machine (EVM)[2]

To Cluster unknown samples

- Unsupervised clustering algorithms ➔ **FINCH**[3] Algorithm

[1] Bendale et al. Towards Open Set Deep Networks. (CVPR, 2016)
[2] Rudd et al The Extreme Value Machine. (2018)
[3] Saquib et al. Efficient parameter-free clustering using first neighbor relations. (CVPR, 2019)

Time-Series Sensing Data

Synthetic Sensors[1]

- A single sensor board can capture multiple environmental facets
- Can be deployed into different environments to recognize different sets of events

  Limitations ➔ requires access to a server for training and classification

DeepSense[2]

- A unified framework for time-series sensing data
- Achieve high inference performance for both classification and regression problems

Limitations ➔ network architecture changes based on #of sensors

BOTH

- Cannot incrementally learn new classes for a data-stream

[1] Laput et al. Synthetic Sensors: Towards General-Purpose Sensing. (HFCS, 2017)
[2] Yao et al. DeepSense: A Unified Deep Learning Framework for Time-Series Mobile Sensing Data Processing. (ICWWW, 2017)

Incremental Learning

(Supervised) Incremental Classifier and Representation Learning[1]

- Sets of exemplars to represent previously learned classes
- The model is updated with (**new samples + exemplars sets**)
- Fixed-Representation class incremental learning (FRCI)

(Unsupervised) Open-World Learning without Labels (OWL)[2]

- Uses Extreme Value Machine (EVM) as a classifier
- Cluster unknown samples using Finch Alg.

They only applied the algorithms on computer vision applications

Have not been extended to time-series sensing data
Or evaluated on embedded edge devices

[1] Rebuffi et al. iCaRL: Incremental Classifier and Representation Learning. (CVPR, 2017)
[2] Jafarzadeh et al. Open-World Learning Without Labels. (2020)

Contributions

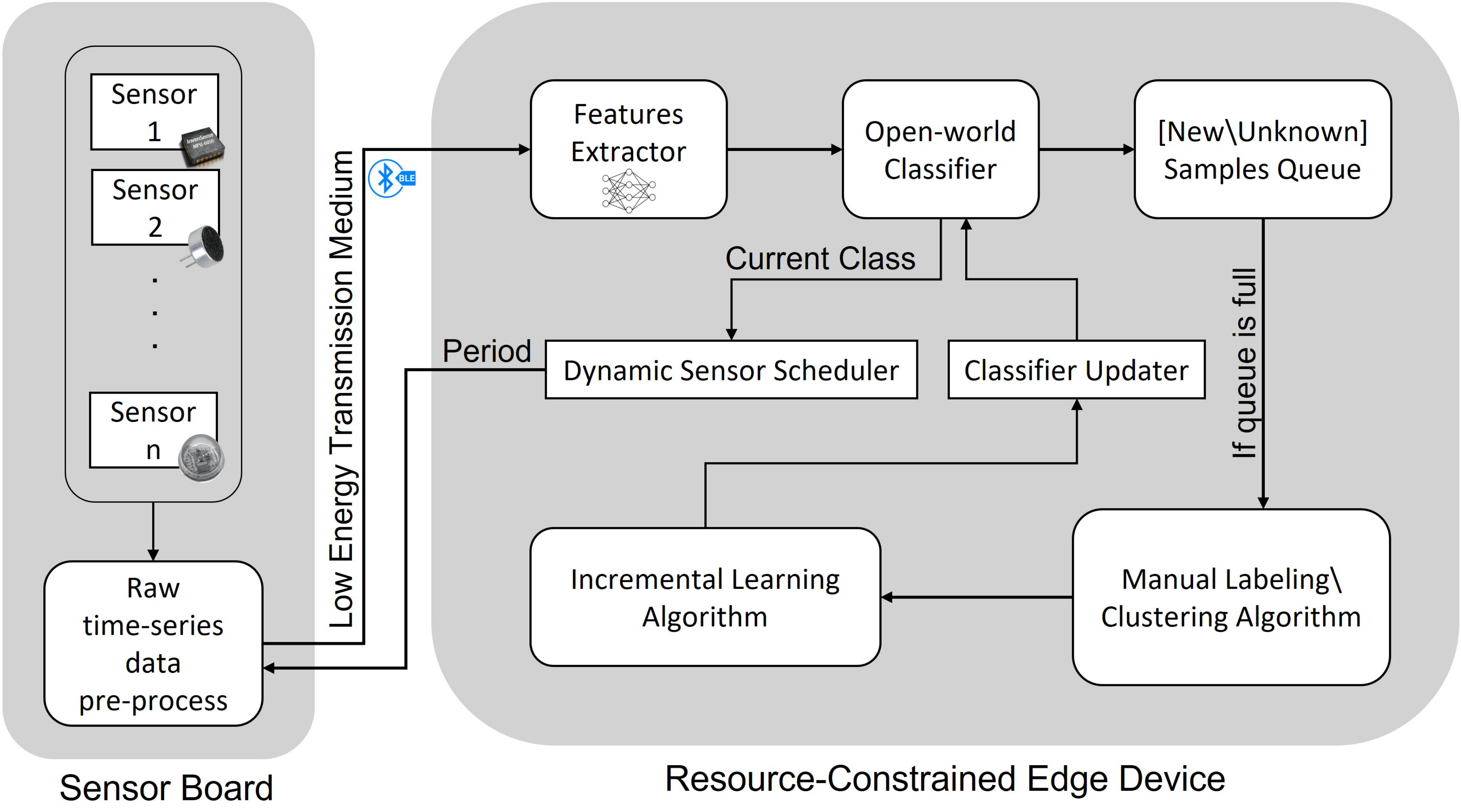
OpenSense: an open-world sensing framework for time-series data for embedded edge devices

- We present a **sensing framework** that can run different incremental learning algorithms for both supervised and unsupervised time-series sensing data problems
- We propose an **efficient DNN architecture** called sDNN, which outperforms the state-of-art architecture in both inference performance and resource efficiency for timeseries activity classification
- We demonstrate the implementation of **OpenSense** on a resource-constrained edge device and its effectiveness in open-world incremental learning of time-series data.

Outline

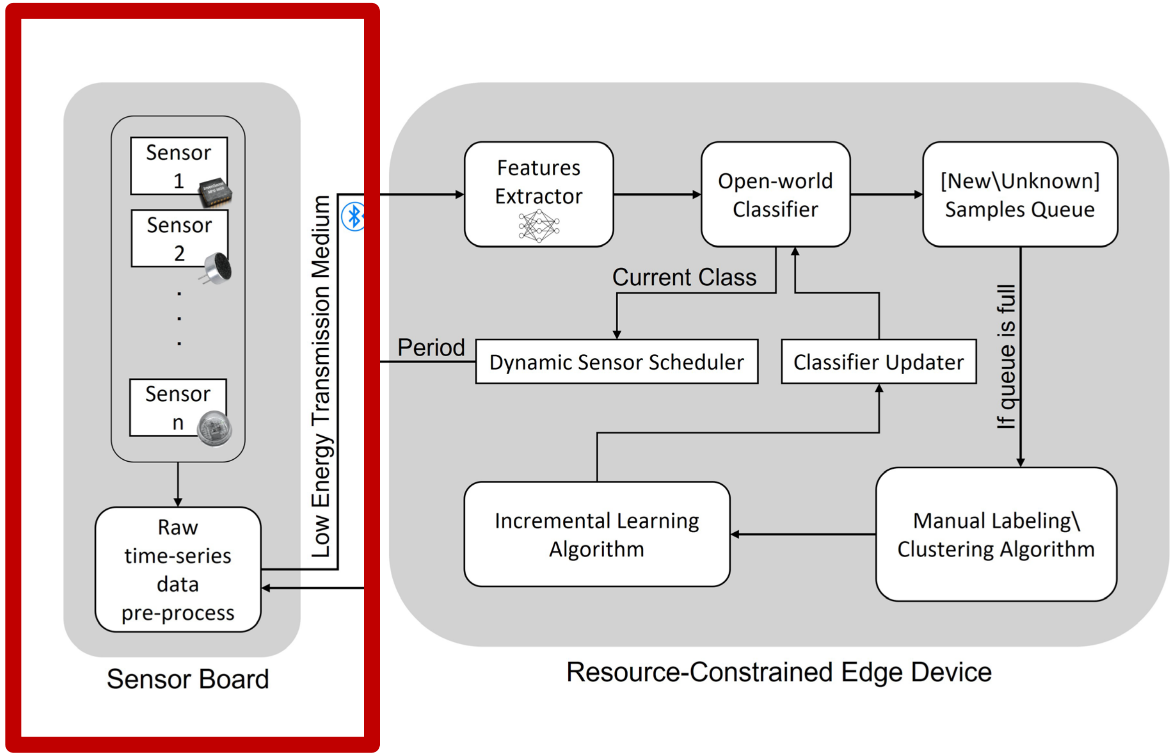
- Introduction
- **OpenSense Framework**
  - Sensor board
  - Embedded edge device
- **Evaluation**
  - Inference and learning performance
  - Latency and energy consumption performance
- **Conclusion**

Overview of OpenSense



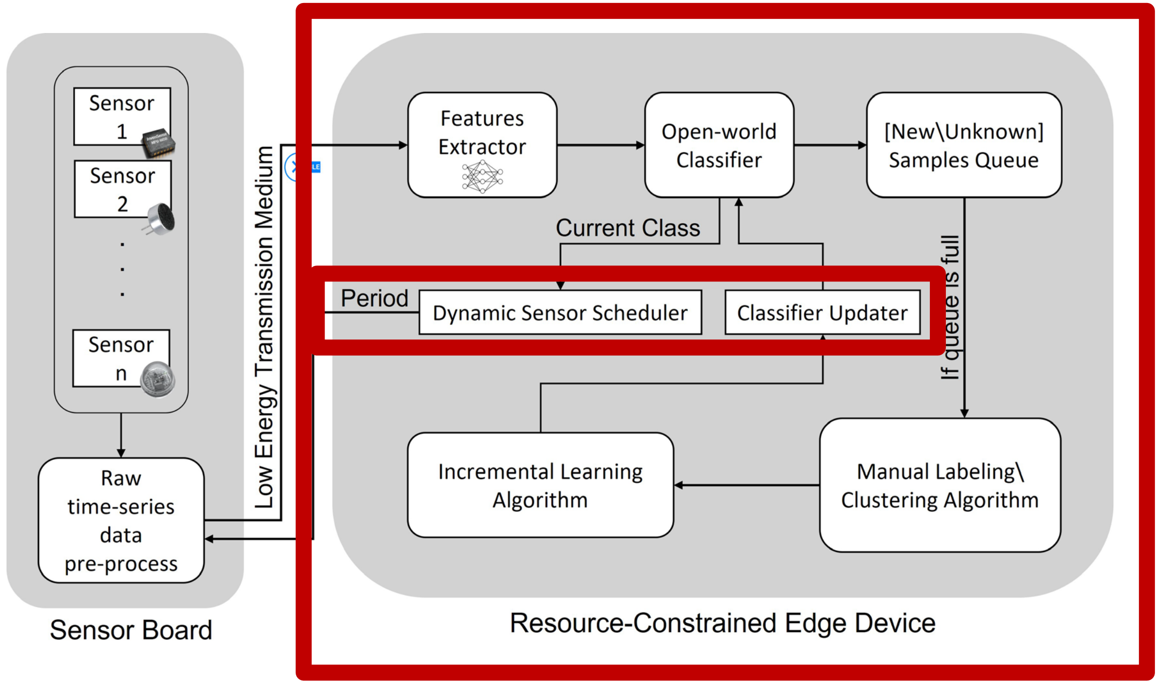
Sensor Board

- Collect raw data from sensors
- Preprocess them as time-series data
- Transmit the data periodically
- The period managed by the dynamic sensor scheduler



Embedded Edge Device

- Features extraction
- Classification
- Incremental learning
- Sensor period and model updating scheduling



Features Extractor

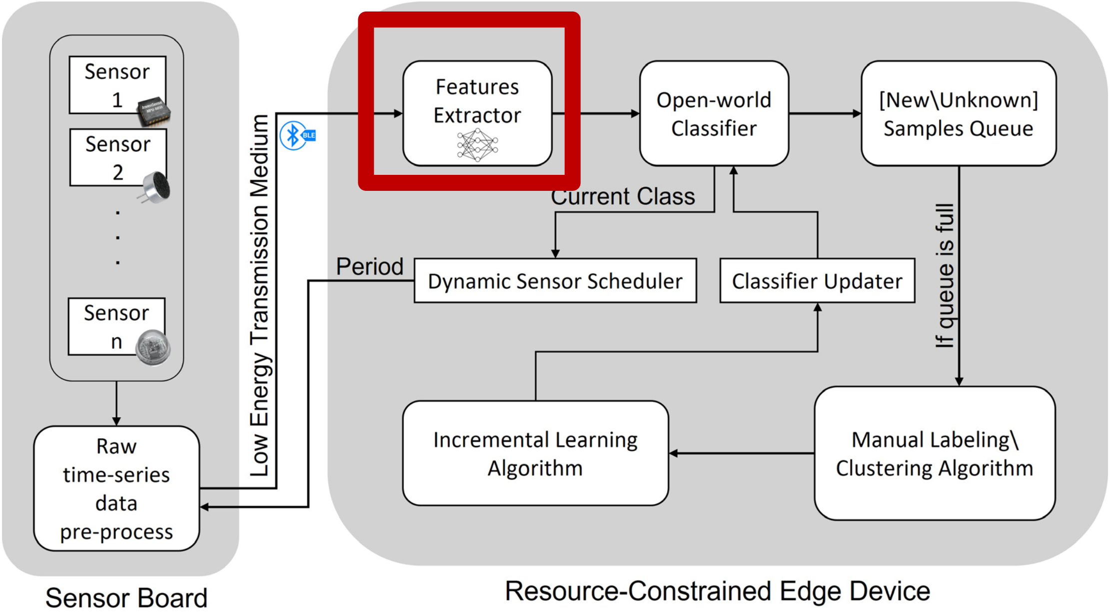
Further data preprocessing based on the sensor data:

- High-sampling rate ➔ Fast Fourier Transform
- Low-sampling rate ➔ Statistical information

The feature extractor is based on the DNN model

Features are extracted by taking the output data of the last layer before the output layer

We need a light-weight DNN model that extract reliable features at low computational cost



Our Proposed DNN

We call it a simple DNN (**sDNN**)

- Convolutional Layers ➔ spatial relationship
- LSTM Layers ➔ temporal relationship
- $\mathbf{f}$ is the filter size
- An output layer

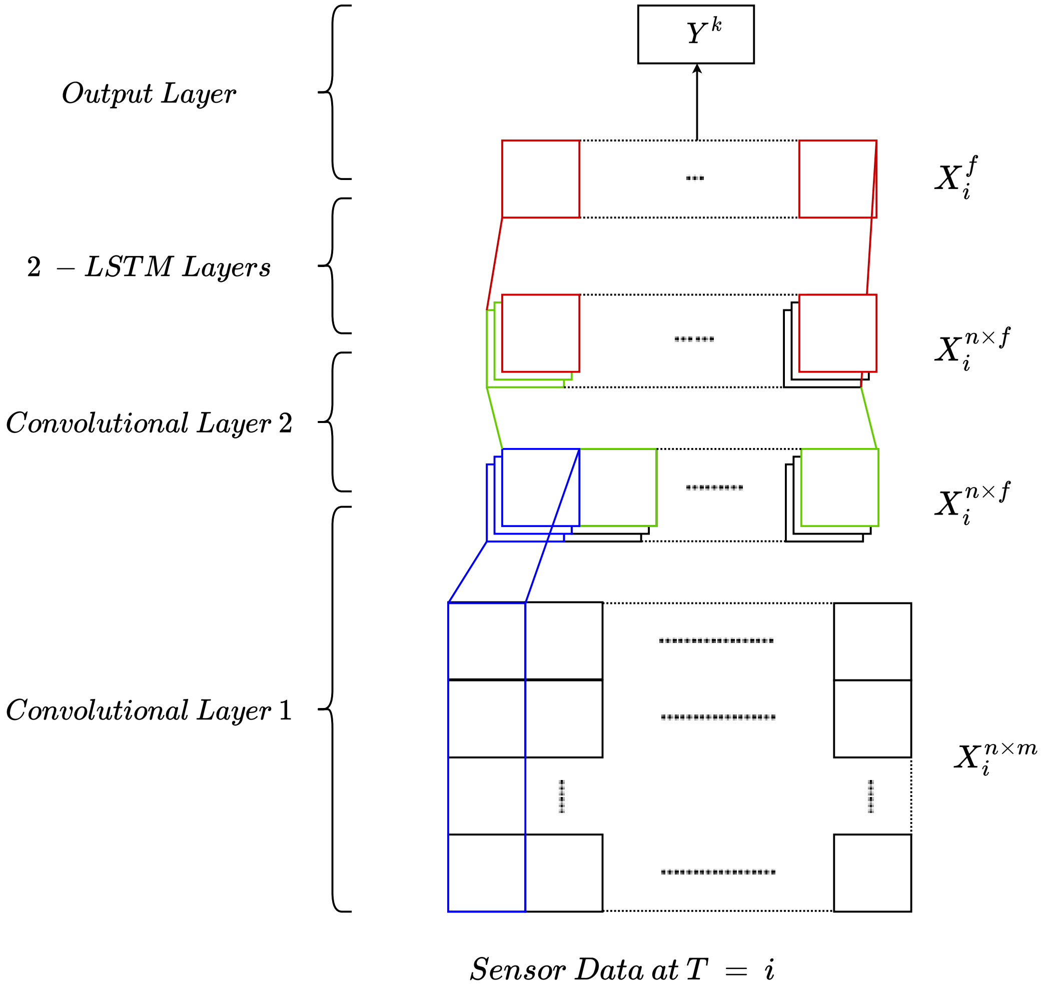
The size of input data $\boldsymbol{X}$ is $\boldsymbol{n} \times \boldsymbol{m}$

- $\boldsymbol{n}$ is the length of preprocessed time-series data
- These data are stacked $\boldsymbol{m}$ times to make a tensor $\boldsymbol{X}$

Input example:

- 1-second sensor data is divided into ($\boldsymbol{m}$=10) slices
- Each slice is 100ms
- Preprocessing a slice produce an $\boldsymbol{n}$-length vector

Feature extractor ➔ Last LSTM layer



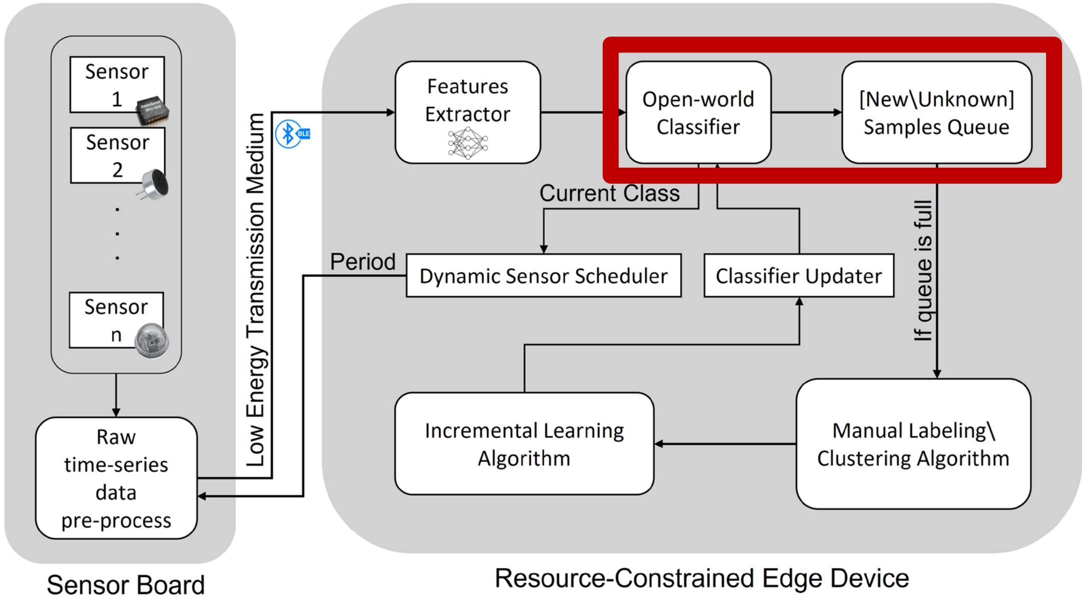
Open-World Classifier

Requirements for an open-world classifier:

1. Accurately classify samples from known classes
2. Recognize and reject unknown samples

For supervised approach ➔ a classical classifier

The rejected samples will be collected in a queue for incremental learning



Incremental Learning Algorithm

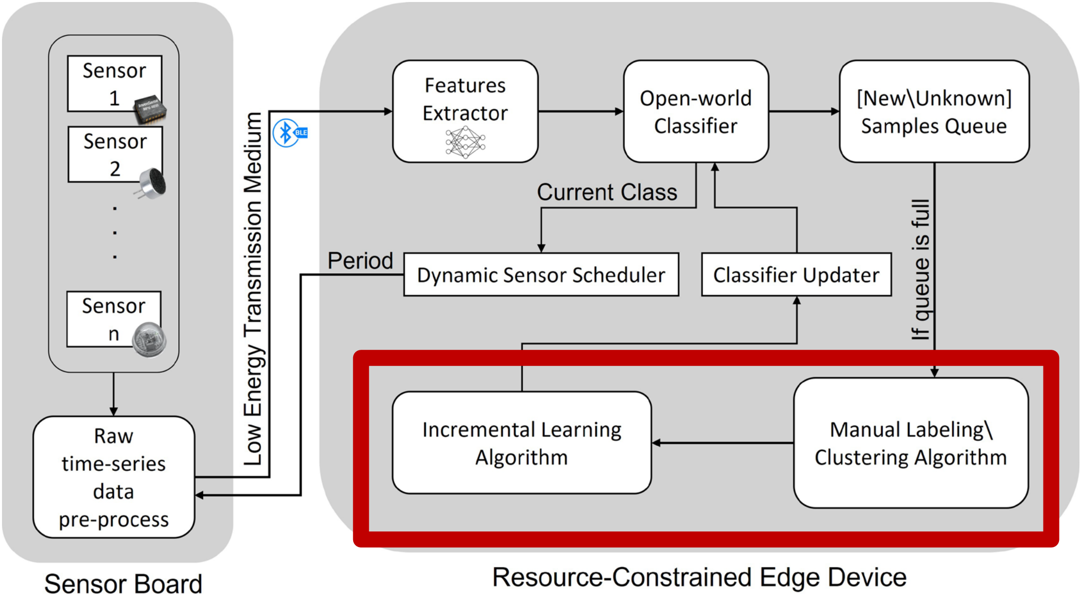
A true incremental learner must meet **3 criteria**:

1. Can be trained from a stream of data with new classes
2. The inference performance must stay competitive
3. Updating the model must meet the resource requirements of the system

We evaluated **three algorithms** on our framework:

- The naïve approach (NA)
- Fixed-Representation class incremental learning[1] (FRCI)
- OpenSense based on EVM[2]

For unsupervised learning we use Finch algorithm to cluster the unknown samples

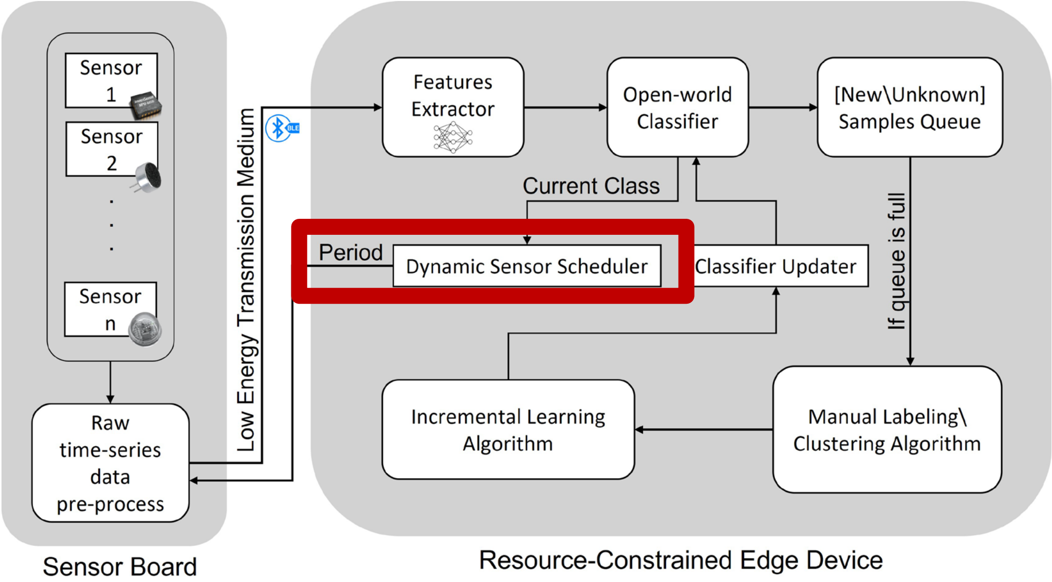


[1] Rebuffi et al. iCaRL: Incremental Classifier and Representation Learning. (CVPR, 2017)
[2] Jafarzadeh et al. Open-World Learning Without Labels. (2020)

Sensor Dynamic Scheduler

Goal: change the sensor data transmission period to:

- Reduce the energy consumption on the sensor board
- Increase the idle time ➔ free time for other tasks (e.g., learning)



Propose a Class-based Sensor Dynamic Scheduler:

The sensor data update period T_{sp} to detect an event **C** must meet the following condition:

$$n \times T_{sp} - T_e \leq CL$$

n: #T_{sp} repeated until a new event occurs
T_e: the time when the current event **C** ends
Classification Latency (CL) constraint: the maximum time for the current class to change to a different class while the sensor is idling

Algorithm 1

We propose a searching algorithm to find the maximum feasible sensor update period (T_{sp}) that does not exceed CL

Such as:

$$n \times T_{sp} - T_e \leq CL$$

1. Select minimum time among all time intervals for a class C as a base idle period T_{sp}
2. The base period is compared with all other time intervals such that the difference after n cycles does not exceed CL
3. If does not meet the condition ➔ decrement T_{sp} by 1

Algorithm 1: Sensor Dynamic Scheduler

```
Input  : T_e: All time intervals for class C
         CL: User allowable classification latency
Output: T_sp: Sensor idle period for class C
T_e ← AscendingSort(T_e)
T_sp ← FindMinimum(T_e)
L1 ← T_sp
L2 ← Length(T_sp)
i ← 0; j ← 0
for i ≤ L1 do
    for j ≤ L2 do
        n ← Ceil(T_e[j]/T_sp)
        Thresholds ← T_e[j] + CL
        if n × T_sp ≥ Threshold then
            T_sp ← T_sp − 1
            break
        end
        j + +
    end
    i + +
end
if T_sp > 1 then
    return T_sp
else
    return fail
end
```

Model and Classifier Updater

The model will be updated when number of samples of a new class meet the minimum requirement

BUT: resource-constraint edge devices cannot update the model with all new samples

We propose a model updating scheduler:
to partially update the model during the T_{sp} set by the dynamic scheduler

T_{min}: the minimum average time to train 1 sample

Algorithm 2: Model Update Scheduler

Input: T_{sp}: The sensor period for a given class
N_u: # of samples of the new discovered class
S_u: Samples from the new discovered class

$N_{old} \leftarrow 0$
while $N_u \neq 0$ **do**
  if $T_{sp} \geq T_{min}$ **then**
    $N_{ST} \leftarrow ComputeSamplesToTrain(T_{sp})$
    $UpdateTheModel(S_u[N_{old} : N_{ST}])$
    if $N_u \geq N_{ST}$ **then**
      $N_{old} \leftarrow N_{ST}$
    else
      return success
    end
  else
    return fail /* wait for next T_{sp} */
  end
end

Experiment Sets

Classification and learning performance

1. Compare DeepSense vs. Proposed sDNN vs. Proposed sDDN + EVM
2. Evaluate incremental learning algorithms in a supervised setting
3. Evaluate open-world learning algorithms in an unsupervised setting

Latency and energy consumption performance

4. Compare the execution time for different tasks from experiment #3
5. Run the open-world learning based on OWL-EVM on an embedded device and compare the execution time of different tasks
6. Evaluate the latency performance of the sensor dynamic scheduler
7. Evaluate the energy consumption of the sensor dynamic scheduler
8. The Model Updater Scheduler Performance

Evaluation

Evaluation platforms:

- Intel i7 with a dedicated NVIDIA GeForce GTX 1060 GPU [experiments: 1-4]
- Raspberry Pi 4 Model B with 2GB memory as an edge device [experiment: 5]
- TI CC2640R2 LAUNCHXL Board as a sensor board [experiment: 6-8]

Datasets:

- HHAR[1]: the Heterogeneous Human activity recognition (~120k samples)

  [Biking, Sitting, Standing, 'Walking, Stair Up and Stair down]
- PAMAP2[2]: Physical Activity Monitoring Data Set (~27k samples)

  [lying, sitting, standing, walking, running, cycling, Nordic walking, watching TV, computer work, car driving, ascending stairs, descending stairs, vacuum cleaning, ironing, folding laundry, house cleaning, playing soccer, rope jumping]

[1] Allan Stisen et al. Smart Devices are Different: Assessing and Mitigating Mobile Sensing Heterogeneities for Activity Recognition (SenSys, 2015)
[2] Reiss et al. Introducing a New Benchmarked Dataset for Activity Monitoring. (ISWC, 2012)

DeepSense vs. sDNN vs. sDNN + EVM

Objective: compare the inference performance

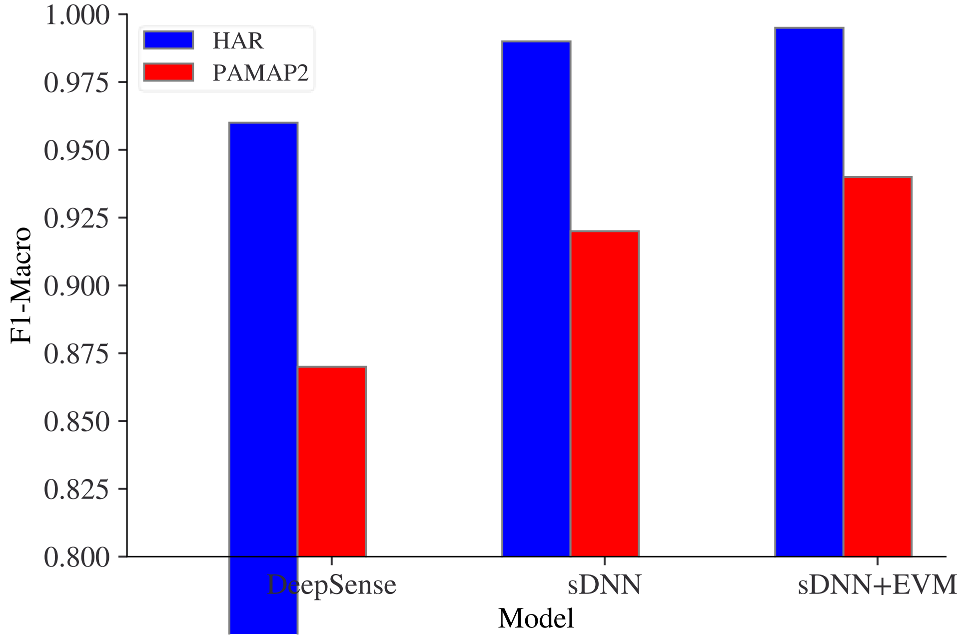
HHAR	➔	6 classes
PAMAP2	➔	18 classes

Metrics:

- Classification Accuracy
- F1-Macro score

The testing set is the same on all variants

Reason: DeepSense overfits the training dataset due to unnecessarily complex model



DeepSense vs. sDNN vs. sDNN + EVM

Objective: compare the training efficiency of each architecture

Dataset	HHAR		PAMAP2	
DNN Model	DeepSense	sDNN	DeepSense	sDNN
#epochs to converge	100	10	150	50
avg. execution time\epoch	37s	16s	18s	3s
speed-up\epoch	2.3x		6x	
total training time	61m 40s	2m 31s	45m	2m 30s

Supervised Incremental Learning

Objective: evaluate incremental learning algorithms:

1. The naïve approach (NA)
2. Fixed-Representation class incremental learning (FRCI)
3. OpenSense (Ours)

Initial training set

HHAR ➔ 2 classes **PAMAP2** ➔ 3 classes

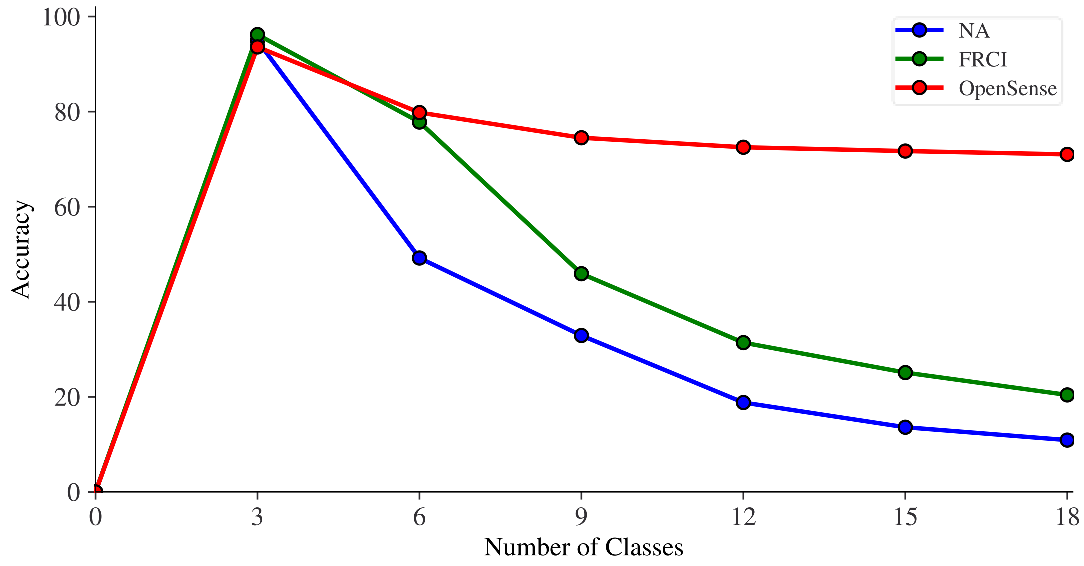
new classes in each data-stream

HHAR ➔ 2 classes **PAMAP2** ➔ 3 classes

Metrics:

- Classification Accuracy

The testing set add classes at each increment



PAMAP2 Dataset

Open-World Incremental Learning

Objective: evaluate the same incremental learning algorithms from the previous experiment in unsupervised setting

Initial training set

PAMAP2 ➔ 9 classes

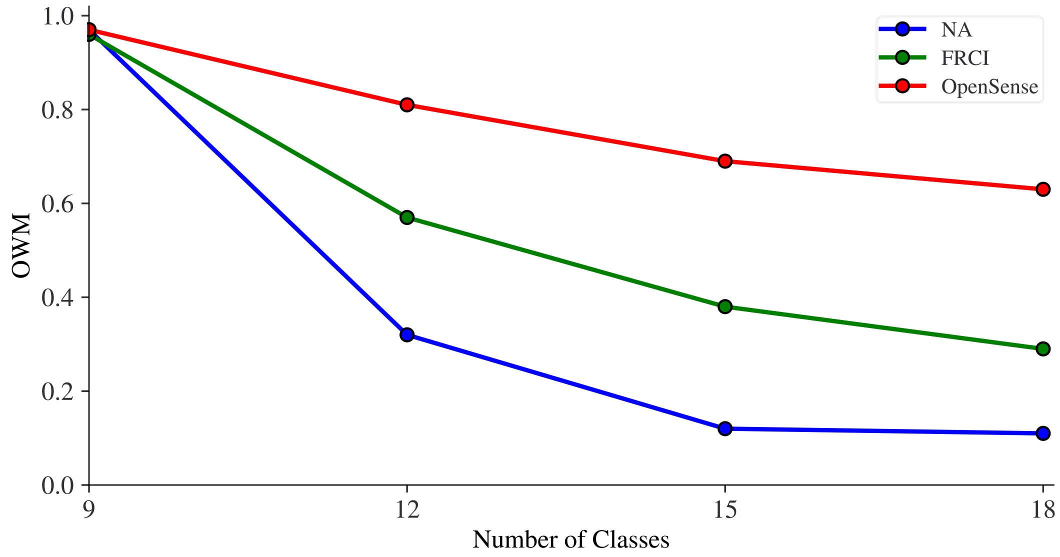
#unknown classes in each data-stream

PAMAP2 ➔ 3 classes

Metrics: Open-World Metric[1]

$$OWM = \frac{N_{KK}\ Acc(X_{KK}) + N_{UU}\ B3(X_{UU})}{N_{KK} + N_{KU} + N_{UK} + N_{UU}}$$

The testing set add classes at each increment



PAMAP2 Dataset

Algorithm	NA	FRCI	OpenSense
#discovered new classes	3/9 classes	5/9 classes	9/9 classes

[1] Jafarzadeh et al. Open-World Learning Without Labels. (2020)

Execution Time of Open-World Incremental Learning

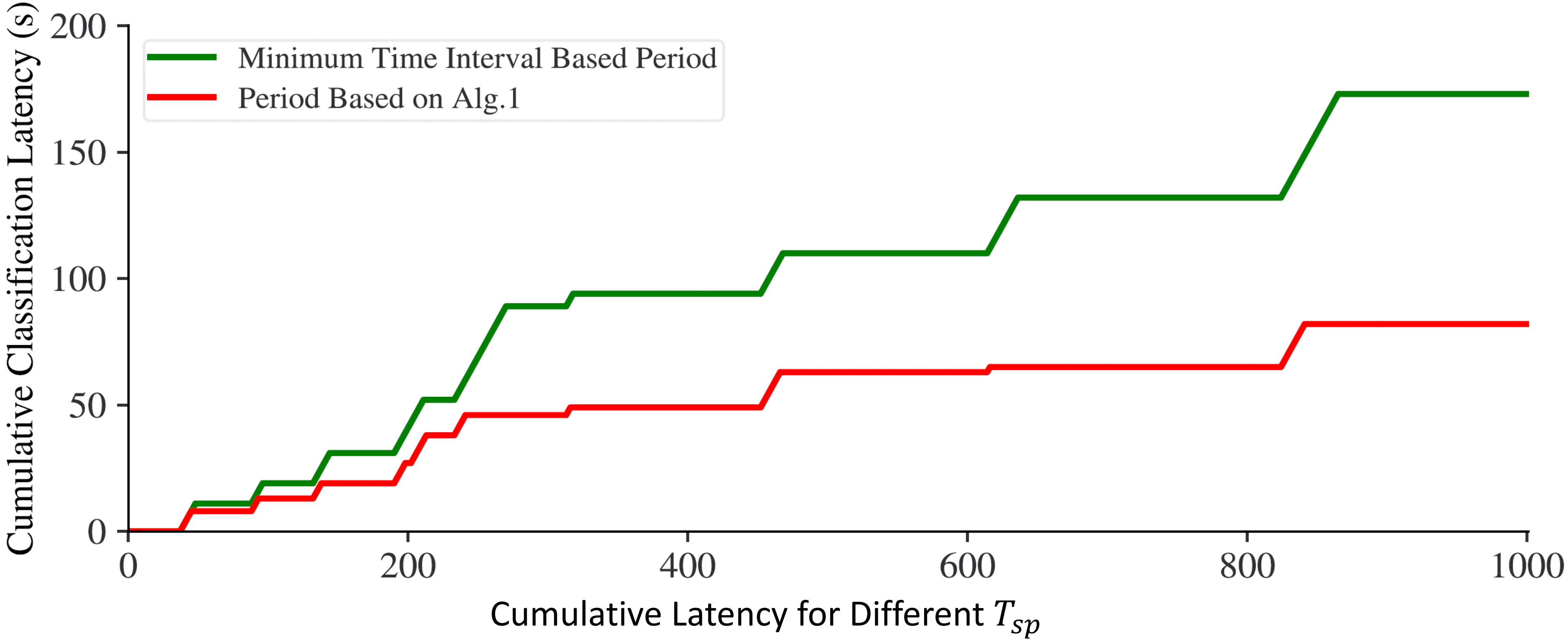
Objective: Compute the average execution time of different tasks in the framework from the previous experiment

All algorithms ran on the same training and testing datasets

Task	Inference			Incremental Learning		Total Session Time
	Feature Extraction	Classification	Queuing	Clustering	Model Updating	
NA	0.5s	12ms	16µs	0.46s	17.4s	67.8s
FRCI	0.47s	35ms	26µs	0.27s	16.5s	64.5s
OpenSense	0.49s	31ms	18µs	0.13s	0.92s	6.1s

Sensor Dynamic Scheduler Latency Performance

- We ran the sensor board for 1000s and capture each event for a random duration
- Assign T_{sp} for each class based on the history of each event
- Used different CL for each class



Conclusion

Proposed OpenSense Framework

- We evaluated different incremental learning algorithms on our framework
- OpenSense can successfully run on the resource-constraint edge device
- sDNN outperforms DeepSense on different datasets
- The proposed sensor dynamic scheduler and model updater scheduler make the framework efficiently runnable on resource-constraint edge devices

Future work

- Extend OpenSense to consider other resources on edge devices, e.g., accelerators

Thank you

Q & A

Algorithm 1

We propose a searching algorithm to find the maximum feasible sensor update period (T_{sp}) that does not exceed CL

Such as:

$$n \times T_{sp} - T_e \leq CL$$

1. Select minimum time interval among all time intervals for a class C as a base idle period T_{sp}
2. The base period is compared with all other time intervals such that the difference after n cycles does not exceed CL
3. If does not meet the condition ➔ decrement T_{sp} by 1

Algorithm 1: Sensor Dynamic Scheduler

Input : T_e: All time intervals for class C
CL: User allowable classification latency

Output: T_{sp}: Sensor idle period for class C

```
T_e ← AscendingSort(T_e)
T_sp ← FindMinimum(T_e)
L1 ← T_sp
L2 ← Length(T_sp)
i ← 0; j ← 0
for i ≤ L1 do
    for j ≤ L2 do
        n ← Ceil(T_e[j]/T_sp)
        Thresholds ← T_e[j] + CL
        if n × T_sp ≥ Threshold then
            T_sp ← T_sp − 1
            break
end
if T
else
    return fail
end
```

In the worst case where no feasible T_{sp} is found, the user may decide to set CL to the minimum value of one, which ensures T_{sp} to be at least two, i.e., CL = 1 and T_{sp} = 2

Example: Fixed T_{sp}

Example:

Time-intervals history for the following events:

A: [5, 9, 11, 16, 19] seconds
B: [3, 7, 15, 20, 23] seconds
C: [10, 18, 23, 31, 39] seconds

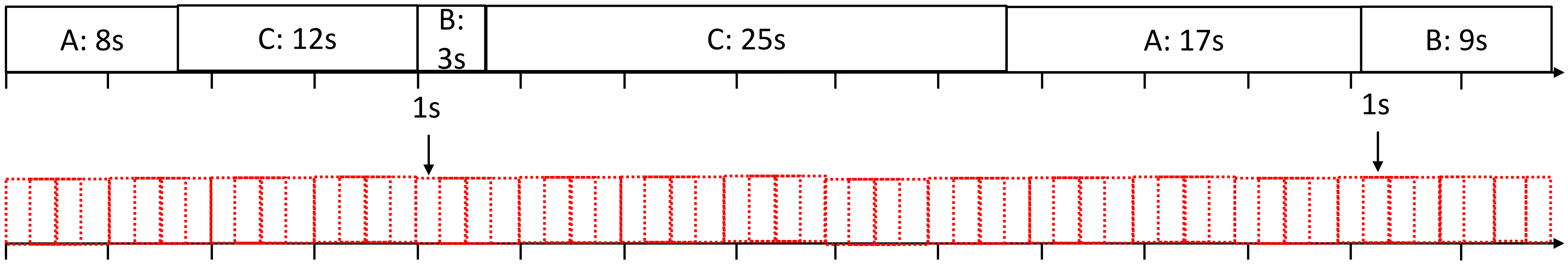
If the following sequence of events occurred and the user set CL = 2 seconds

Naive approach 1: Focus on satisfying CL at T_{sp} = 2 for all events

#Sensor transmission = 39 times

Total classification latency = 2 sec

Missing CL = 0 times



Example: Minimum Interval T_{sp}

Example:

Time-intervals history for the following events:

A: [5, 9, 11, 16, 19] seconds
B: [3, 7, 15, 20, 23] seconds
C: [10, 18, 23, 31, 39] seconds

If the following sequence of events occurred and the user set CL = 2 seconds

Naïve approach 2: Focus on maximizing idle time

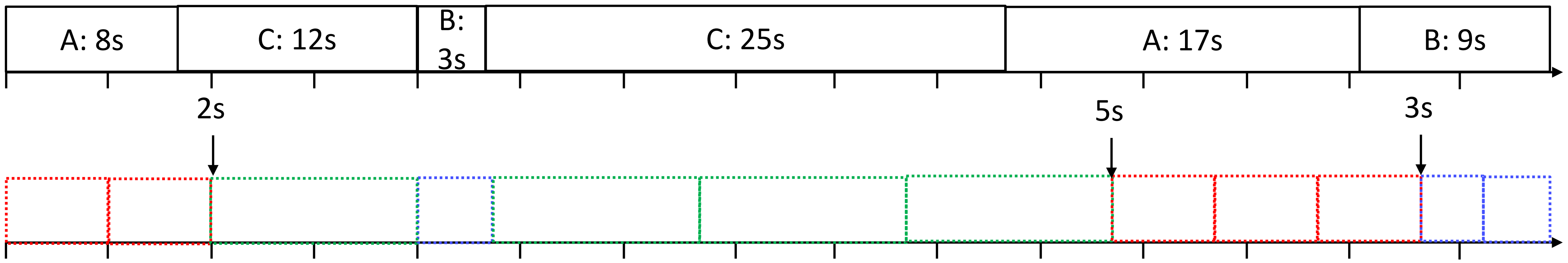
For each event, T_{sp} = minimum interval of that event

T_{sp}A = 5s , T_{sp}B = 3s, T_{sp}C = 10s

#Sensor transmission = 12 times

Total classification latency = 10 sec

Missing CL = 2 times



Example: T_{sp} Based on Alg.1

Example:

Time-intervals history for the following events:

A: [5, 9, 11, 16, 19] seconds
B: [3, 7, 15, 20, 23] seconds
C: [10, 18, 23, 31, 39] seconds

If the following sequence of events occurred and the user set CL = 2 seconds

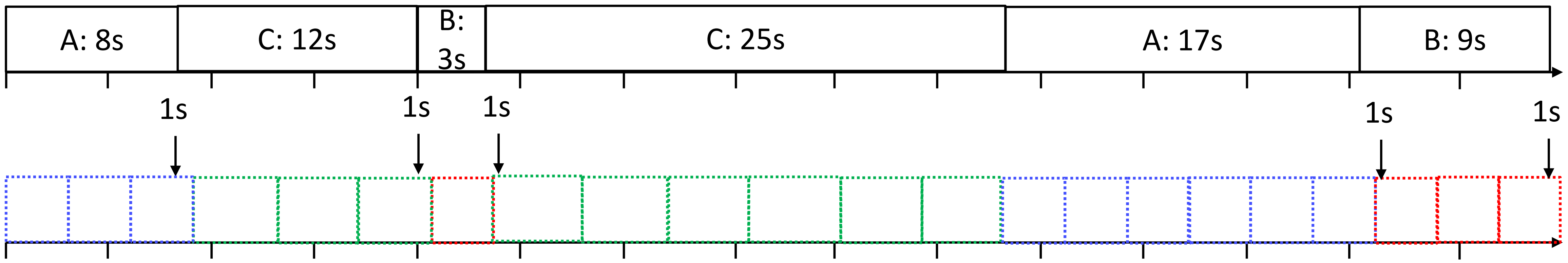
Based on the proposed algorithm

T_{sp}A = 3s , T_{sp}B = 3s, T_{sp}C = 4s

#Sensor transmission = 22 times

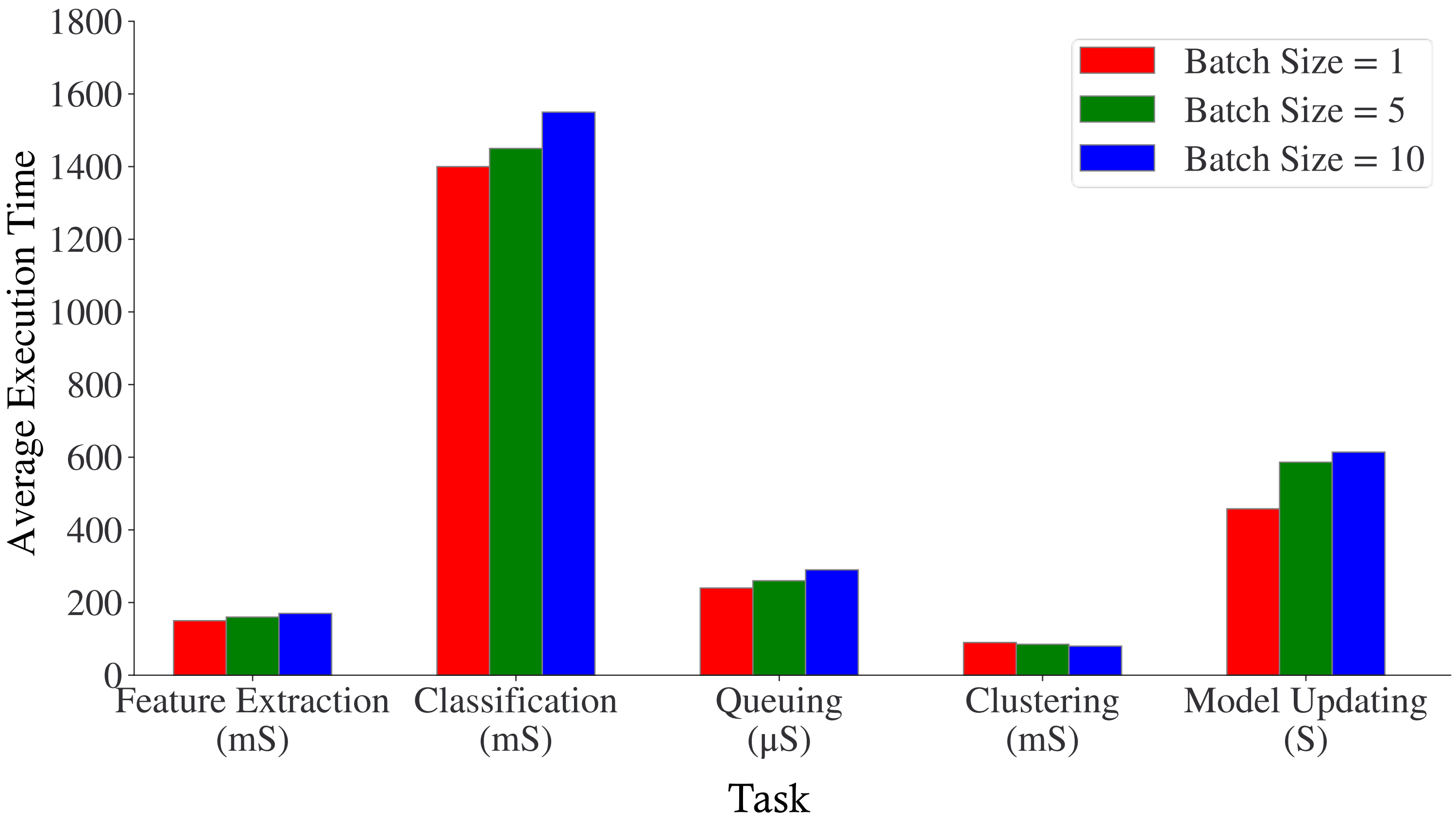
Total classification latency = 5 sec

Missing CL = 0 times



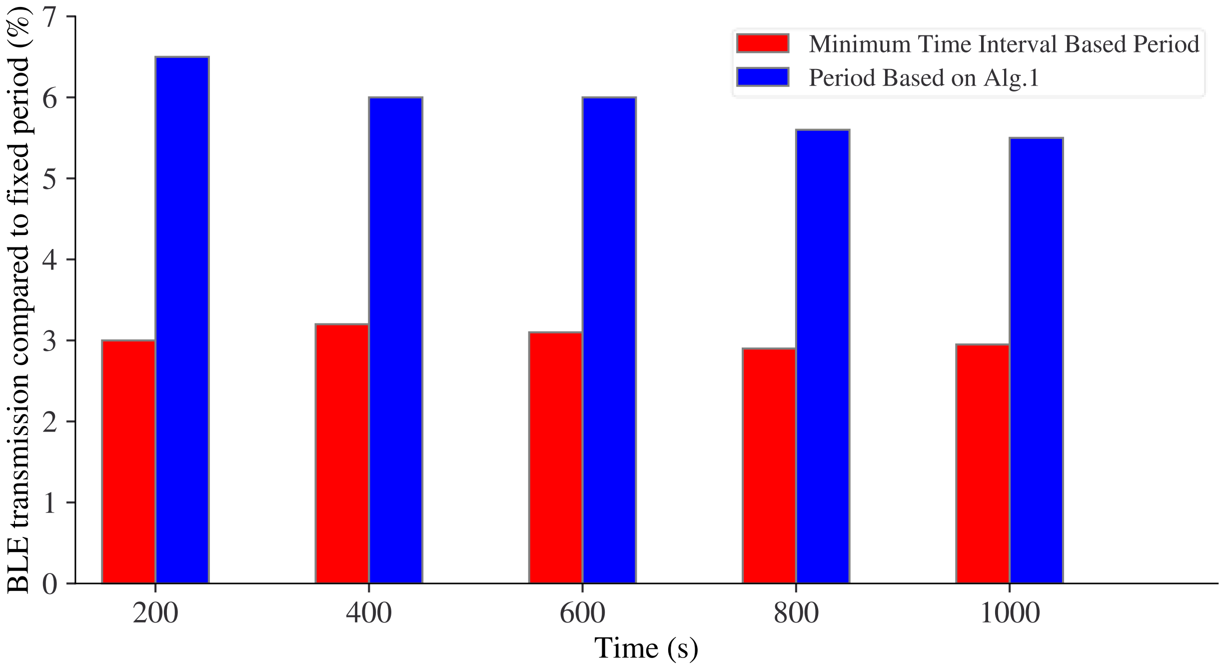
Execution Time of OpenSense on an Embedded Device

Objective: Compute the average execution time of different tasks using different batch sizes



Sensor Dynamic Scheduler Energy Consumption Performance

- The number of BLE transmissions is compared to a fixed period of 1 seconds
- The transmitted BLE packets using Alg. 1 is approximately 6% of the total number of transmissions made by the fixed period approach
- 3% more of polling requests is an acceptable trade-off



Model Updater Scheduler Performance

- We assume there are 200 samples of an unknown class
- The model updater is triggered when it meets the conditions in Alg.2
- the model updater is triggered 3 times to adapt the 200 samples into the model
- T_{sp} is based on the Minimum Time Interval Based Period