CS60055: Ubiquitous Computing

Context Sensing

Department of Computer Science and Engineering

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Sandip Chakraborty
sandipc@cse.iitkgp.ac.in

What is Context in Ubiquitous Computing?

- Any relevant information about the environment, user, or device that can influence the behavior of an application or system

- **Examples:** Several activities have similar patterns

What is Context in Ubiquitous Computing?

- Any relevant inform
  influence the behav
- **Examples:** Several a

Location information can help differentiating such activities: Kitchen vs Washroom

Some Key Context Information

- **Location**
  - Information about where the user or device is physically located
- **Identity**
  - Details about who is involved, which might include information about the user, their preferences, profiles, or other identifying attributes that can help tailor the system's responses
- **Time**
  - The time at which an interaction occurs.
  - Time can include aspects such as the current date and time of day or more abstract notions like a user’s schedule or the duration of an interaction.

Some Key Context Information

- **Activity**
  - Information about what the user is doing or the current state of an environment.
  - This can include activities such as walking, driving, working, or interacting with a particular device.

Examples

- Virtual assistants like Siri or Google Assistant use context such as user location, calendar events, and habits to provide relevant reminders or recommendations.

- Wearable devices can track users' physiological data (like heart rate or movement) and use context such as location (e.g., at home vs. exercising outdoors) to interpret health metrics accurately and provide appropriate feedback.

- In a smart home, the system may use context such as the time of day, user presence, and current activity to automatically adjust lighting, temperature, and security settings.

Context-aware Computing

- Systems and applications are designed to dynamically adapt their behavior based on contextual information
  - being aware of the current context and possibly anticipating future needs based on that context

- **Examples:**
  - In a *smart office*, context-aware applications could detect if a user is in a meeting and automatically set their phone to silent mode
  - In a *smart city scenario*, public transportation systems might adapt schedules based on real-time context such as traffic conditions or weather.

Types of Context

- **Explicit Context:** Information that the user directly provides or that is easily obtained, such as location coordinates from GPS

- **Implicit Context**: Information inferred from user behavior, patterns, or machine learning algorithms, such as detecting a user's activity (e.g., jogging or sitting) based on accelerometer data

Context Sensing Modalities

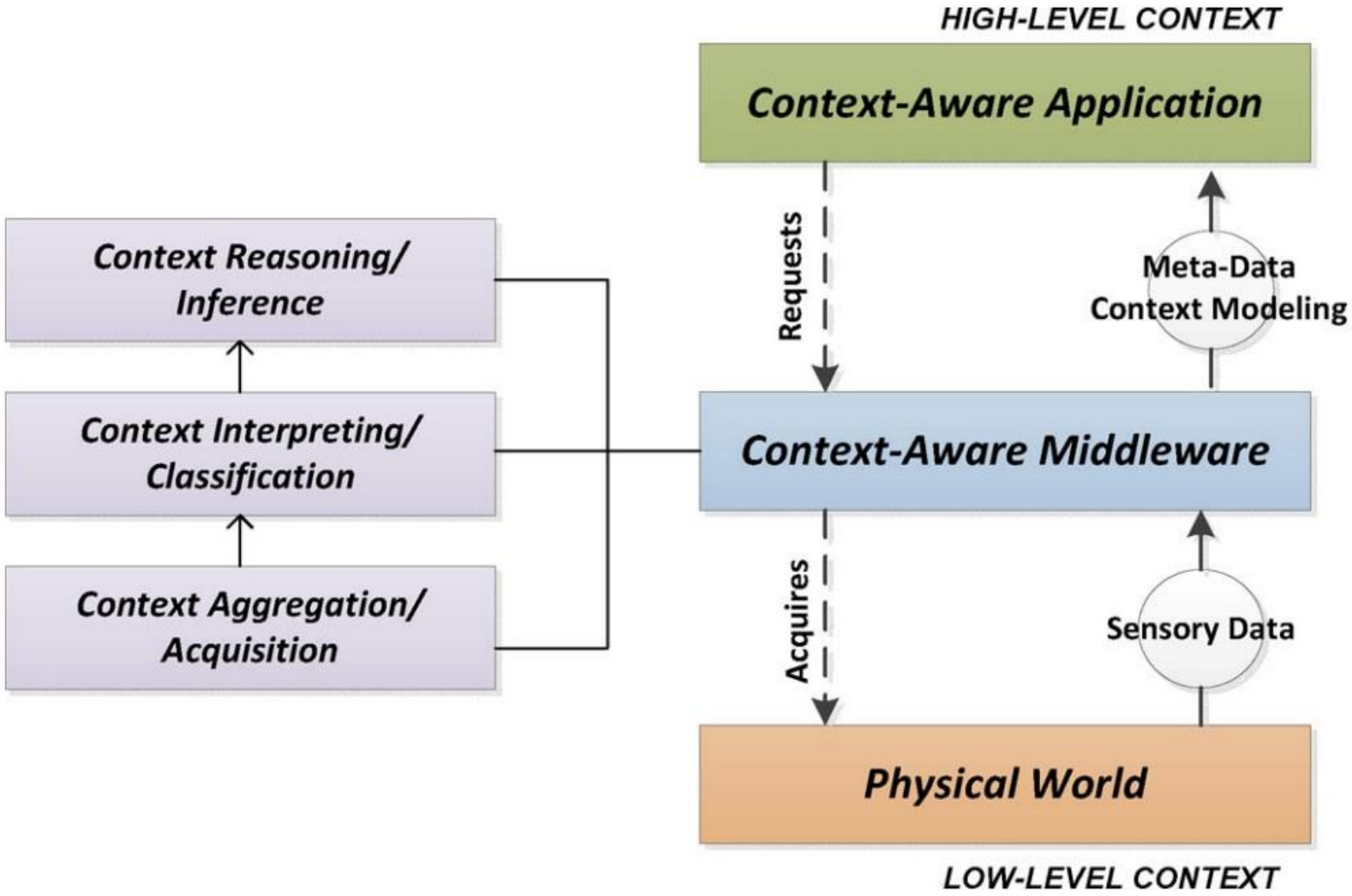
- **Personal/human-centric Sensing**
  - Device user is the point of interest
  - Example: Monitoring and recognition of user related posture and movement patterns for personal fitness log

- **Participatory Sensing**
  - Relies on multiple deployment of mobile devices to interactively and intentionally share, gather and analyze of each local knowledge that is not solely based on a human activity, but also based on surrounding environment
  - Example: Delivering an intelligent traffic congestion report in case where many users provide their speed and location information

Context Sensing Modalities

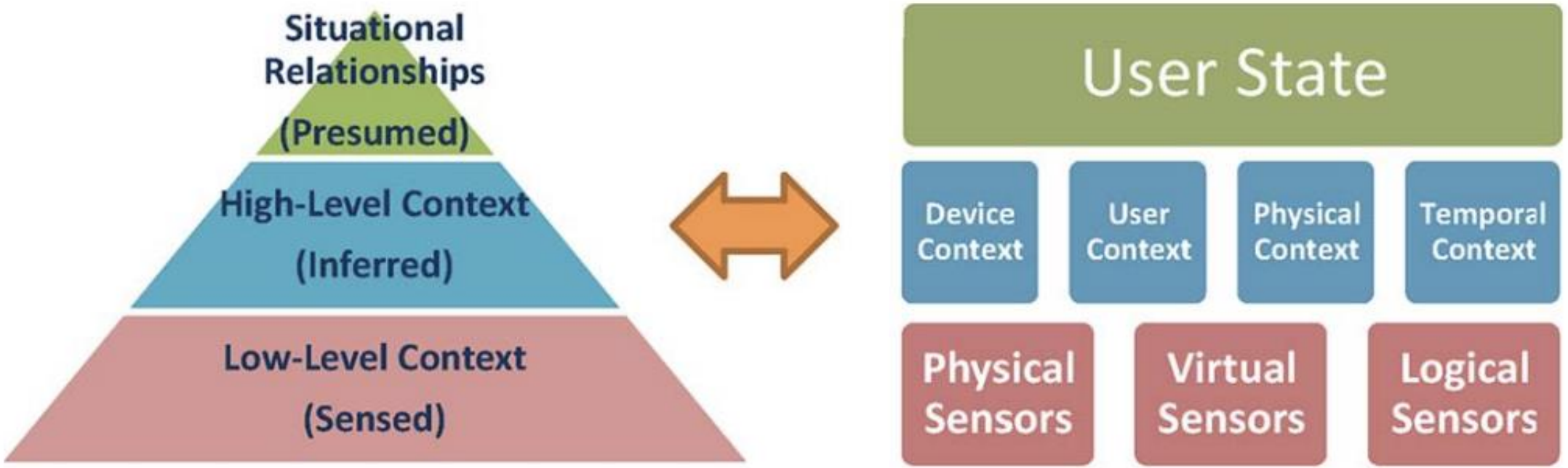
- **Opportunistic Sensing**
  - Collects sensory data in a fully autonomous way without active user interaction (runs as a background service)
  - Example: Ambient sound recognition (Siri)

General Architecture of a Context-Aware System



Source: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6985718

Hierarchical Representation of Context

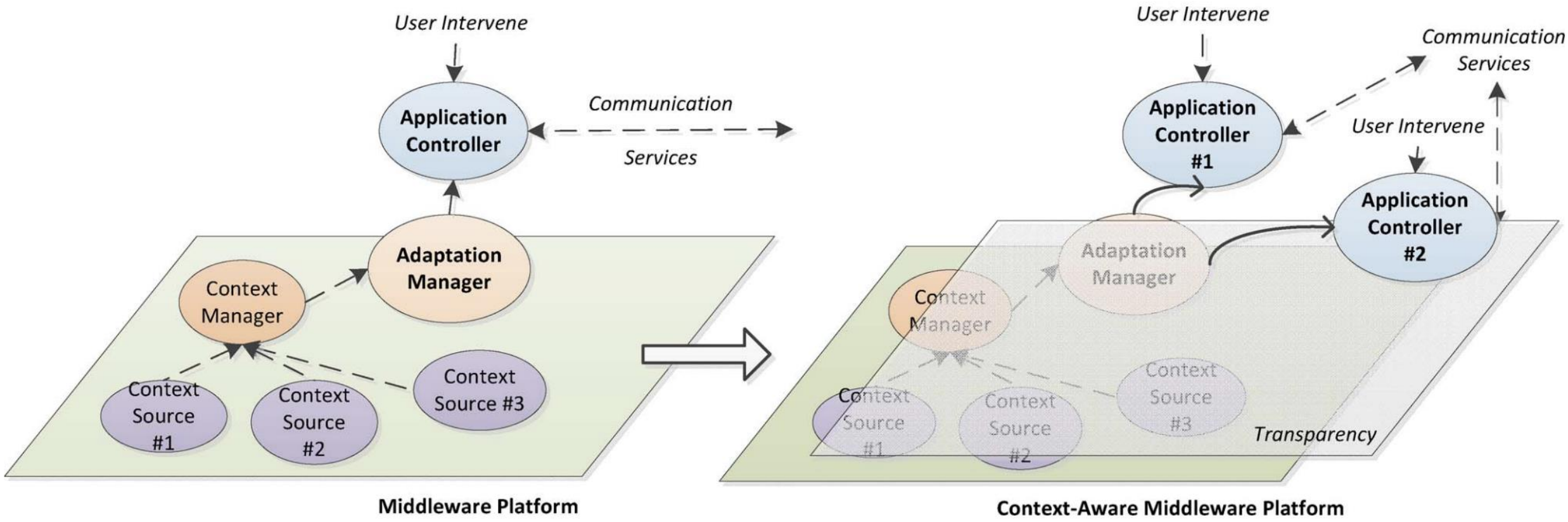


Source: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6985718

Context-aware Middleware

- A *layered* system architecture that can response effectively for optimal sensor utilization, large sensory data acquisitions as well as meeting application requirements
  - leverages the pervasive context-processing software libraries
  - Considers mobile device resource constraints while handling context analysis and context feedback to the applications
- Creates a shielded interface
  - Enhances the level of abstraction support needed by the application
  - Hides lower layer operations between the physical layer (i.e., hardware and communications) at the bottom and the application layer at the top

A Typical Context-Aware Middleware



Source: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6985718

How Context-Awareness Help in Smart and Safe Driving

2023 IEEE International Conference on Pervasive Computing and Communications (PerCom)

DriCon: On-device Just-in-Time Context Characterization for Unexpected Driving Events

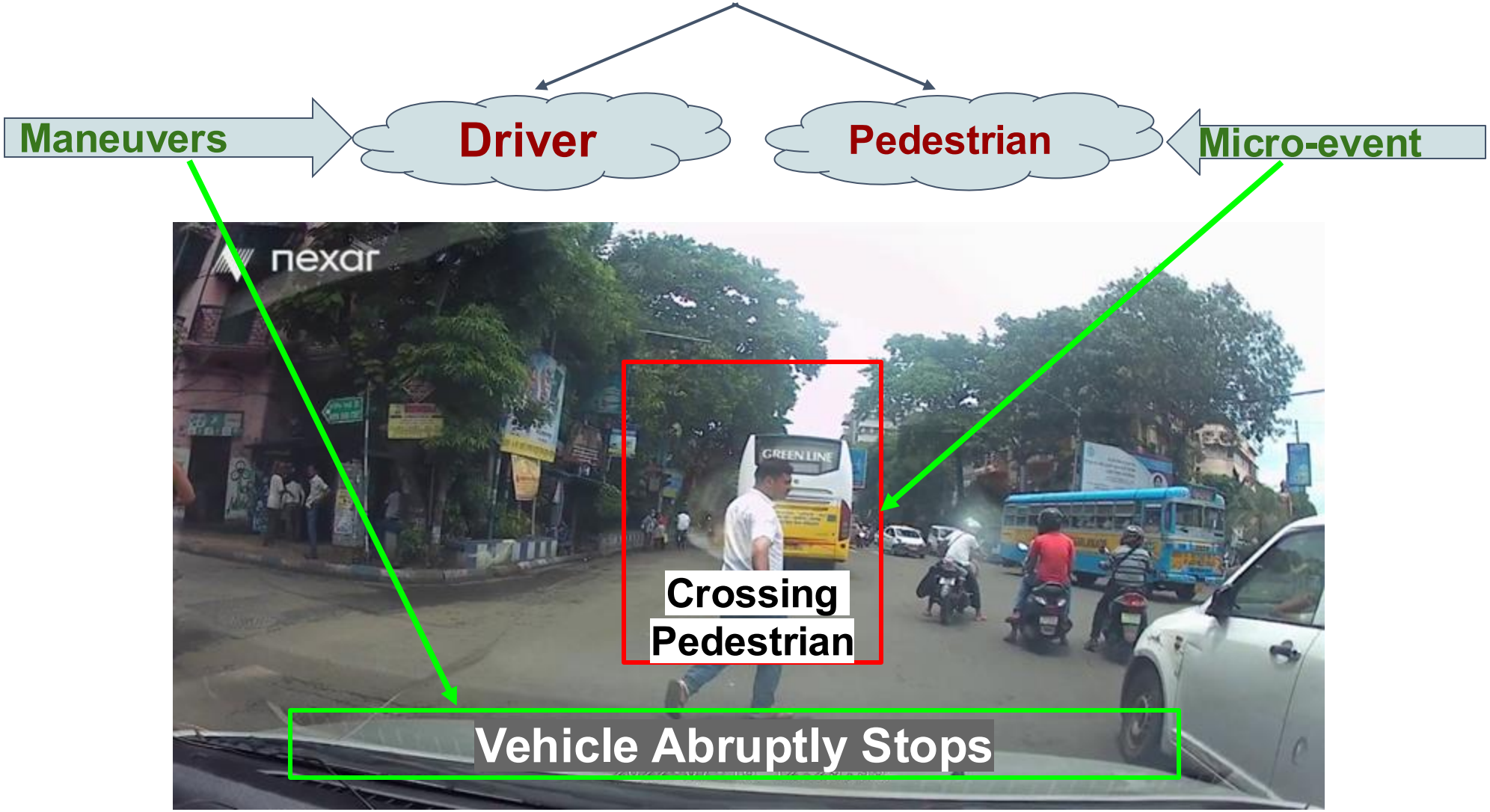
Debasree Das, Sandip Chakraborty, Bivas Mitra
Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, INDIA 721302
Email: {debasreedas1994, sandipchkraborty, bivasmitra}@gmail.com

Can My Smartphone Mark My Uber Driver as Good/Bad?

Can My Smartphone Mark My Uber Driver as Good/Bad?

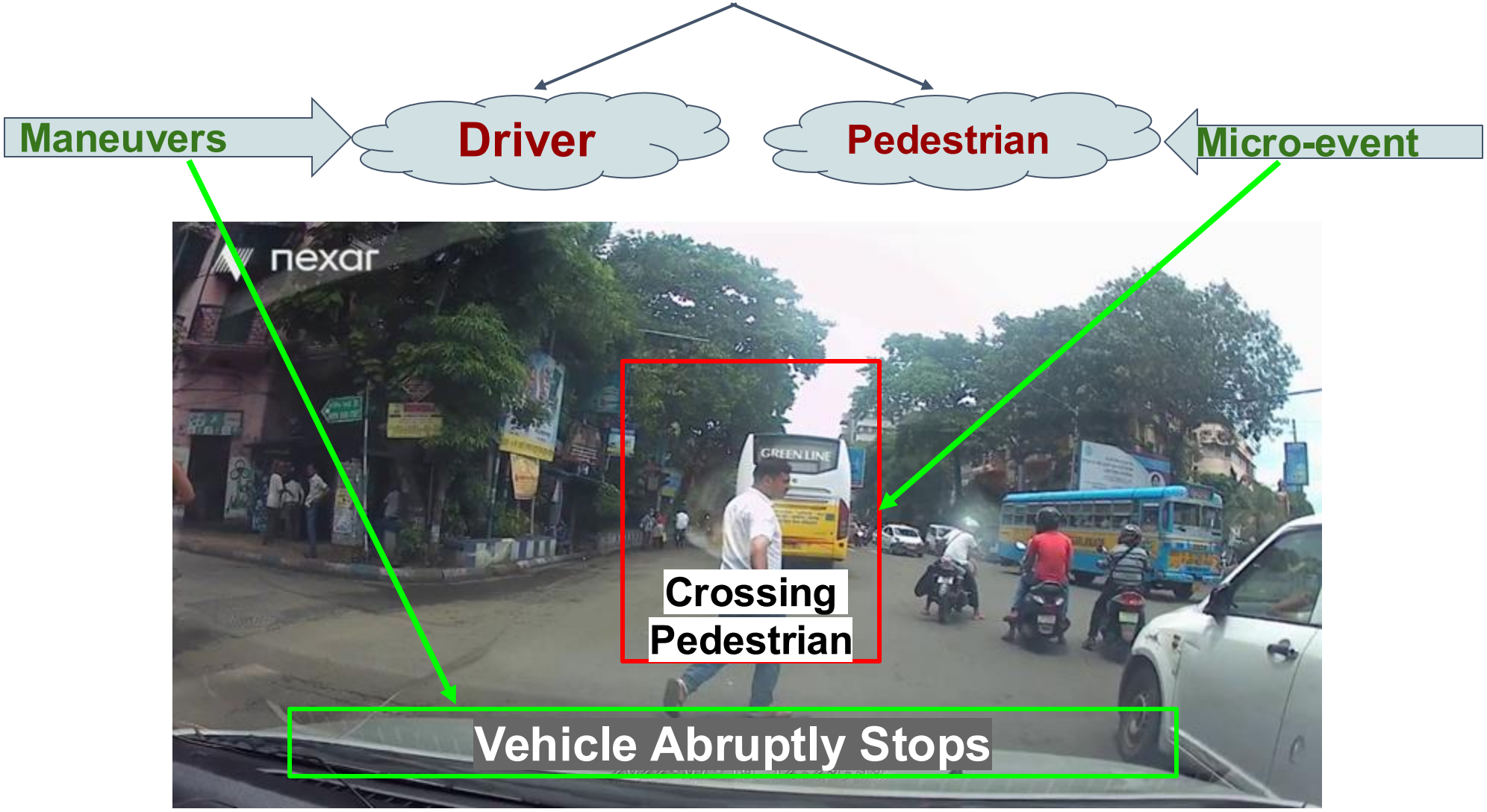
Can My Smartphone Mark My Uber Driver as Good/Bad?

- Who is at the fault?



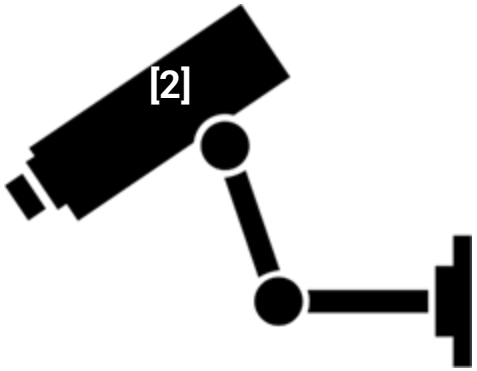
Can My Smartphone Mark My Uber Driver as Good/Bad?

➢ How Can we Infer Such Contexts?



Existing Modalities to Solve This Problem

[1] SigSpatial 2017: Understand Reasoning behind Slowdown, Congestion.



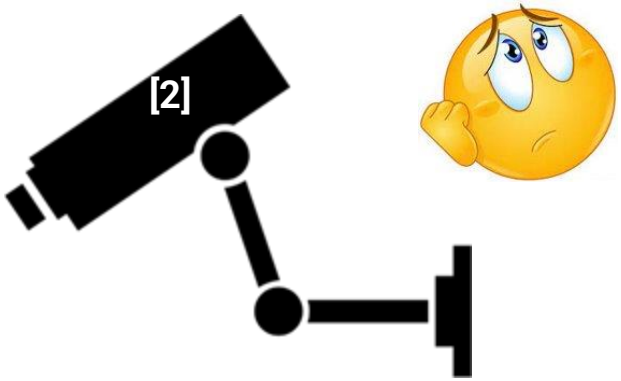
[2] TUAT 2017: Analyzes Drive Recorders from Taxis

[3] AutomotiveUI 2019: Tracks Cooperative Overtaking for Connected Vehicles

[4] CVPR 2018: Causal Reasoning behind Driving Behavior

Existing Modalities to Solve This Problem

[1] No surrounding traffic information

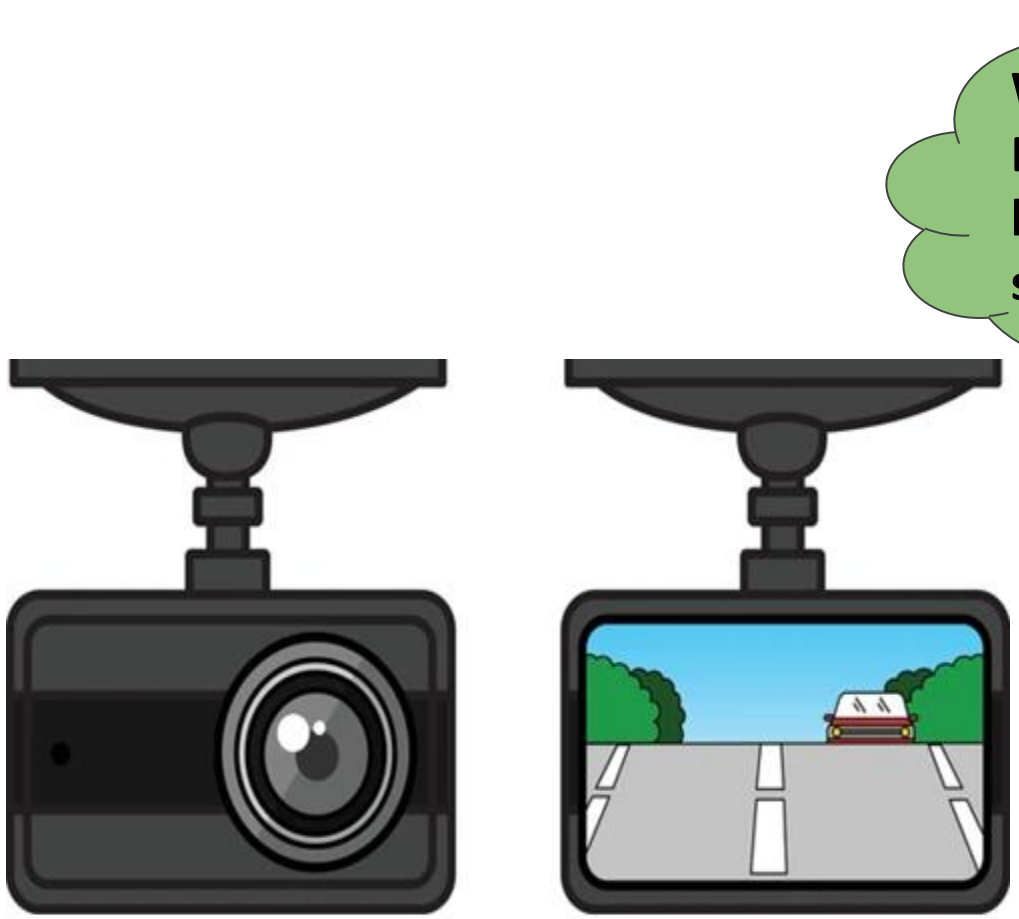


[2] Manual and offline analysis

[3] Needs connected vehicles

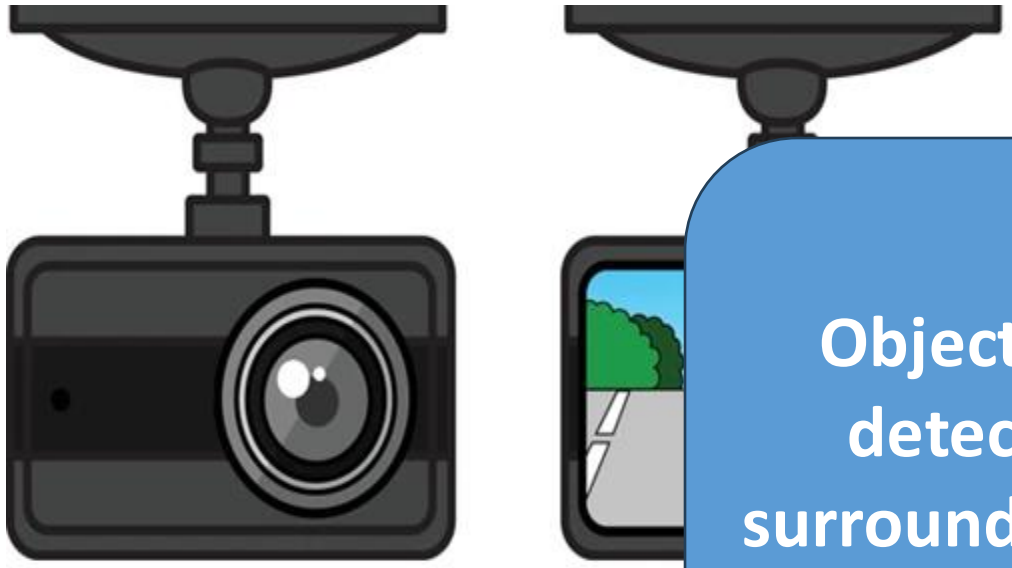
[4] Human annotation and limited resoning

How Can We Sense the Environment?



Can we use a smart dash-cam to understand the context behind poor driving behavior?

How Can We Sense the Environment?



Wait!! A smart DashCam can sense both in-vehicle and surrounding!!!

Objective: Develop a Proof-of-Concept model to detect the context in terms of interaction with surrounding traffic behind degraded driving behavior and generate human explainable output on the fly

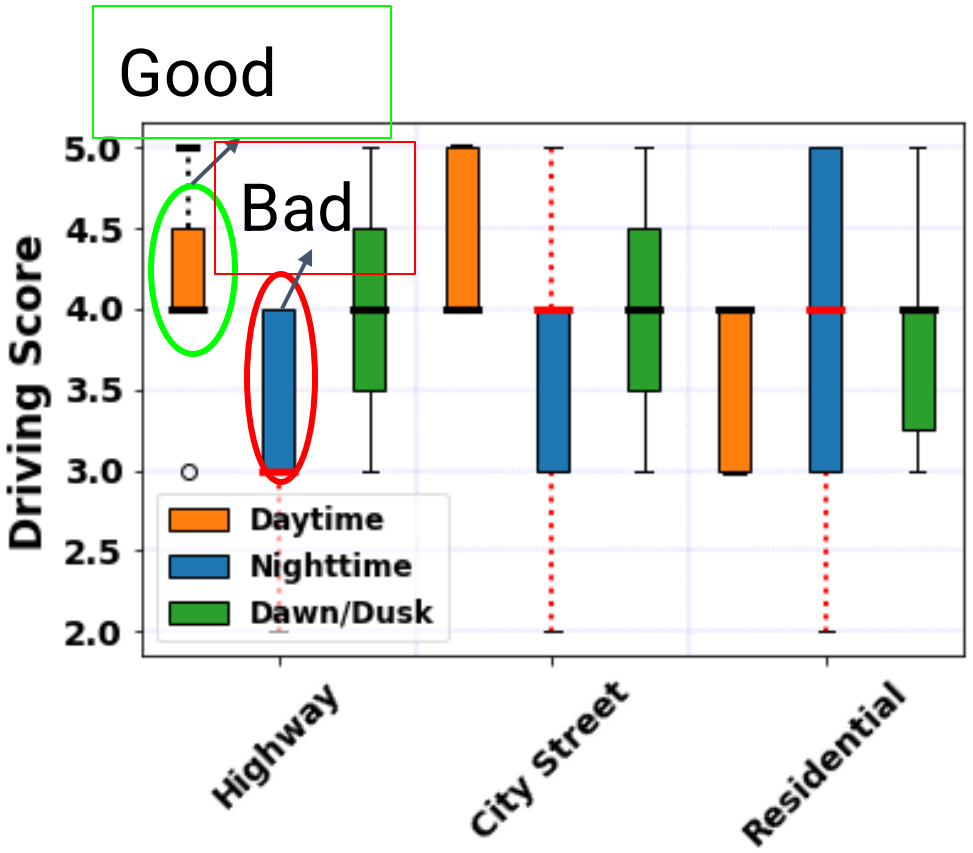
Dataset Used

Public Dataset		
Accelerometer	Gyroscope	GPS
Road type	Time of the day	Weather
Video data captured through rear camera		Driving Score[2]

- *We use Berkeley DeepDrive (BDD)[1] dataset covering **USA** and **Israel** and comprises 100k trips*

[1] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In IEEE CVPR, 2020.
[2] D. Das, S. Pargal, S. Chakraborty, and B. Mitra. Dribe: on-road mobile telemetry for locality-neutral driving behavior annotation. In 23rd IEEE MDM, 2022.

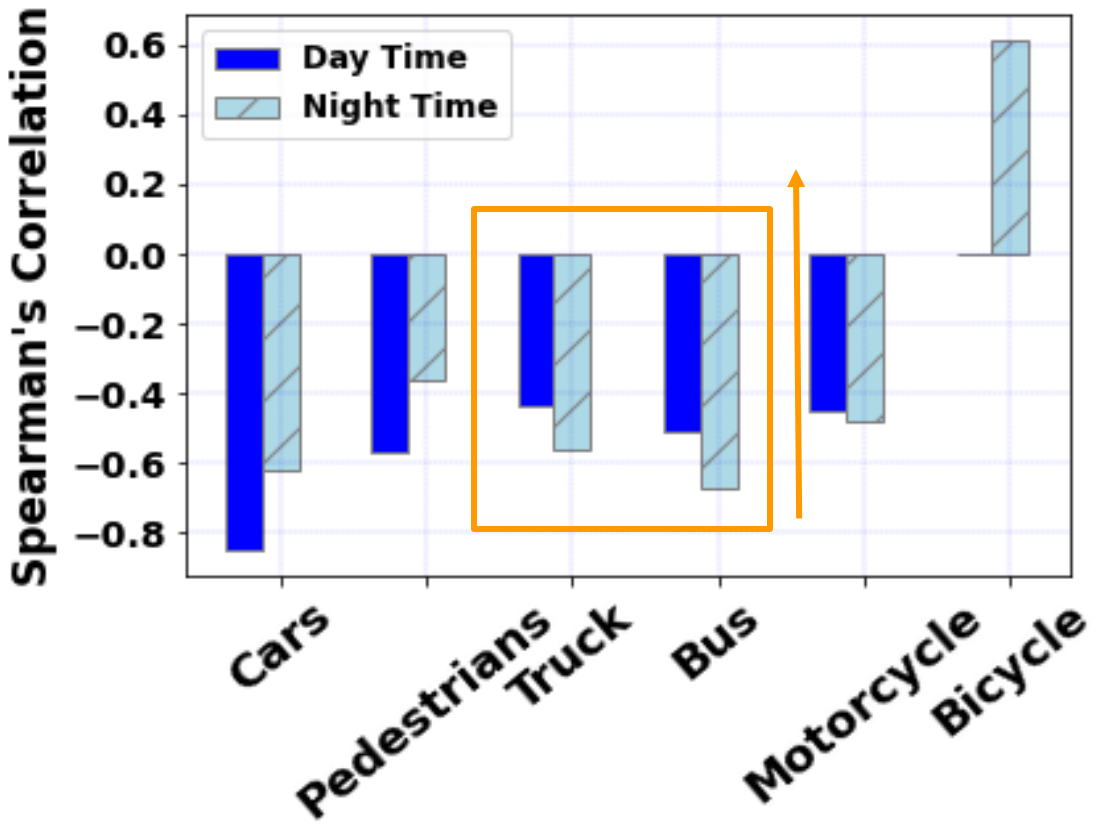
Pilot Study



- Driving behavior variation over space and time

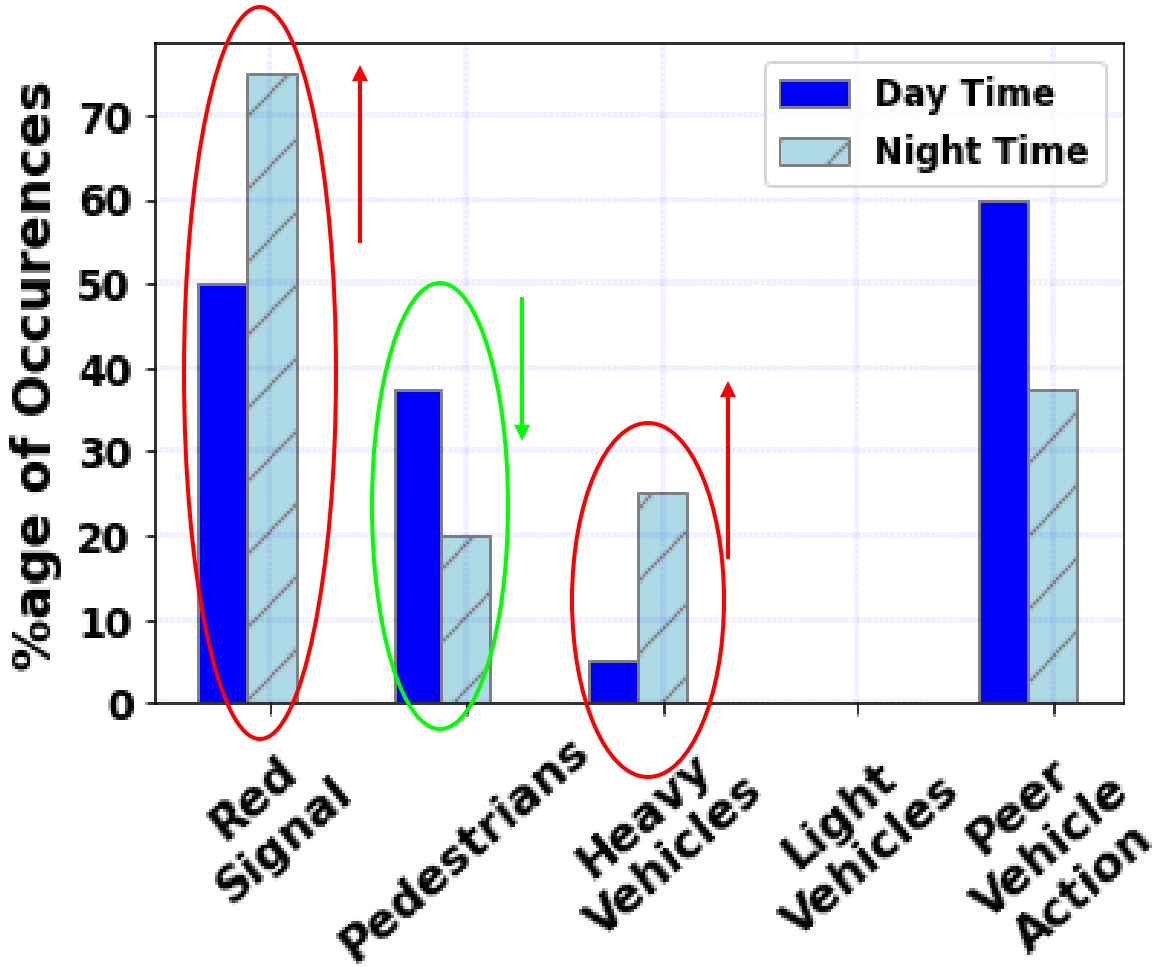
- Two parameters are varied:
  - Road Type (“Highway”, “City Street”, “Residential”)
  - Time of the Day (“Daytime”, “Nighttime”, “Dawn/Dusk”)

Pilot Study



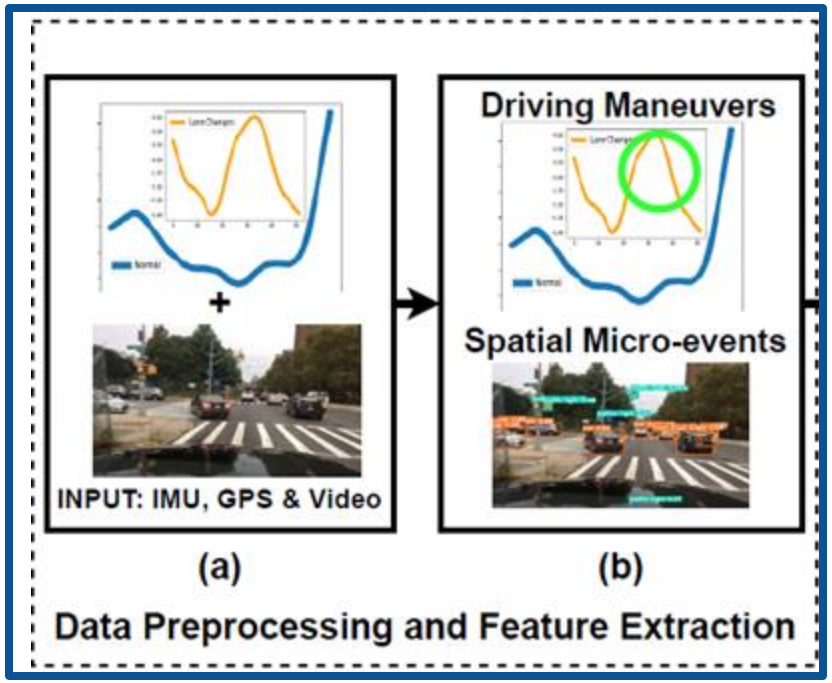
- Impact of **Spatial Micro-events** on driving behavior
- Spearman's Correlation is computed to see the significance
- Observation differs for time of the day

Pilot Study



- What are the Spatial Micro-events responsible behind a driving maneuver?
- Use Case: Abrupt Stop
- Not all of them are equally responsible

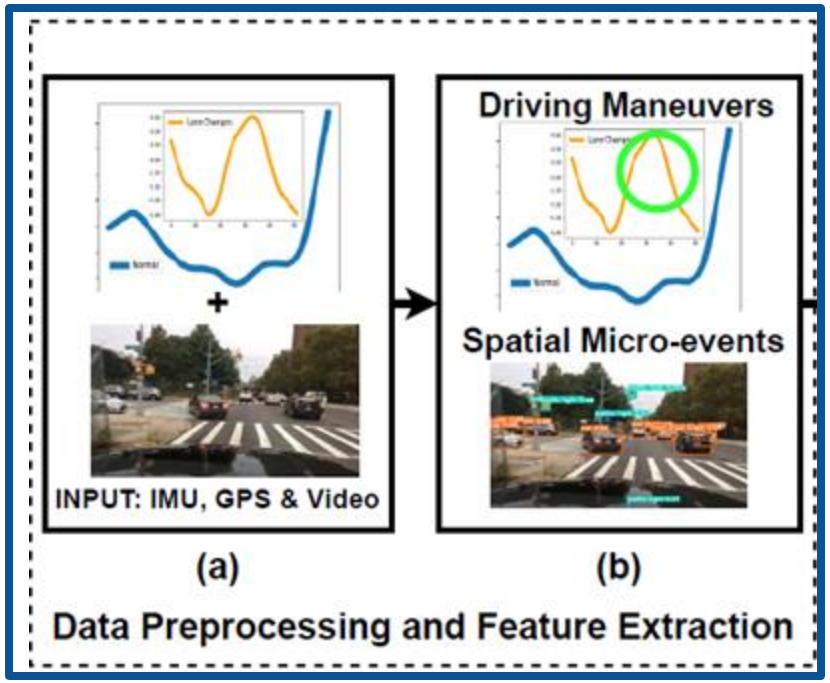
DriCon Overview



We extract a set of well-known driving maneuvers

Driving Maneuvers
➢ Weave
➢ Swerve
➢ Sideslip
➢ Abrupt Stop
➢ Sharp Turn
➢ Severe Jerkiness

DriCon Overview

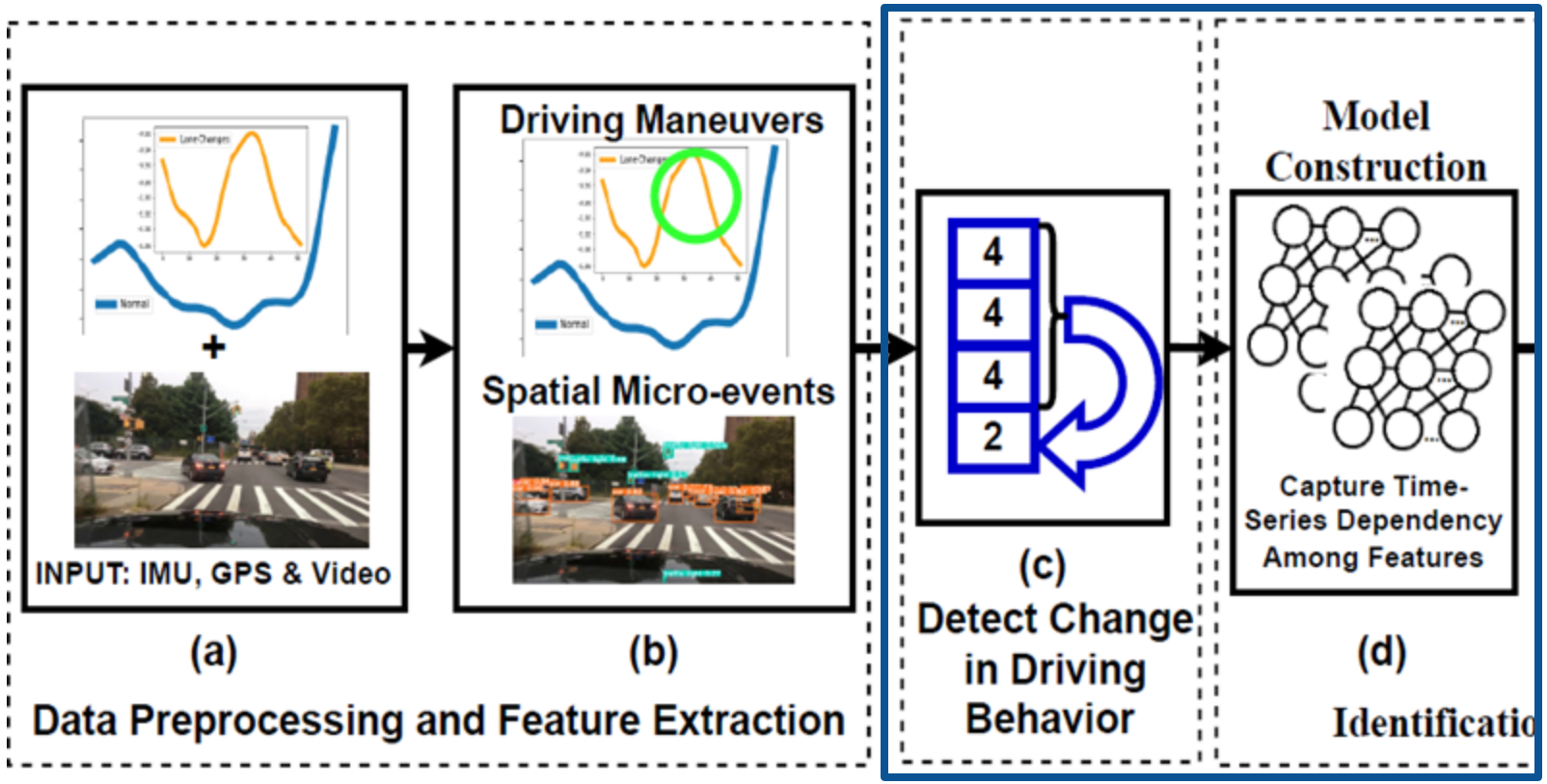


Spatial Micro-events are classified among Action of *Preceding Vehicle*, *Peer Vehicles*, *Pedestrians*, *Static Road Attributes* & *Weather Condition*

Spatial Micro-events

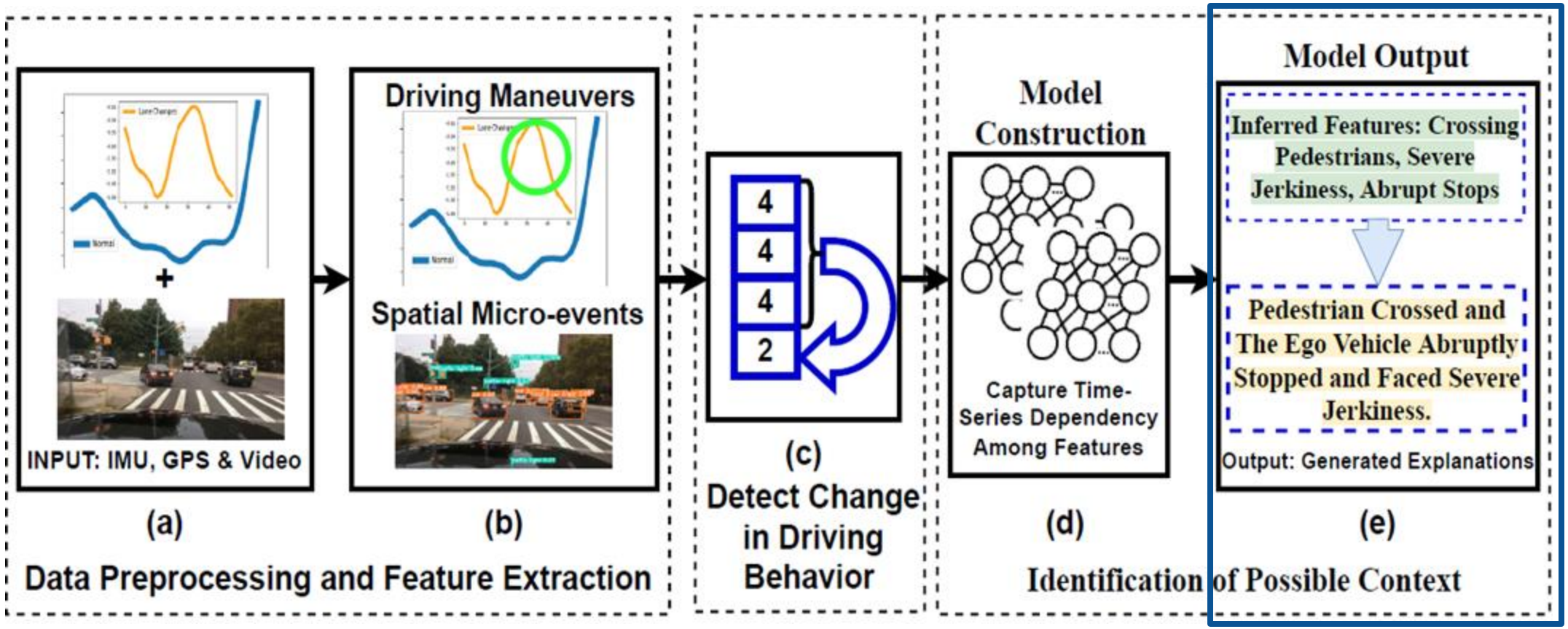
- Relative Speed
- Relative Distance
- Prec. Vehicle Brake
- Congestion
- Pedestrian
- Pedestrian's Speed
- Traffic Light
- Type of Road
- Heavy Vehicle
- Weather Condition

DriCon Overview



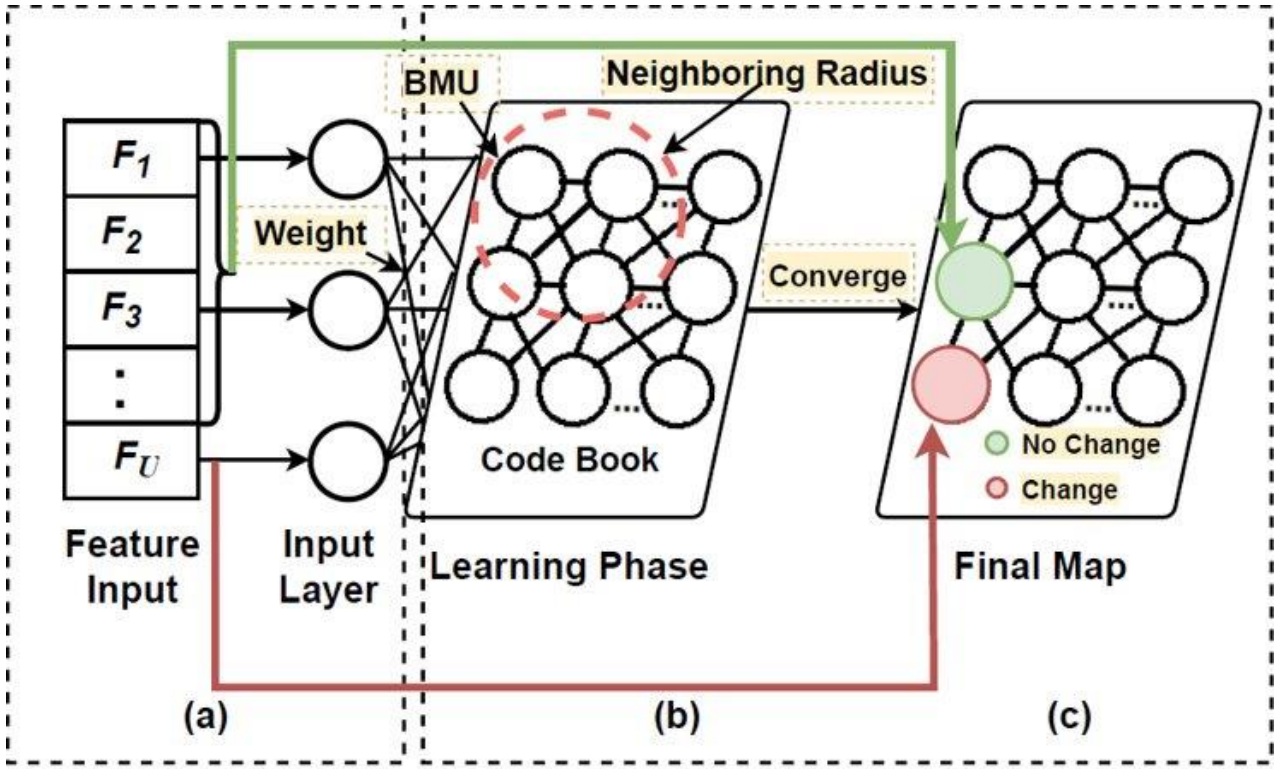
- Opportunistically detect the change in driving behavior
- Design a model which can identify the responsible features

DriCon Overview



Generate human-interpretable explanation

Methodology

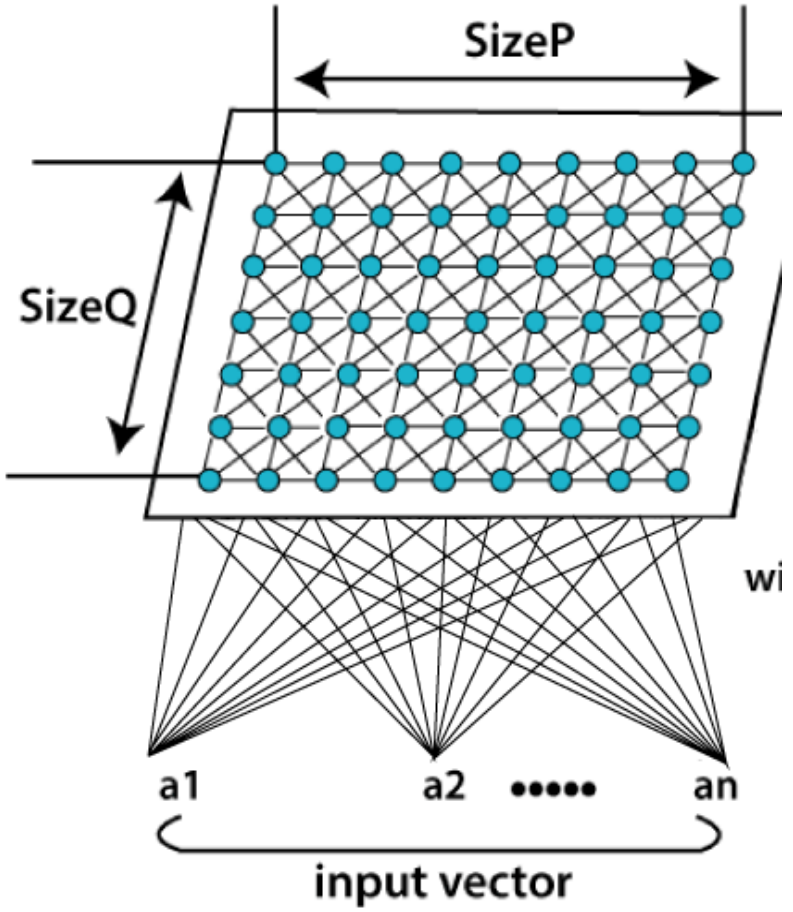


- Lightweight ANN model is chosen
- ***Self Organizing Maps*** is utilized
- Unsupervised technique applied to learn feature dependency

What is a Self-Organizing Map (SOM)

- An unsupervised neural network algorithm that is used for data visualization, clustering, and dimensionality reduction
  - Also known as a **Kohonen Map** or **Kohonen Network**
  - It organizes data into a two-dimensional grid, preserving the topological structure of the input data as much as possible

- SOMs maintain the spatial relationships in the data.
  - Points that are close in the input space are mapped to neighboring neurons on the 2D grid, preserving the similarity in a low-dimensional representation



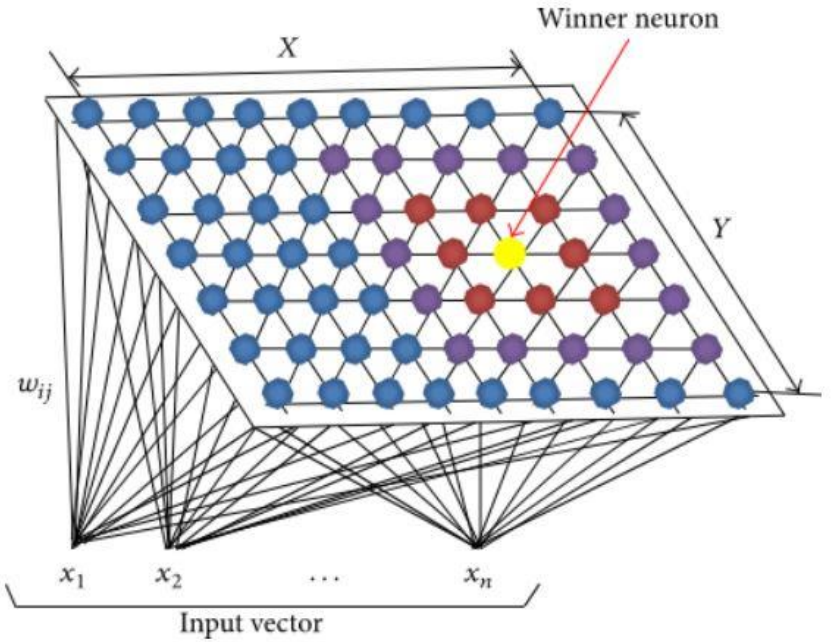
Kohonen Architecture

How A SOM Works

- **Initialization**
  - Each neuron in the SOM grid is initialized with a random weight vector, with the same dimension as the input data
- **Input Data Presentation**
  - For each input vector in the dataset, the algorithm finds the Best Matching Unit (BMU), which is the neuron whose weight vector is closest to the input vector (usually by Euclidean distance).

How A SOM Works

- **Initialization**
  - Each neuron in the SOM grid is initialized with a rand dimension as the input data
- **Input Data Presentation**
  - For each input vector in the dataset, the algorithm fi (BMU), which is the neuron whose weight vector is cl by Euclidean distance).



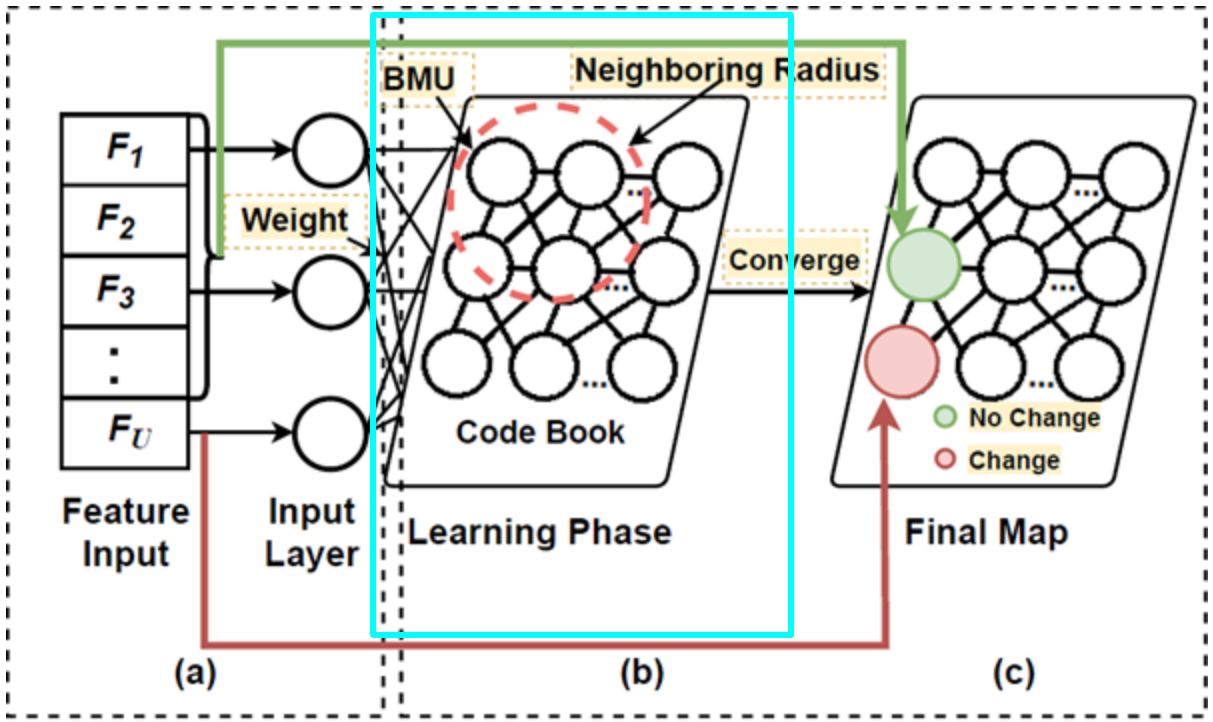
- **Updating Wights**
  - The weight vector of the BMU and its neighboring neurons are updated to become more similar to the input vector. The amount by which the weights are updated is determined by a learning rate and a neighborhood function.
  - The neighborhood function determines the radius around the BMU within which neurons are affected, and it usually decreases over time, so that the map gradually refines itself.

How A SOM Works

- **Iteration**
  - The process is repeated for multiple iterations, with the learning rate and neighborhood size decreasing gradually, allowing the map to stabilize.

- **Convergence**
  - After sufficient iterations, the SOM reaches convergence, meaning the neurons' weight vectors have adapted to the input data's distribution.
  - Similar data points are mapped to nearby neurons, creating a structured, low-dimensional representation of the data.

SOM in DriCon

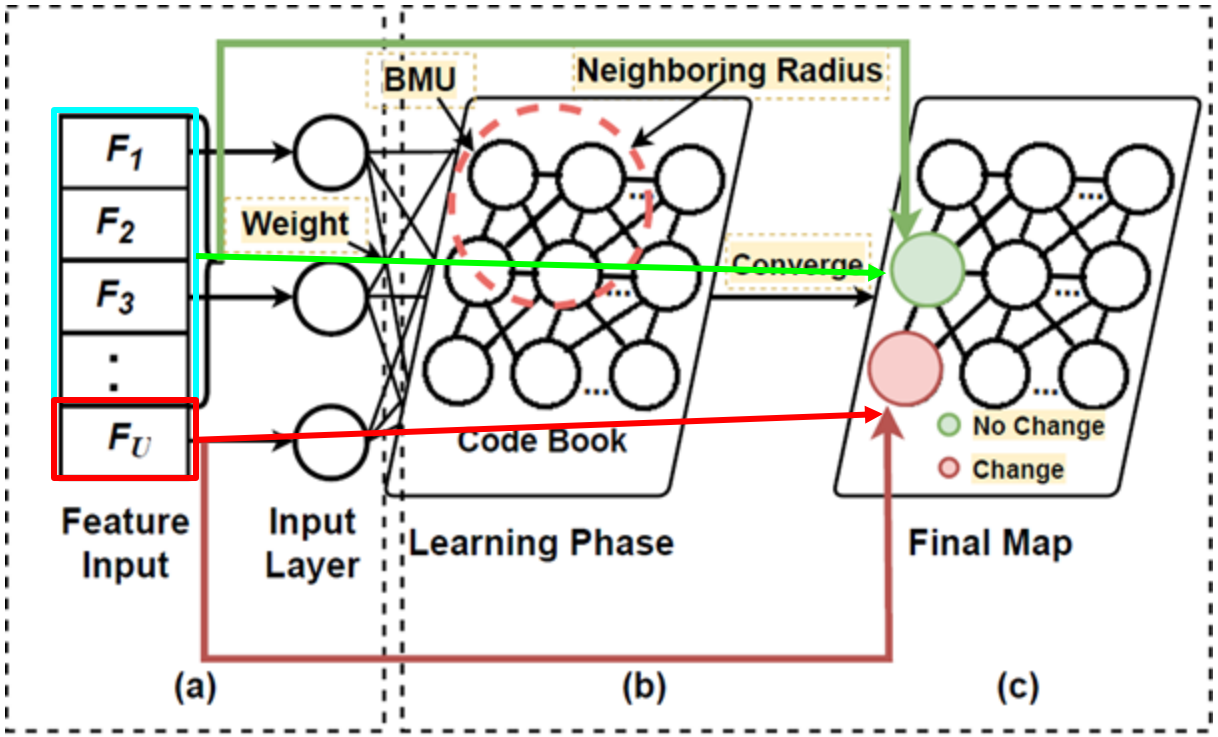


- A codebook (2D Grid) with 147 neurons are implemented
- Neurons are spread over 7 x 21 2-D array

Past Window Length

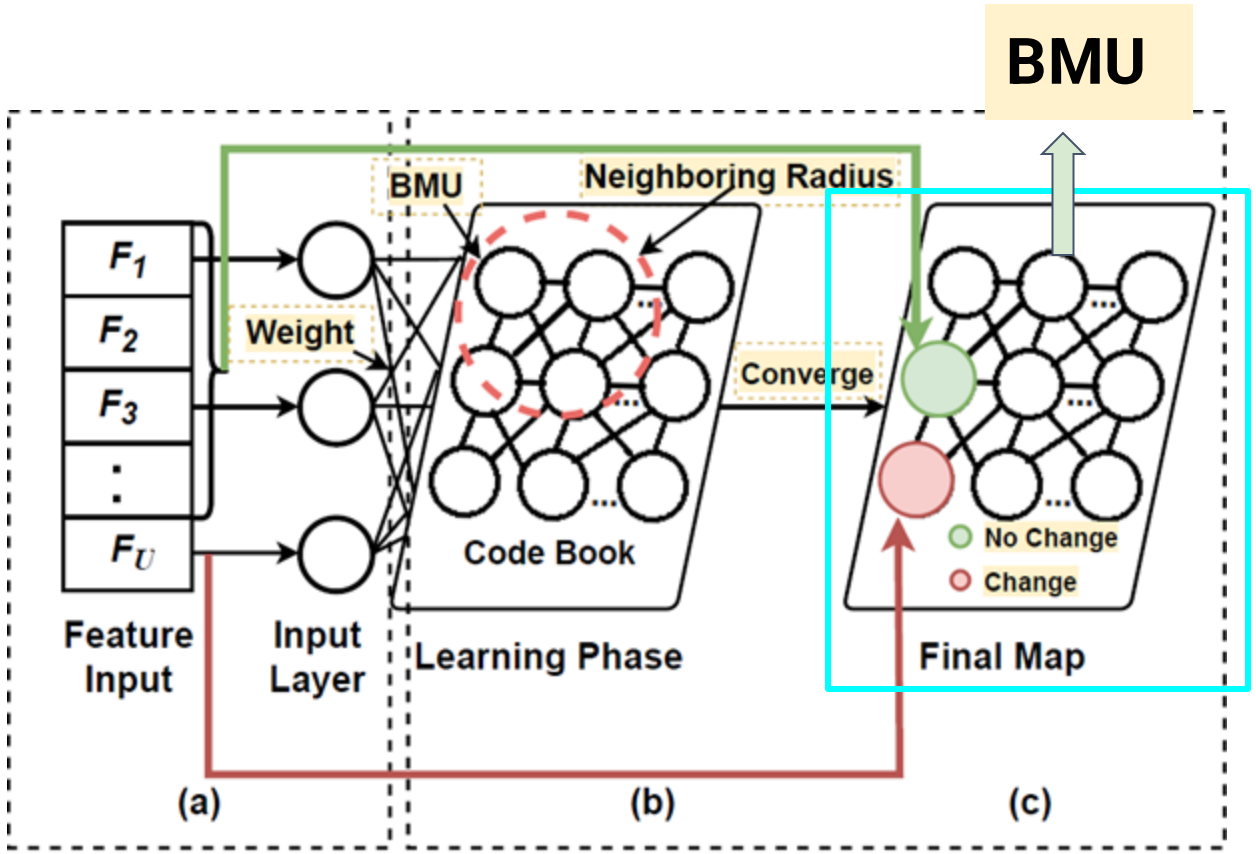
Feature Dimension

SOM in DriCon



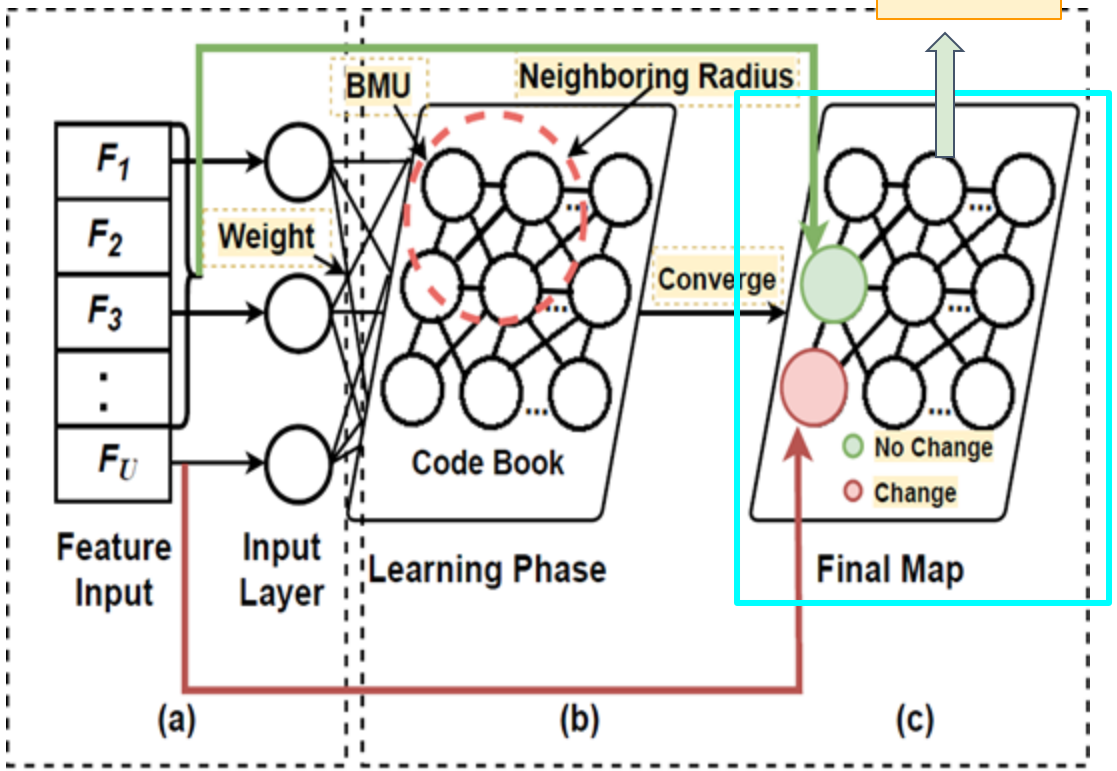
- Each window Fi ϵ (f1 U f2 U … U fn)
- fi is either driving maneuver or spatial micro-event
- Feature vector at windows [1… (U-1)] gets mapped to one unit
- Feature vector at window U gets mapped to a different unit

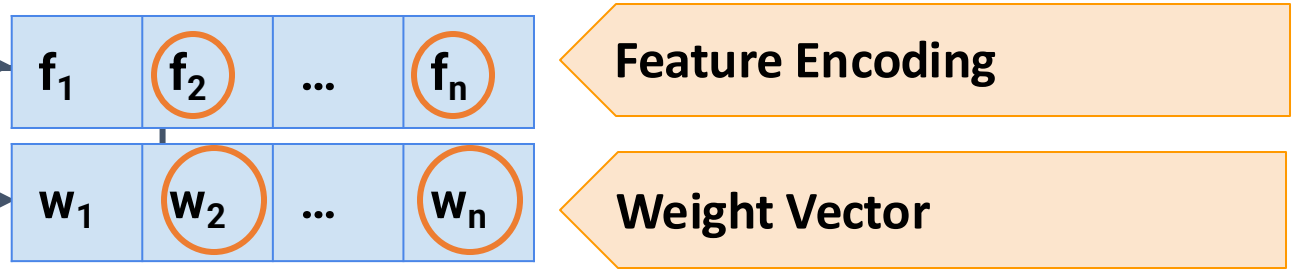
SOM in DriCon



- Finally, we get the codebook upon convergence
- For each trip instance, we get one neuron which is the Best Matching Unit (BMU) for our SOM structure

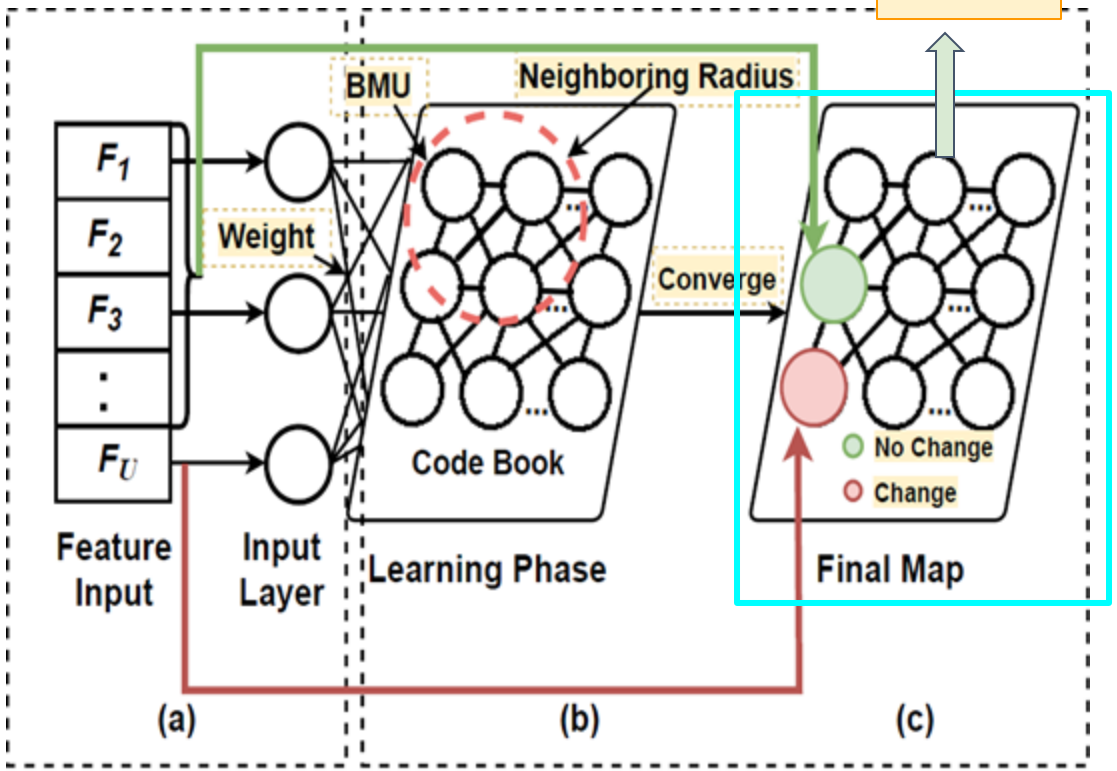
SOM in DriCon

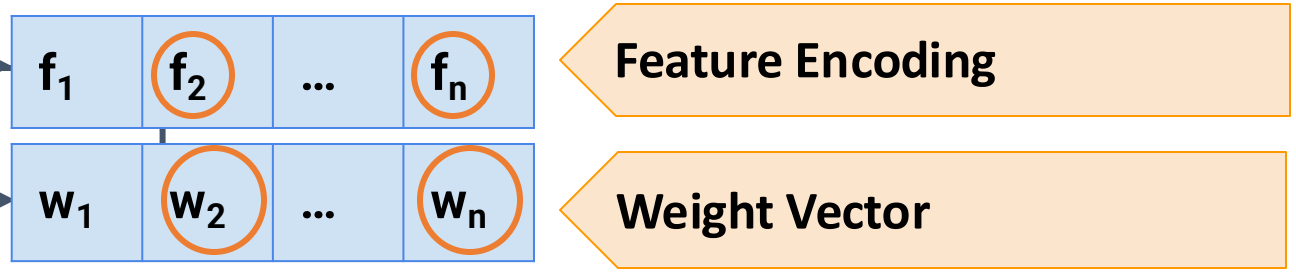




- From the BMU, we get the weight vector and feature encoding
- Extract top-k features with higher weights; keep in F_{GEN}

SOM in DriCon

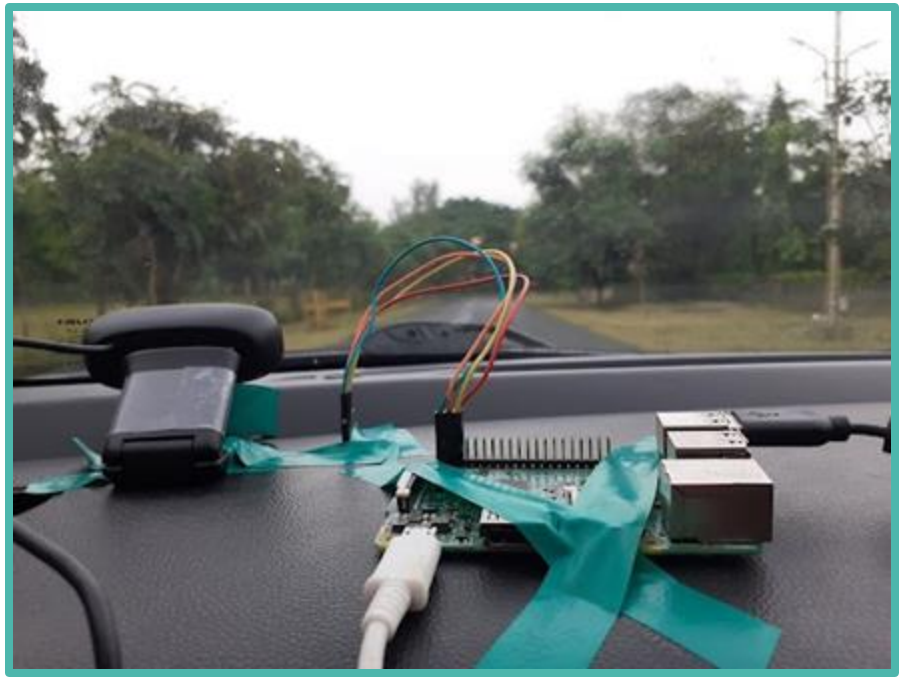




- Determine if feature is an action or subject
- Depending upon maneuver or micro-event
- For action, use “ego vehicle” as the subject and describing action
- For subject, use TF-IDF vectorizer for curating similar guidelines
- Sample Output: “jerks”→“the ego vehicle severely jerks”

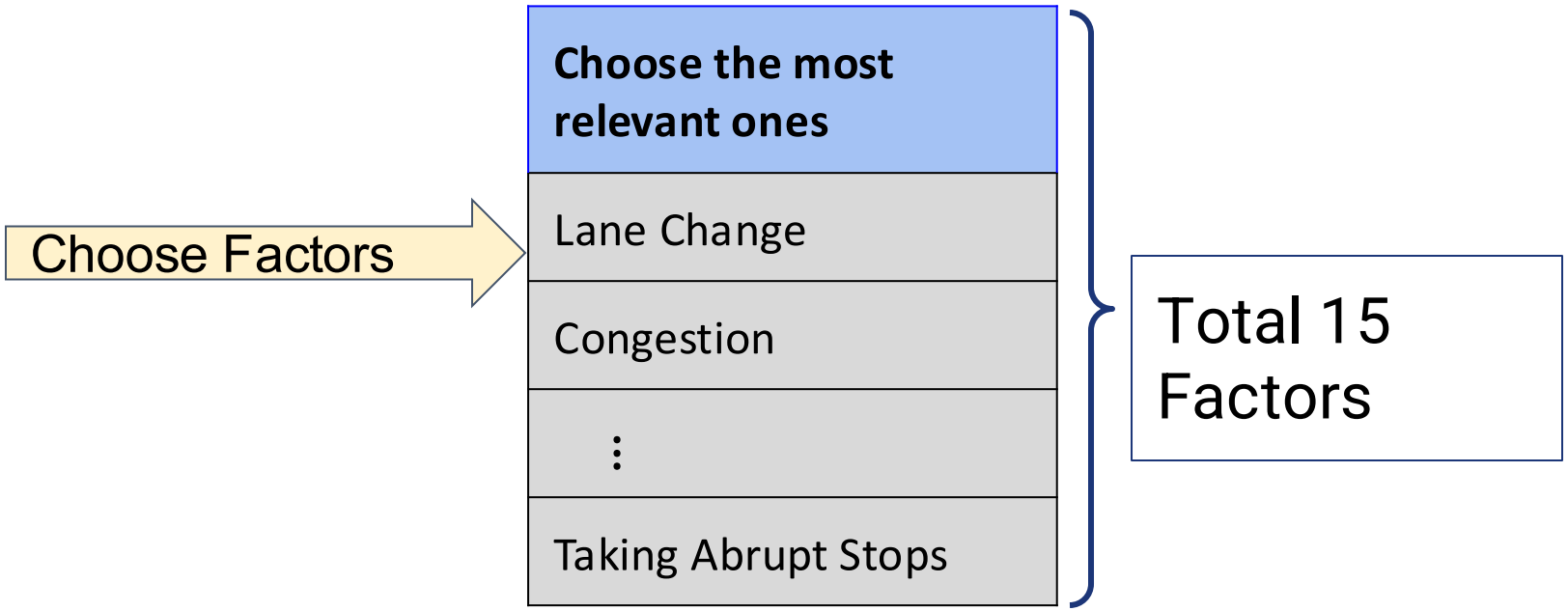
Evaluation Setup

Edge Device	Raspberry PI 3 Model B		
OS	Linux kernel version 5.15.65 – v7+		
Primary Memory	1 GB	**Processor**	ARMv7
Sensors	MPU–9250 IMU, u-blox NEO–6M GPS & Logitech USB camera	**Driving Data**	33 Hours
Type of Vehicles	SUV, Sedan & Hatchback	**#Cities**	3
#Drivers	6 (5 Male & 1 Female)	**#Type of Road**	3



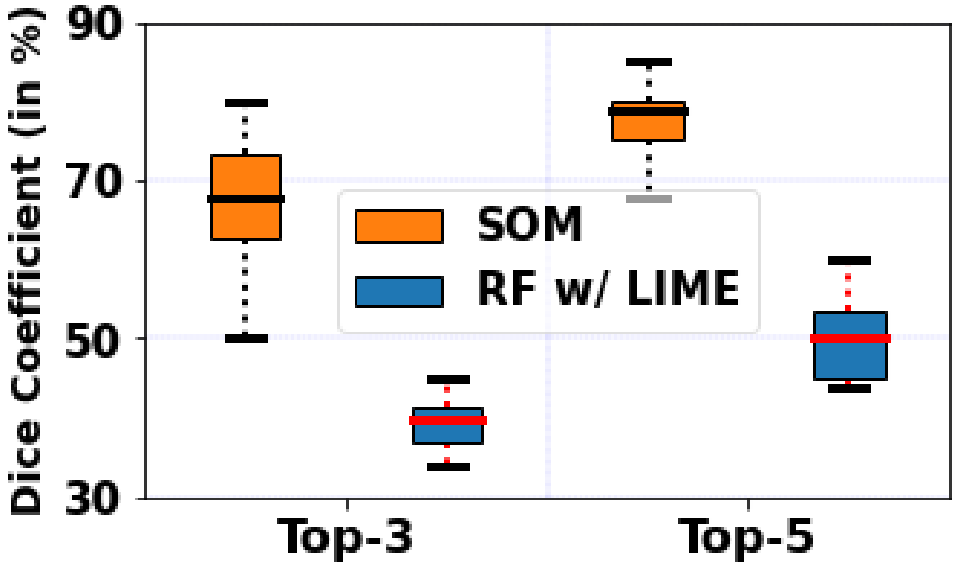
Evaluation Setup

- Each of the video clip is annotated by at least 3 annotators
- They have chosen the most relevant factors
- Validated factors (using majority voting) are kept at F_{GT}
- Dice Coefficient is used between F_{GT} and F_{GEN}



Evaluation Results

- Human Annotated (F_{GT}) vs DriCon Generated (F_{GEN}) Feature similarity
- Supervised (RF with LIME) vs Unsupervised (SOM)
- Achieved a maximum percentage similarity of 79% using SOM



Instance#	Human Annotated (F_{GT})	Model Generated (F_{GEN})	Similarity (in %)	ATE[1]
1	Sideslip, Taking Abrupt Stop, Traffic Lights: Red	Traffic Lights: Red, Congestion, Abrupt Stop	66.67%	2.5

Inferring Contexts

TAO: Context Detection from Daily Activity Patterns Using Temporal Analysis and Ontology

SUDERSHAN BOOVARAGHAVAN, Carnegie Mellon University, United States
PRASOON PATIDAR, Carnegie Mellon University, United States
YUVRAJ AGARWAL, Carnegie Mellon University, United States

ACM IMWUT 2023

Activity to Context

Activities detected:

(1) Taking over the Phone
(2) Exercise

Activity to Context

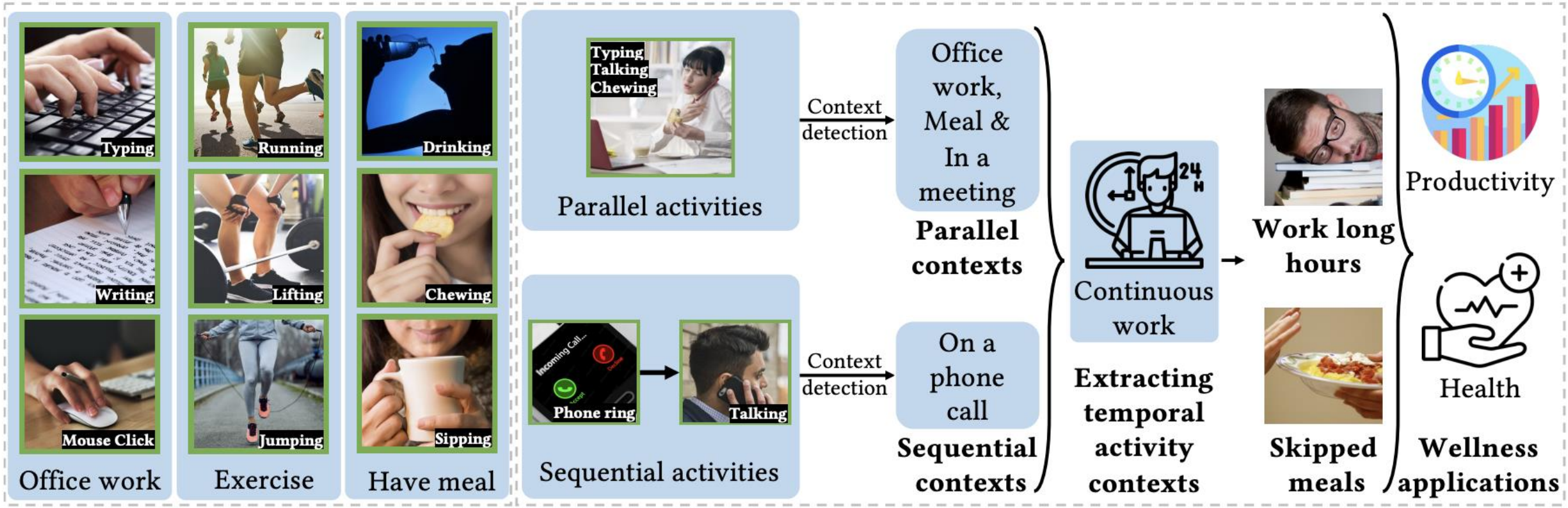
Activities detected:

(1) Taking over the Phone
(2) Exercise

Context:

"on a phone call for 20 minutes while exercising"

Inferring Context from Spatio-temporal Activity Sequence



Ontology-based Framework to Model Contexts

- An ontology is a structured framework to represent knowledge, typically consisting of classes (concepts), properties (relationships), and individuals (instances)

- Web Ontology Language (OWL)
  - A formal language designed for representing rich and complex knowledge about things, groups of things, and relations between them

- OWL 2 supports rich expressiveness by allowing complex class descriptions, property characteristics (like symmetry, transitivity), and constraints

OWL 2 Example

- Ontology for different types of animals and their characteristics, such as "*Mammal*," "*Bird*," and "*Fish*," along with some specific animals like "*Dolphin*" and "*Penguin*."

- Classes:
  - **Animal**: The root class representing all animals.
  - **Mammal, Bird, Fish**: Subclasses of Animal.
  - **Dolphin**: A subclass of Mammal.
  - **Penguin**: A subclass of Bird.

OWL 2 Example

- Properties:
  - **hasWarmBlood**: A property indicating whether an animal is warm-blooded.
  - **canFly**: A property indicating whether an animal can fly.
  - **livesInWater**: A property indicating whether an animal lives in water.

- Individuals:
  - **Flipper**: An individual of the class Dolphin.
  - **Pingu**: An individual of the class Penguin.

OWL 2 Example (OWL 2 Axioms)

1. Define Class Hierarchy (SubClassOf)

```
Mammal SubClassOf Animal
Bird SubClassOf Animal
Fish SubClassOf Animal
Dolphin SubClassOf Mammal
Penguin SubClassOf Bird
```

OWL 2 Example (OWL 2 Axioms)

2. Define Properties and Their Characteristics

- **Define Domains and Ranges**:
  - `hasWarmBlood Domain Animal, Range Boolean`
  - `canFly Domain Animal, Range Boolean`
  - `livesInWater Domain Animal, Range Boolean`
- **Assign Characteristics to Classes**:
  - `Mammal hasWarmBlood true` (All mammals are warm-blooded)
  - `Bird hasWarmBlood true` (All birds are warm-blooded)
  - `Fish hasWarmBlood false` (All fish are not warm-blooded)
  - `Bird canFly true` (All birds can fly, but ...)
  - `Penguin canFly false` (Penguins are an exception; they cannot fly)

OWL 2 Example (OWL 2 Axioms)

3. Define Individuals and Their Properties

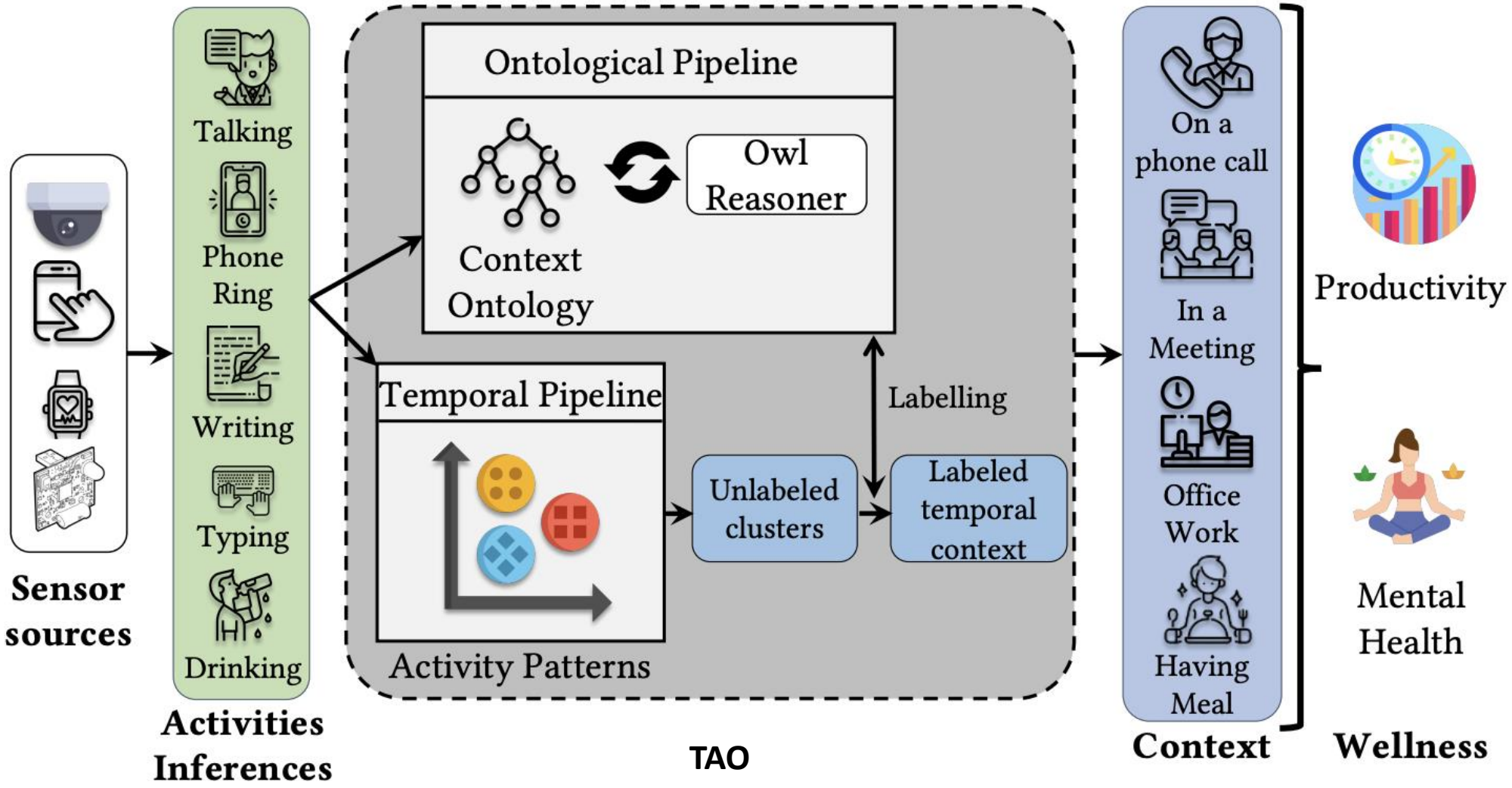
- **Flipper** (Instance of Dolphin):
  - `Flipper Type Dolphin`
  - `Flipper livesInWater true`
- **Pingu** (Instance of Penguin):
  - `Pingu Type Penguin`
  - `Pingu livesInWater false`

OWL 2 Functional Syntax

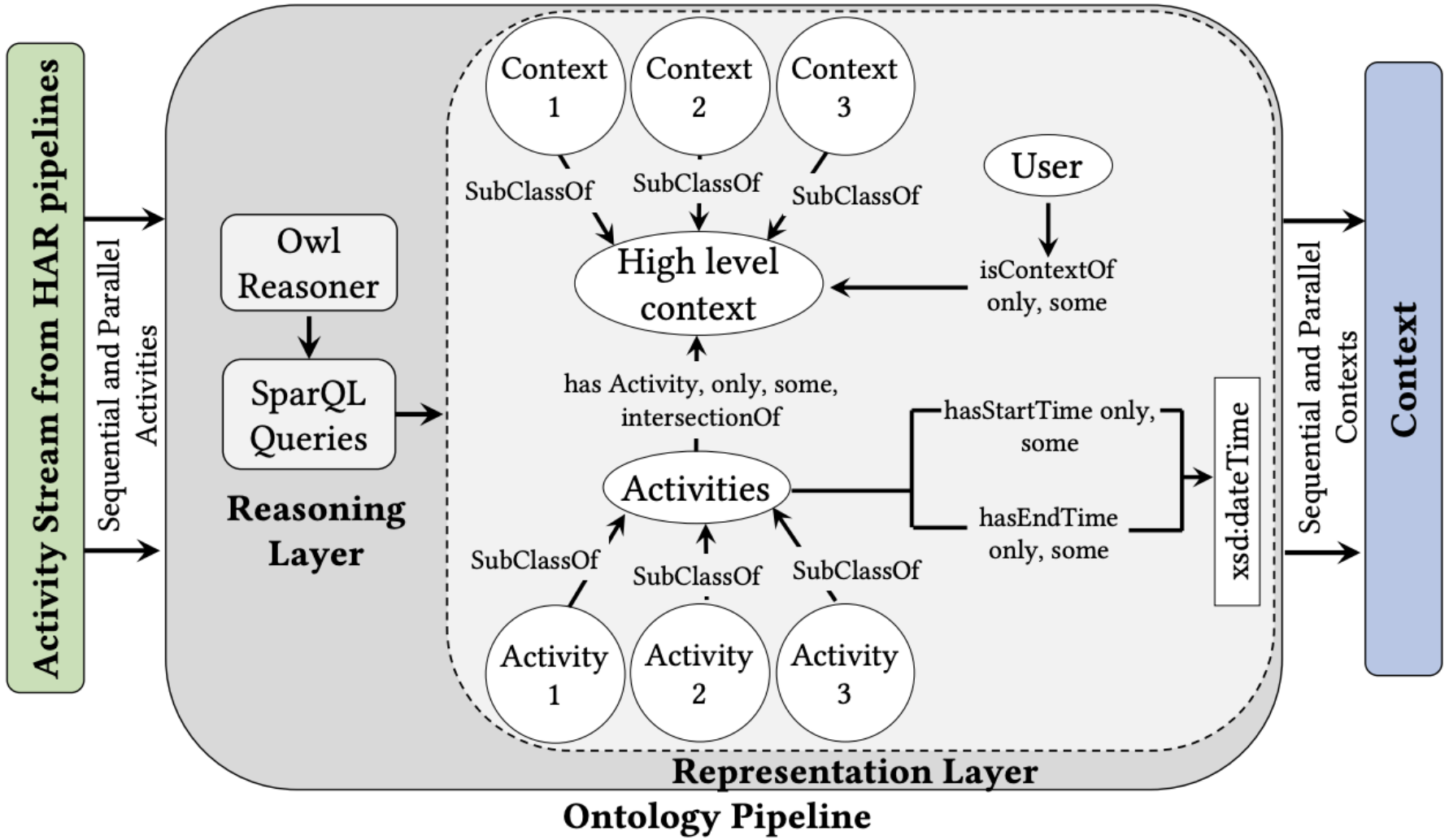
```
Class: Animal

Class: Mammal
    SubClassOf: Animal
    EquivalentTo: hasWarmBlood value true

Class: Bird
    SubClassOf: Animal
    EquivalentTo: hasWarmBlood value true
    SubClassOf: canFly value true

Class: Fish
    SubClassOf: Animal
    EquivalentTo: hasWarmBlood value false
```

```
Class: Dolphin
    SubClassOf: Mammal

Class: Penguin
    SubClassOf: Bird
    EquivalentTo: canFly value false

Individual: Flipper
    Types: Dolphin
    Facts: livesInWater true

Individual: Pingu
    Types: Penguin
    Facts: livesInWater false
```

TAO's System Architecture



Ontological Pipeline



Context Characterization

- Contexts based on the sequential activity patterns
  - **Context: *Phone Call***
    - **Phone ring -> Talking**

- Context based on parallel activity patterns
  - **Context: *In a Meeting***
    - **Talking**
    - **Writing**

- Context based on variation in the duration of activity
  - **Context: *On a zoom call***
    - **Talking (30 mins)**
  - **Context: *On a phone call***
    - **Talking (5 mins)**

Inferring the Contexts

- SPARQL Protocol and RDF (Resource Description Framework) Query Language
  - a query language and protocol used to retrieve and manipulate data stored in RDF format
  - RDF triples consist of three parts: *subject*, *predicate*, and *object*, which together represent relationships between resources
- **SPARQL Query Types**
  - **SELECT**: Used to retrieve specific data. It allows you to specify variables to extract and filter specific patterns in the RDF data.
  - **CONSTRUCT**: Allows you to create a new RDF graph based on the results of the query. This is useful for transforming data or creating derived datasets.
  - **ASK**: Returns a boolean result (TRUE or FALSE) indicating whether any data matches the query pattern. This is often used to check for the existence of data.
  - **DESCRIBE**: Retrieves information about a resource, typically returning all known triples associated with it.

Inferring the Contexts: Activity-Context Binding

```
CONSTRUCT {
?x a :OfficeWork .
    :hasActivity ?x;
    :hasActivity ?y;
}
WHERE{

?x a typing;
?y a talking;
BIND((URI(?x, ?y) as ?new) .
NOT EXISTS (?new a [] .)

}
```

SPARQL Query to Infer C: Office Work

```
CONSTRUCT {
?x a :OnAPhoneCall .
}
WHERE{
    ?x a phoneRing;
        :hasStartTime ?st;
        :hasEndTime ?et.
    ?t a talking;
        :hasStartTime ?st1;
        :hasEndTime ?et1.
    FILTER (contains(?st, ?et))
    FILTER (before(?st1, ?et1))
}
```

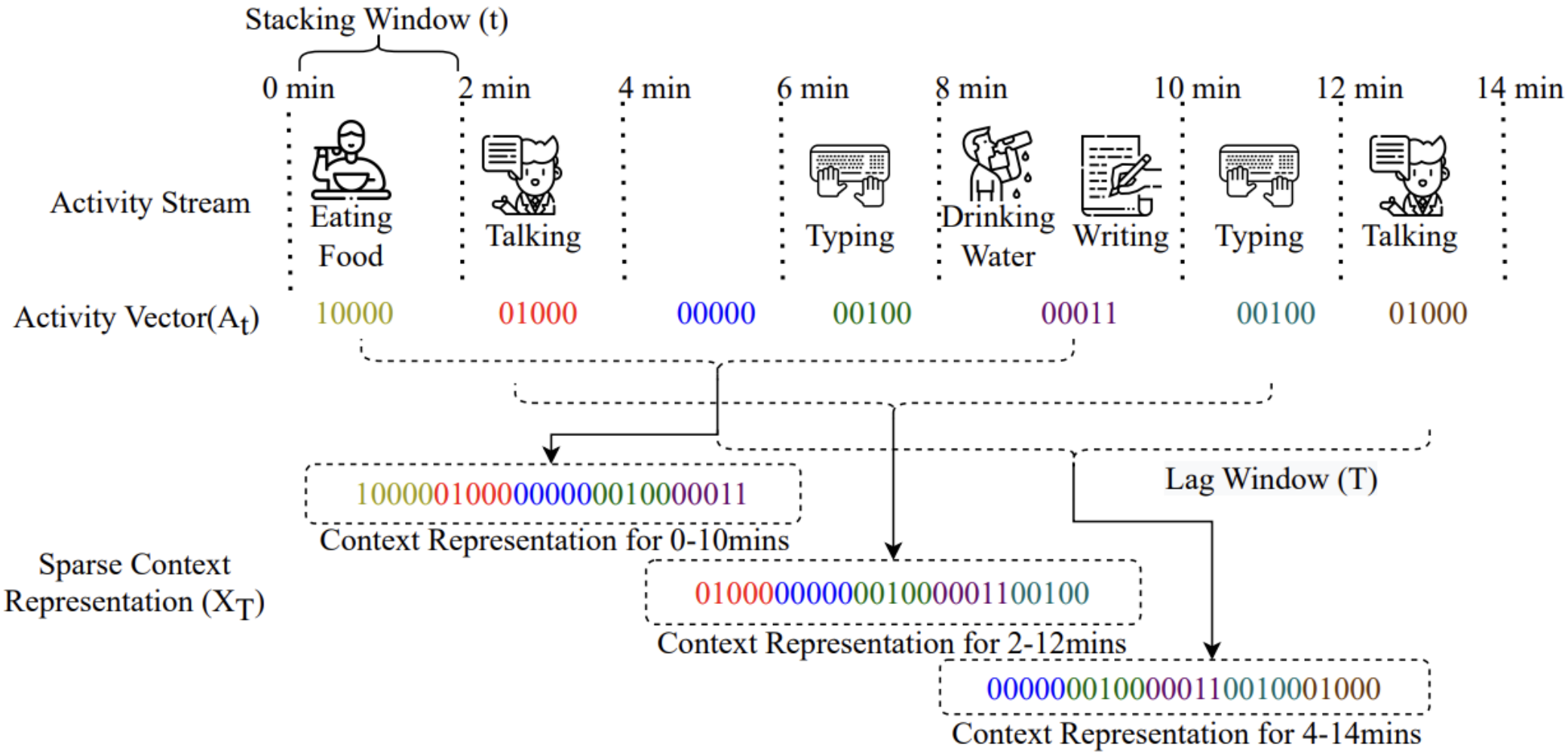
SPARQL Query to Infer C: On a Phone Call

Interleaved Contexts

- During daily activities, contexts can be overlapped
  - Reading and having a meal
  - Listening to music while vacuuming

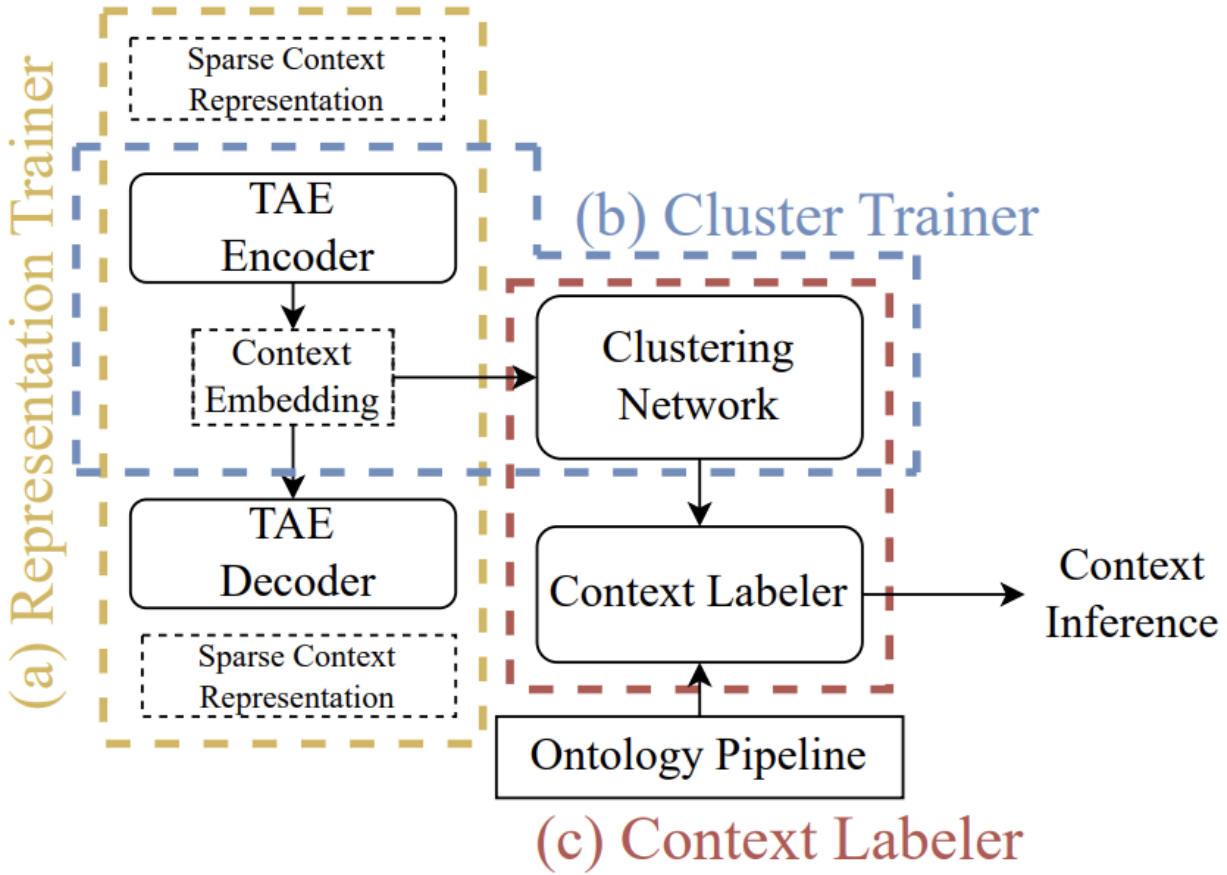
- TAO uses a temporal clustering-based approach to detect such interleaved and recurring patterns
  - Uses a stacking window (the time interval to wait for receiving activity inferences) and a lag window (incoming activities for a time interval)
  - **Examples**:
    - Eating is detected for 5 mins (stack window)
    - Eating, listening to music, watching a movie detected for 20 mins (lag window)

Interleaved Contexts



Extracting Contexts

- TAO uses a temporal autoencoder (TAE) to encode the similar activity sequences to learn dense context embedding from sparse context (activity) representations
  - A clustering is used to discover the optimum cluster count to extract overlapping context clusters
- Example: Jumping->jogging and jogging->jumping both represents similar contexts (exercise)



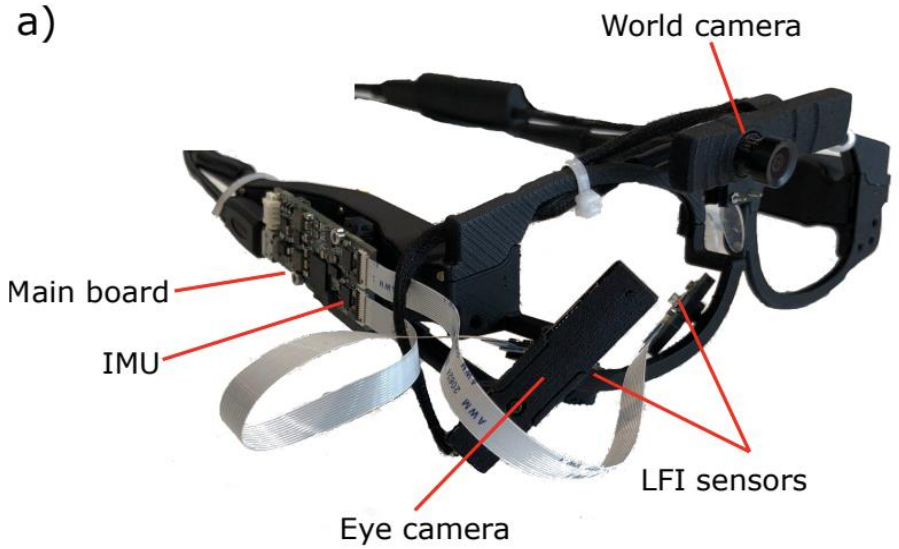
Summary

- Contextual information are useful to extract inferences in ubicomp
  - Combine multiple sensors together to extract context that can help in inferences

- Several use-cases in practical systems
  - AR/XR/MR devices: Use eye-gesture to infer emotions and correlate it with activity patterns
  - An interesting work to monitor eye gesture and use it as a context behind HAR

A CNN-based Human Activity Recognition System Combining a Laser Feedback Interferometry Eye Movement Sensor and an IMU for Context-aware Smart Glasses

JOHANNES MEYER, Corporate Sector Research and Advance Engineering, Robert Bosch GmbH, Germany
ADRIAN FRANK, Corporate Sector Research and Advance Engineering, Robert Bosch GmbH, Germany
THOMAS SCHLEBUSCH, Corporate Sector Research and Advance Engineering, Robert Bosch GmbH, Germany
ENKELEJDA KASNECI, Human-Computer Interaction, University of Tübingen, Germany




Happy
Learning!
Some resources
related to this topic