ECE209AS (Winter 2021)

Lecture 6: Miscellaneous Topics

Mani Srivastava
mbs@ucla.edu
Networked & Embedded Systems Lab
ECE & CS Departments
UCLA

Beyond Temporal: Spatial and Spatiotemporal

Deep Learning typically on Euclidean Data

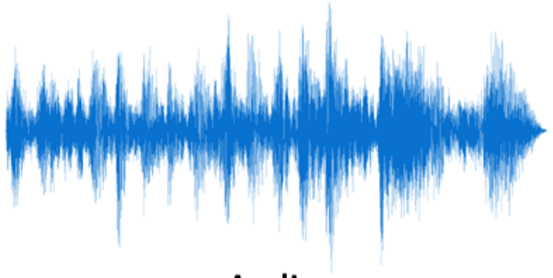
7

Numbers

Images

The quick brown fox jumps over the lazy dog

Text

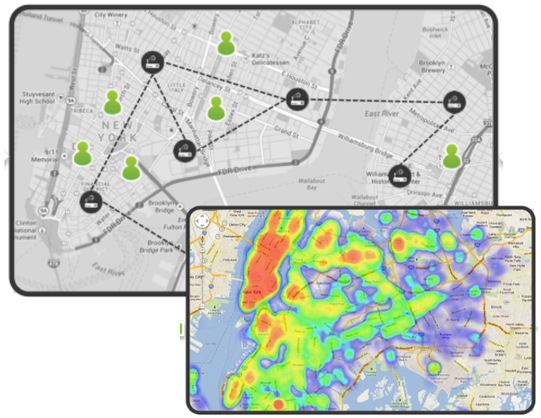


Audio

- Data sampled on a grid in 1D (speech), 2D (image), or 3D (video)
- Deep learning leverages statistical properties such as:
  - *stationarity* (due to shift invariance)
  - *locality* (local connectivity), and
  - *compositionally* (multiresolution structure)
- CNNs exploit these properties
  - extract local features that are shared across space or time to reduce parameters
  - impose some priors about natural data

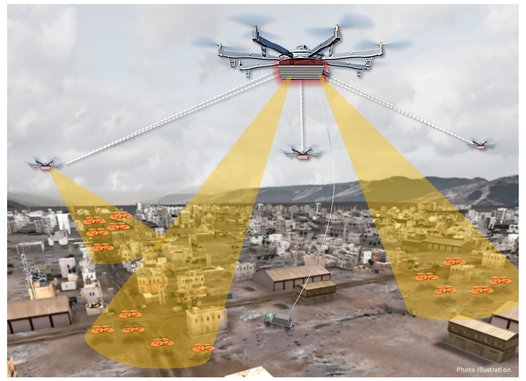
https://flawnsontong.medium.com/what-is-geometric-deep-learning-b2adb662d91d

Sensors often distributed in and moving through space

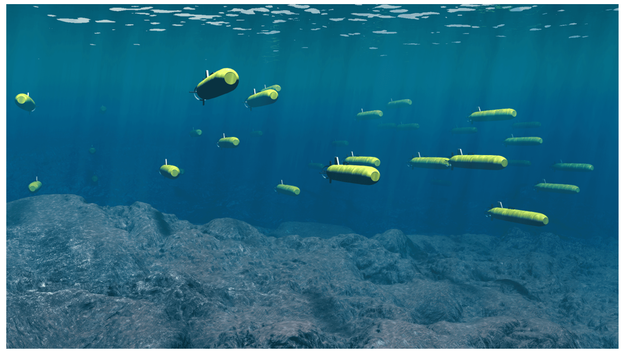


SONYC @ NYU

Wireless BAN



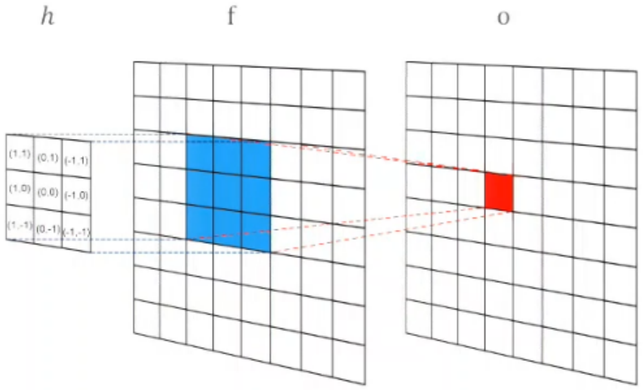
Aerial Urban Monitoring



UUV Swarm @ WHOI

- Measurements at (x, y, z, t) that do not fall on a regular grid in space or time
  - data may be on some underlying complex low-dimensional structure
    e.g. sensors on body, reports by people, measurements from sensors in cars on roads

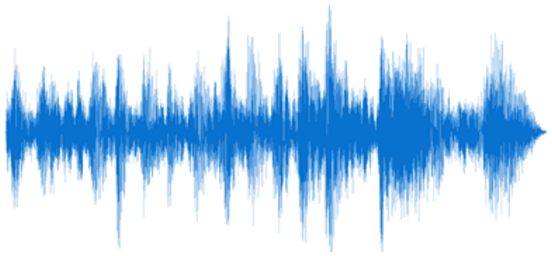
Deep Learning for Distributed Sensors



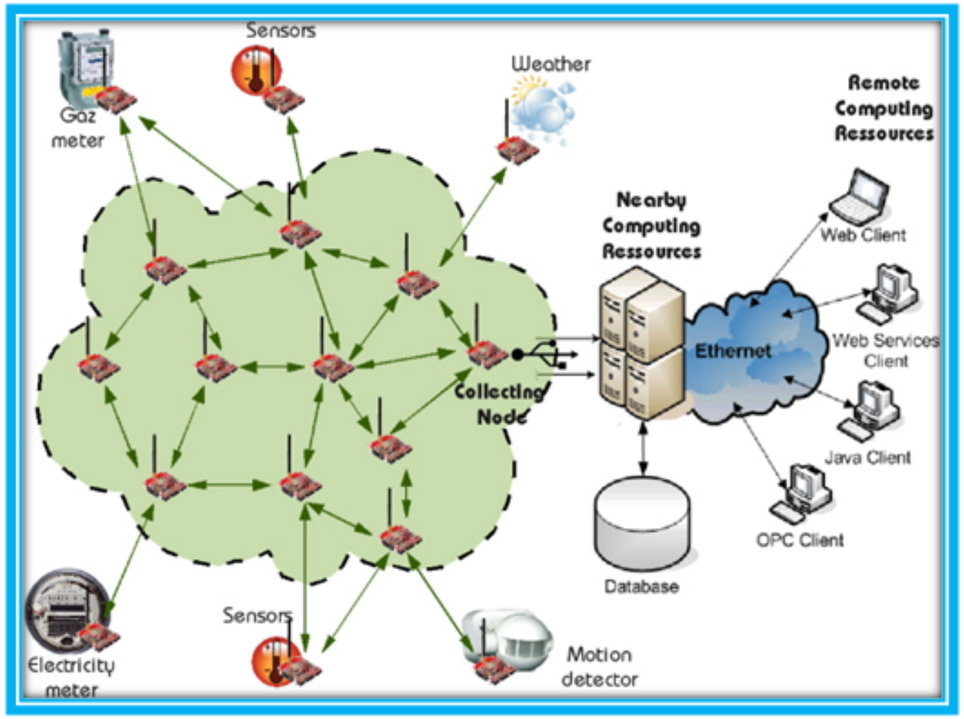
$$g(m,n) = (h * f)(m,n) = \sum_{i,j=-w}^{w} h(i,j) \cdot f(m-i, n-j)$$

CNN

The quick brown fox jumps over the lazy dog



RNN



What?

https://towardsdatascience.com/geometric-deep-learning-convolutional-neural-networks-on-graphs-and-manifolds-c6908d95b975

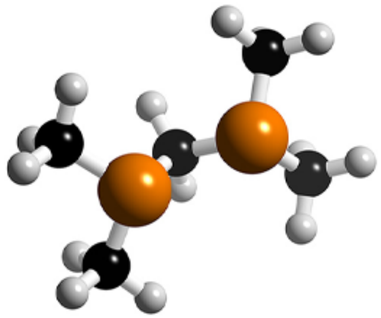
One Approach: Consider Measurements as a Set

- We saw this in Set Functions for Time Series (SeFT) for irregularly samples time series
  - Can generalize to spatial or spatiotemporal
  - Measurements in the most general case as $(\mathbf{v}, x, y, z, \rho, \theta, \phi, t)$

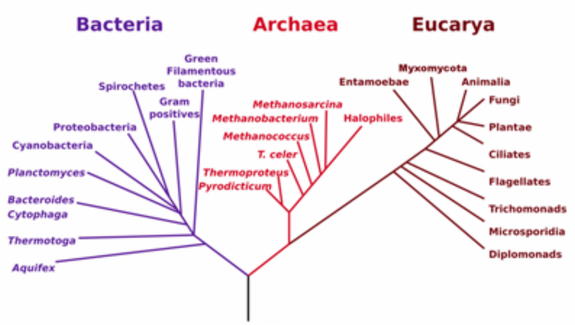
- Desired properties
  - **Unordered**
    - must be permutation invariant
  - **Interaction among measurements**
    - model should capture local spatial or spatiotemporal structures, and combinatorial interactions among local structures
  - **Invariance under spatial transformations**
    - the learned representation should be variant to spatial transformations such as rotation and translation

Another Approach: Geometric Deep Learning (GDL)

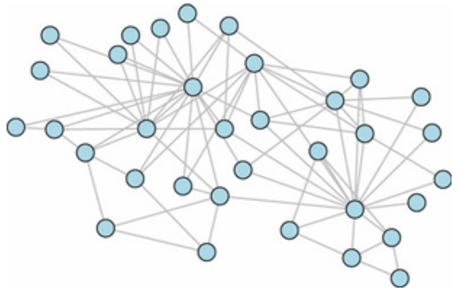
- Generalizes convolution neural networks to data that is over non-Euclidean spaces such as manifolds, graphs etc
- Perform tasks such as
  - classify nodes with similar characteristics
  - classify graphs, e.g. dangerous molecules, buggy circuit, 3D objects from point clouds
  - 3D shape correspondence to 2D images
- Data with inherent relationships, connections, and shared properties
- Useful for DL on sensor networks, social networks, genetics, brain, 3D structures etc.
- Of interest in IoT/CPS: data on a graph
  - graphs provide a relational inductive bias



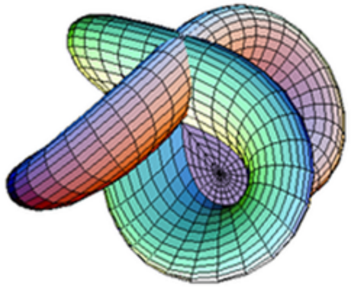
Molecules



Trees



Networks



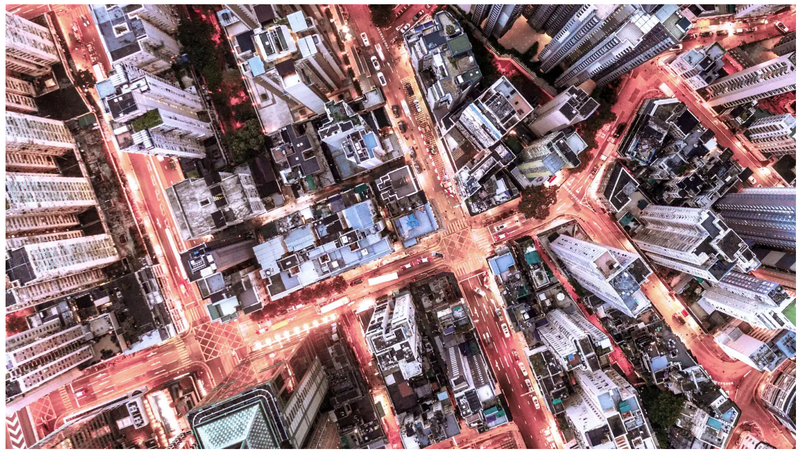
Manifolds

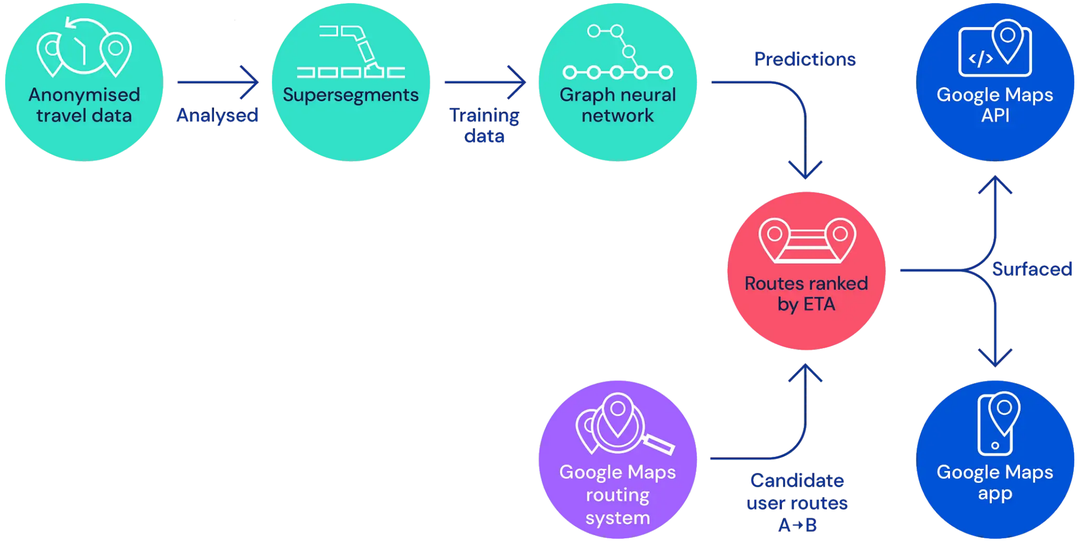
https://flawnsontong.medium.com/what-is-geometric-deep-learning-b2adb662d91d

Example: Traffic Prediction via Neural Networks on Graph Data

- Forecasting traffic speed, volume or the density of roads in traffic networks is fundamentally important in a smart transportation system
  - e.g. ETA in Google Maps

- Traffic network as a spatial-temporal graph
  - nodes are sensors installed on roads
  - edges are measured by the distance between pairs of nodes
  - each node has the average traffic speed within a window as dynamic input features





https://deepmind.com/blog/article/traffic-prediction-with-advanced-graph-neural-networks

Deep Learning on Static Graphs: *Graph Embedding* Approach

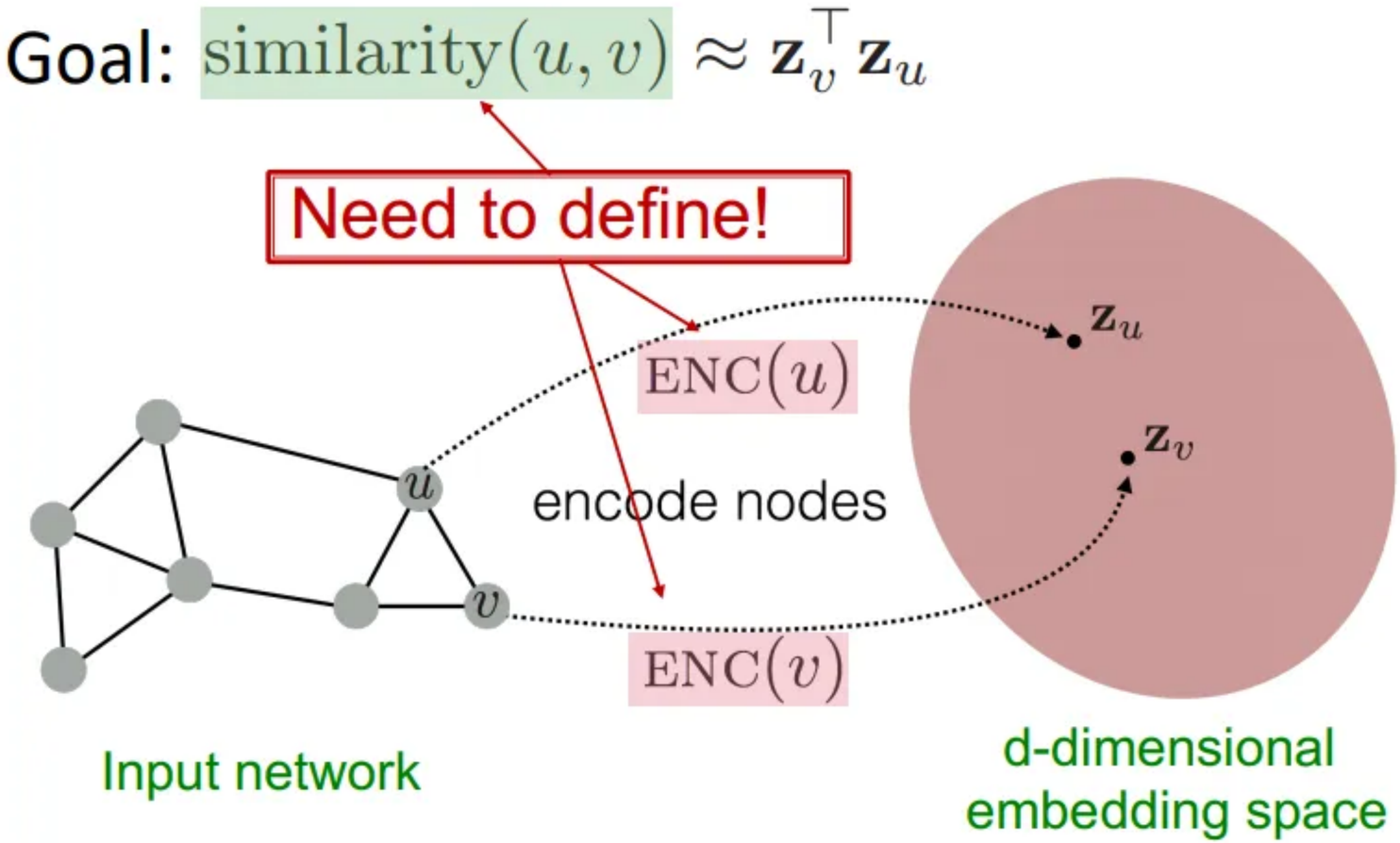
- A static graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ comprises nodes $\mathscr{V} = \{1, \ldots, n\}$ and edges $\mathscr{E} \subseteq \mathscr{V} \times \mathscr{V}$, which are endowed with features $\mathbf{v}_i$ and $\mathbf{e}_{ij}$ for all $i, j = 1, \ldots, n$ respectively

- *Idea:* convert an arbitrary graph into a form that ML algorithms can digest, namely vectors in a continuous space

- *Graph Embedding:* transform nodes, edges, and their features into vector space (a lower dimension) whilst maximally preserving properties like graph structure and information
  - Variety of ways to go about embedding graphs, at different levels of granularity - node level, sub graph level, and via strategies such as graph walks
  - E.g. DeepWalk, Node2Vec, Graph2Vec, Structural Deep Network Embedding (SDNE), Large-scale Information Network Embedding (LINE), and many others

https://towardsdatascience.com/overview-of-deep-learning-on-graph-embeddings-4305c10ad4a4

The Graph Embedding Idea



https://neptune.ai/blog/graph-neural-network-and-some-of-gnn-applications

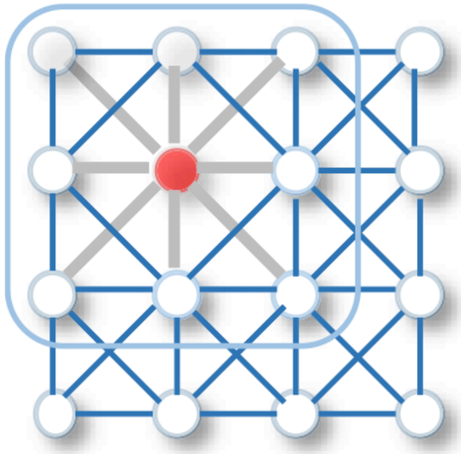
Deep Learning on Static Graphs: *Graph Convolution* Approach

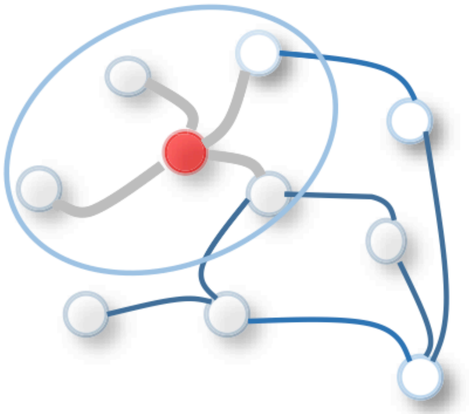
- A static graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ comprises nodes $\mathscr{V} = \{1,\ldots,n\}$ and edges $\mathscr{E} \subseteq \mathscr{V} \times \mathscr{V}$, which are endowed with features $\mathbf{v}_i$ and $\mathbf{e}_{ij}$ for all $i,j = 1,\ldots,n$ respectively

- *Idea:* generalize convolution so that it can directly operate on a representation of the graph that retains its structure and features

- *Intuition:* At a high level, convolutions use "kernels" to aggregate information from surrounding or adjacent entities

- Two approaches to generalizing convolutions to graphs: *Spectral* vs. *Spatial*





https://towardsdatascience.com/graph-convolutional-networks-for-geometric-deep-learning-1faf17dee008

Spectral Graph Convolutional Networks

- Based on ideas from a field called Graph Signal Processing which applies methods such as Fourier Transforms to graphs to create spectral domain representation
  - allows talking about properties of graphs such as bandwidth, smoothness, etc.
- Definition: Laplacian matrix of a simple (one edge or no edge between a pair of vertices) graph $L = D - A$ where D is the degree matrix and A is the adjacency matrix

$$L_{i,j} := \deg(v_i) \text{ if } i = j; -1 \text{ if } i \neq j \text{ and } v_i \text{ is adjacent to } v_j; 0 \text{ otherwise}$$

- Convolutions over graphs computed by:
  1. Finding the Eigen decomposition of L to convert the graph into spectral domain (GFT)
  2. Apply Eigen decomposition to the specified kernel
  3. Multiply the spectral graph and the spectral kernel
  4. Return results to the original spatial domain (inverse GFT)
- Many spectral methods: ChebNets, Kipf and Welling's GCNs, CayleyNets, MotifNets

https://towardsdatascience.com/graph-convolutional-networks-for-geometric-deep-learning-1faf17dee008

Spatial Graph Convolutional Networks

- Basic approach: Create an *embedding* $\mathbf{z}_i$ on the nodes by learning a local aggregation rule of the form

$$\mathbf{z}_i = \sum_{j \in \eta_i} h(\mathbf{m}_{ij}, \mathbf{v}_i) \qquad \mathbf{m}_{ij} = \text{msg}(\mathbf{v}_i, \mathbf{v}_j, \mathbf{e}_{ij})$$

  which is interpreted as message passing from the neighbors j of i.

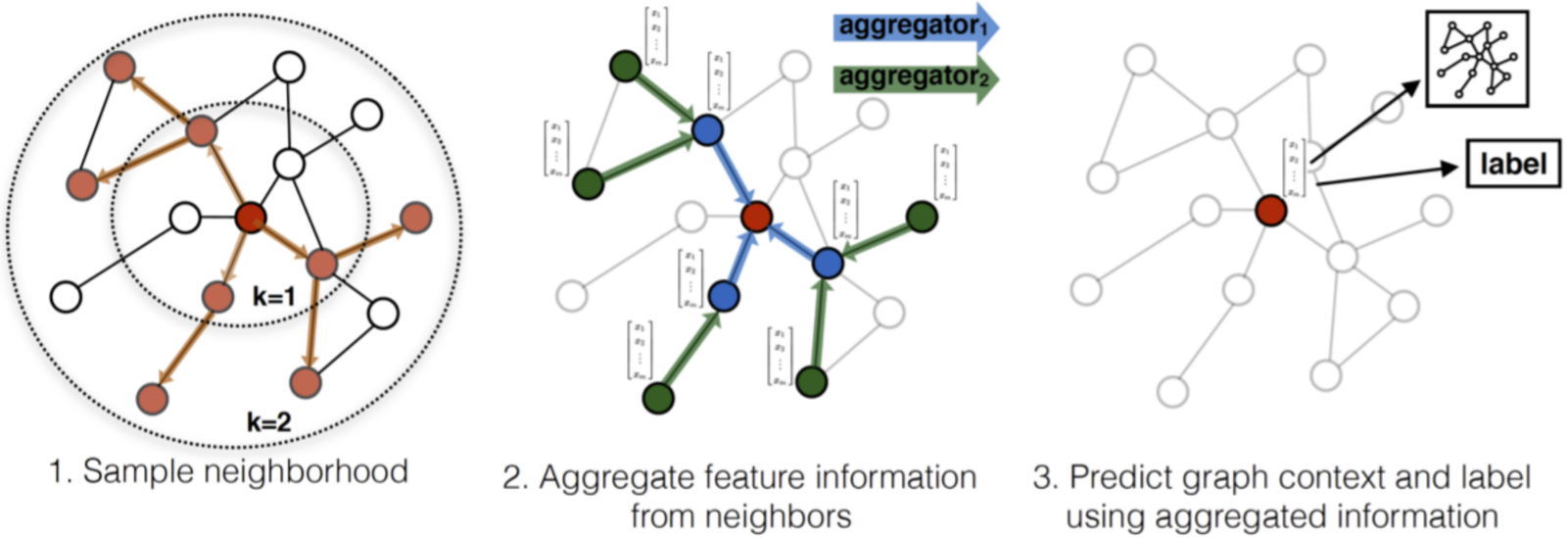
  Here $\eta_i = \{j : (i,j) \in \mathscr{E}\}$ denotes the neighborhood of node i and msg and h are learnable functions

- Embeddings can then be used for various tasks
- Handle cycles by iterating multiple times till embedding converge
- Hybrids that combine ideas from spatial and spectral approaches

https://towardsdatascience.com/graph-convolutional-networks-for-geometric-deep-learning-1faf17dee008

Example of Spatial Graph Convolution: GraphSage

- Three steps:
  1. **Neighborhood Sampling:** Find immediate neighborhood of depth k
  2. **Aggregate Features of Neighbors:** Mean aggregation, LSTM aggregation (in random order), Pooling aggregation (Max pooling does the best)
  3. **Prediction**: e.g. node class, structure/context
     - Target node uses the aggregated neighborhood node features to make a prediction via a neural network (whose weights are learnt during supervised training)



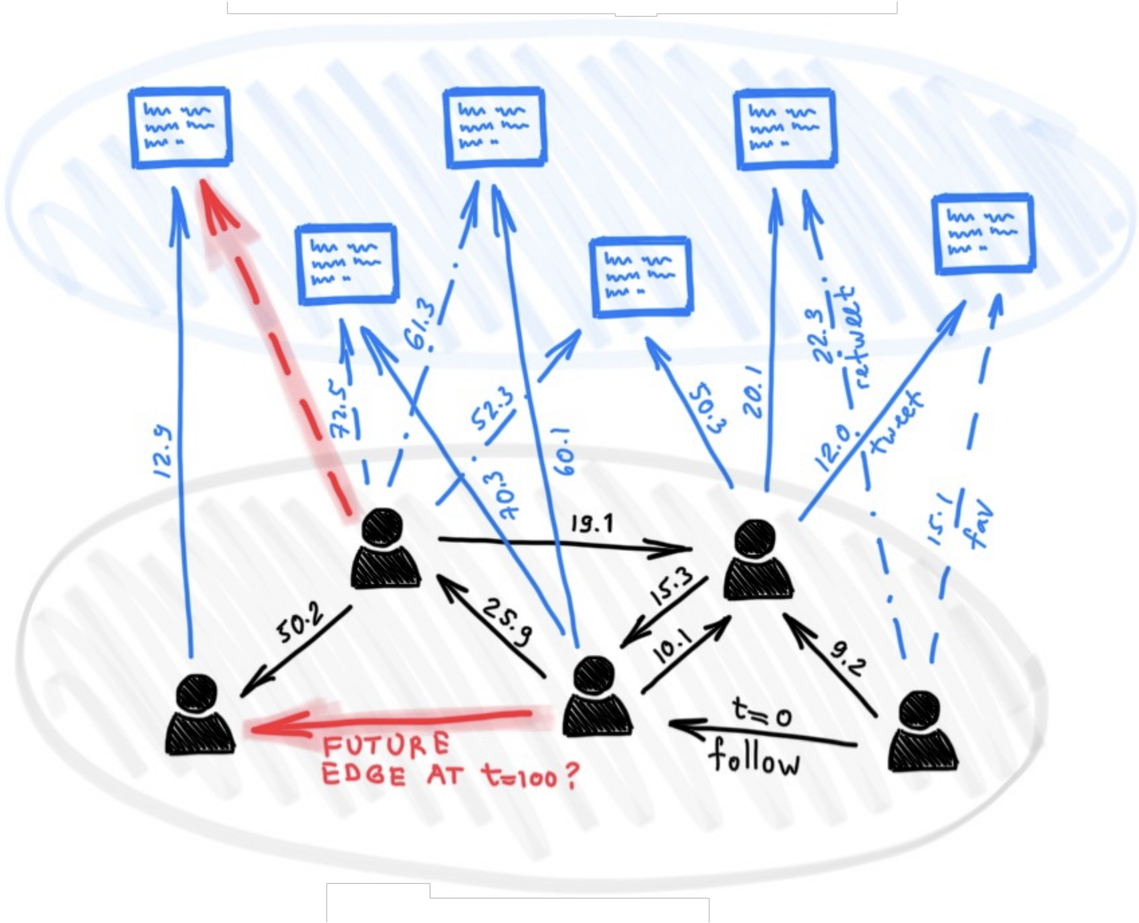
1. Sample neighborhood

2. Aggregate feature information from neighbors

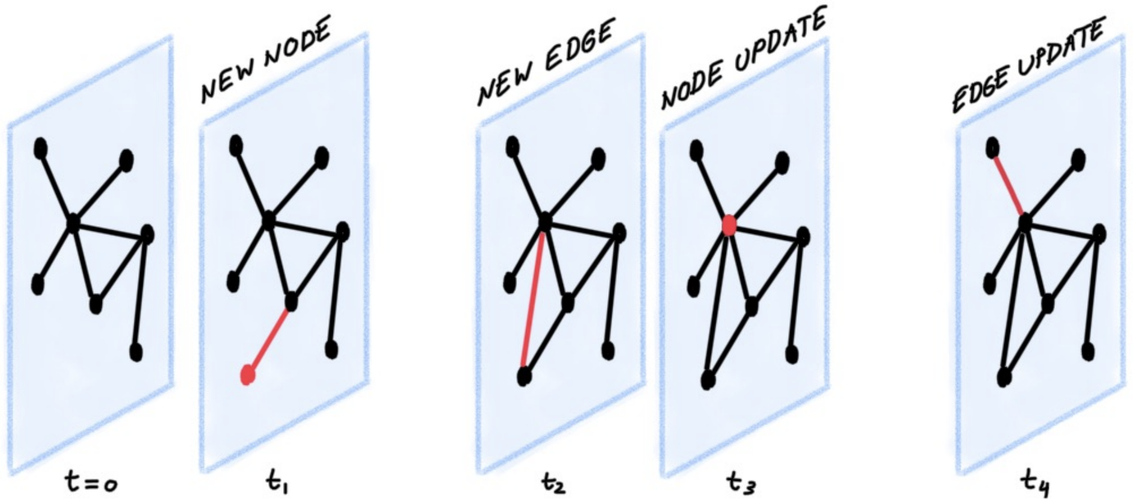
3. Predict graph context and label using aggregated information

https://towardsdatascience.com/graph-convolutional-networks-for-geometric-deep-learning-1faf17dee008

Predictions on Dynamic Graphs



- Traditional Graph Neural Networks (GNNs) developed for static graphs that do not change over time
- However often graph structure changes over time, e.g. mobile nodes, transient nodes, changing attributes



https://towardsdatascience.com/temporal-graph-networks-ab8f327f2efe

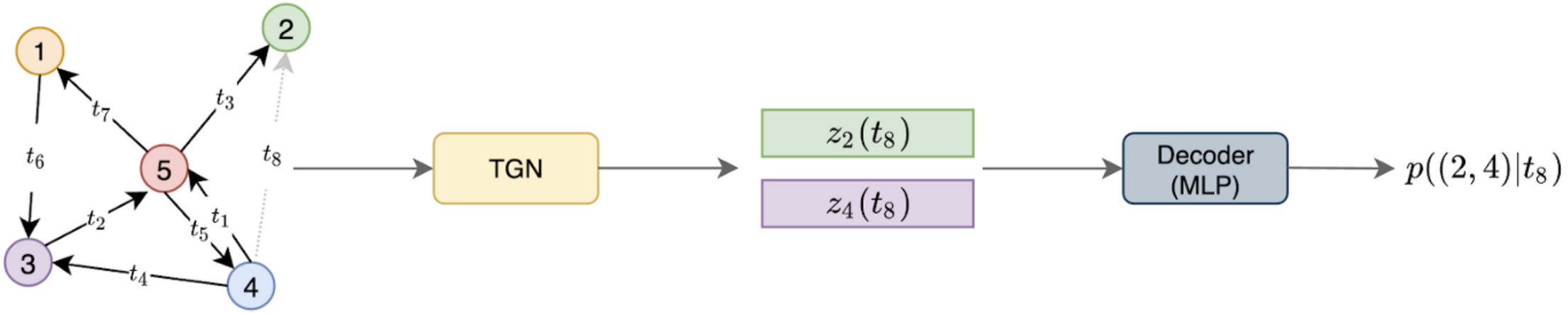
Representing Dynamic Graphs

- Discrete-time dynamic graphs (DTDG)
  - Sequences of static graph snapshots taken at intervals in time

- Continuous-time dynamic graphs (CTDG)
  - More general
  - Represented as timed lists of events
    - edge addition or deletion
    - node addition or deletion
    - node or edge feature transformations

Temporal Graph Networks (TGNs) for Dynamic Graphs

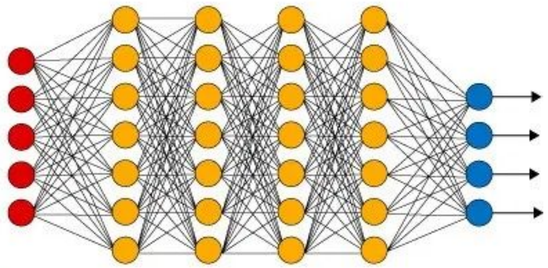
- A general encoder architecture developed at Twitter for tasks such as predicting future interaction among nodes. It has the following main components
  - **Memory**: stores compressed representations $\mathbf{s}_i(t)$ of every node's past interactions
    - When a new node appears, we add a corresponding state initialized as a vector of zeros
  - **Message Function**: analogous to message passing GNN, upon interaction between two nodes, generates messages for them
  - **Memory Updater**: uses a RNN to update the state of a node with the new message
  - **Node Embedding**: obtained by a graph aggregation over the spatiotemporal neighbors



Rossi, Emanuele, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. "Temporal graph networks for deep learning on dynamic graphs." arXiv preprint arXiv:2006.10637 (2020).
https://arxiv.org/pdf/2006.10637

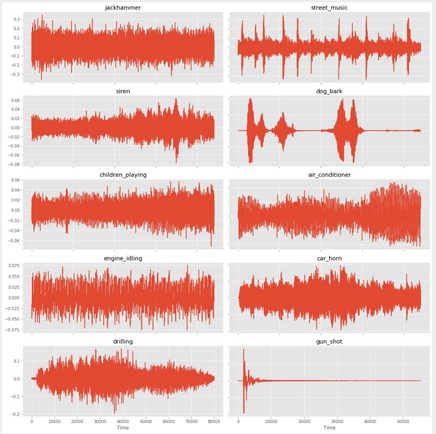
Neurosymbolic Models: Data + First Principles Knowledge

Sensing with Models Learnt from Data Very Successful

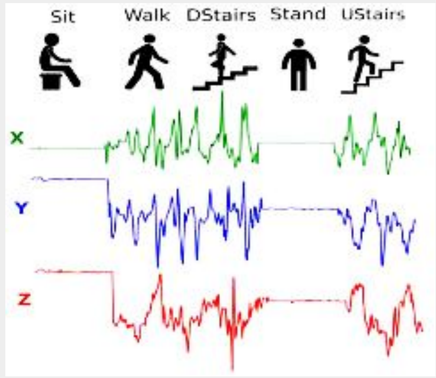


Deep Learning is **faster**, and **more accurate** than humans!

Audio Event Detection



Activity Classification



Visual Anomaly Detection

But Face Challenges Too

- Models are opaque, i.e. not inherently interpretable
  - generating satisfactory explanations is hard
- Difficult to adapt to changed operating environment (“domain shift”)
  - different sensor, different sensing modality, different weather condition, different platform
- Need a lot of data (usually labeled) and time to train
  - becomes a problem when operating in dynamic or novel environments
  - rare and complex events (i.e. higher order spatial and temporal interactions)
- May fail to obey external constraints and expectations
  - relating to scientific theories, safety, bias, privacy, and similar properties
  - worse, may even amplify the bad

Models based on First Principles Knowledge

- First principles knowledge can take many forms
  - Models from fundamental sciences (physics, chemistry, biology)
  - Rules provided by subject matter experts
  - Laws and regulations

- It may be represented in many ways
  - Program, simulators, logic formulae, hardware

- But models based on first principles knowledge alone unsatisfactory
  - Incomplete, approximate, slow (e.g. solve PDEs)

Example: Detecting Complex Events (CE)

Unsanitary Operation

Coordinated Attack

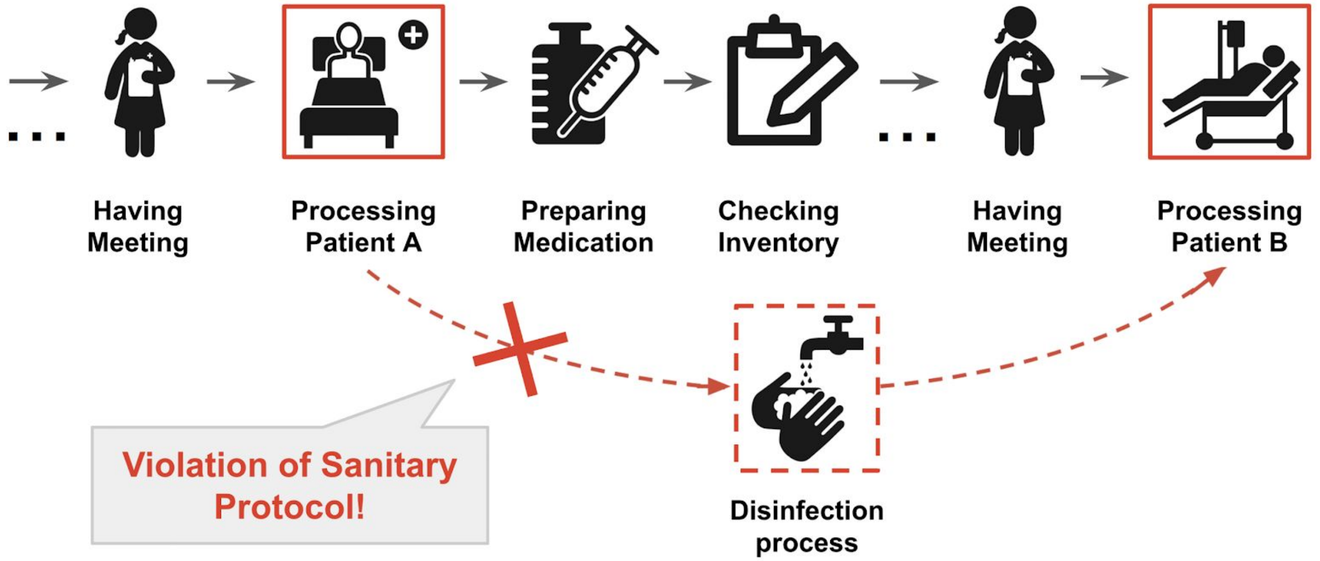
Unattended Bag

Traffic Rule Violation

Detecting Complex Event (CE) is a challenging task:

- Lack of **large scale dataset** to train an end-to-end model
- Requires **raw data processing ability** to extract informative features
- Need **high-level reasoning ability** for analyzing larger spatial and temporal dependencies

Example: Detecting Complex Events (CE)



Example:

Unsanitary Nursing Operation

A nurse forgets to wash their hands between processing different patients.

Complex Event:

- **Pattern** of **simpler events** taking place over a long span of time
- Look for pattern in the large number of events:: **needle in the haystack problem**
- Involve events from **many different sensors**

CE Detection with Deep Learning Models

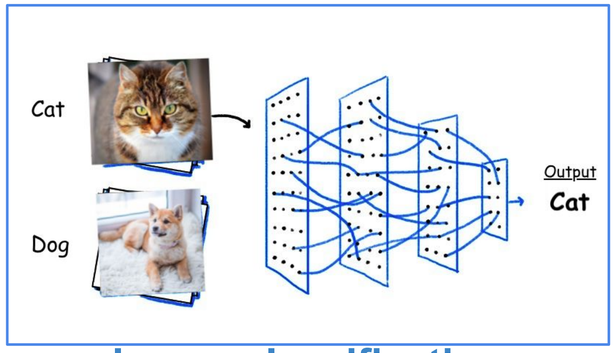
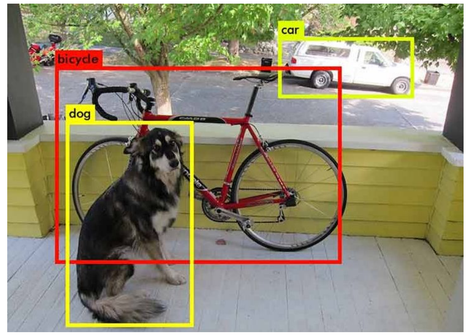
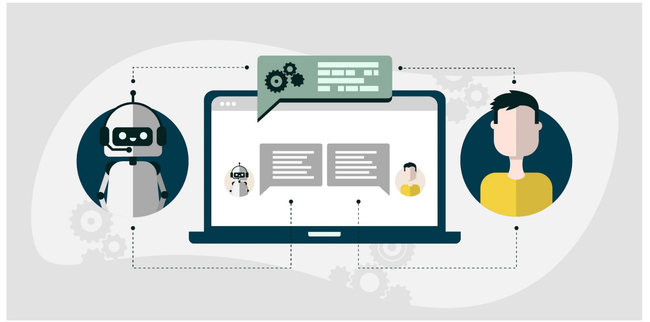


Image classification



Object detection & tracking



Language modeling

Work well but:

- **Require large amount of labeled data** which is a challenge with sensors that cannot be labeled retrospectively via crowdsourcing
- Despite advances such as LSTM, Attention etc. the **memory remains limited** for purposes of complex event sensing (high rate, long time spans)

CE Detection with Deep Learning Models is Challenging

Modeling long-term dependencies requires memory...

Models	Related Work	Input Length
RNN / LSTM and Variants	Bi-LSTM[Singh et al. CVPR'16] CRNN[Cakir et al.]	• Reasonable limit of 250-300 time step with large LSTM model • A few seconds (4-10) on visual and audio analytics tasks
Convolution Based	TCN[Lea et al. ECCV'16]	• A larger receptive field of about 10s on video-based action classification
Attention Mechanism	TransformerXL[Dai et al. Arxiv'19], BERT, GPT model, Informer [Zhou et al. AAAI'21]	• Time-series forecasting on hundreds to 1K of steps. • NLP: sentence → paragraph → article

Gap: DL models and sensing long-term complex patterns

- With different sampling rates, input length can grow to more than thousands
- Existing modern models require large dataset of population-scale
- Memory intensive: knowledge are not compactly stored

CE Detection using Knowledge-based Models

Knowledge-based system (KBS) captures the knowledge of human experts to support making inferences and decisions.

Methods	Related Work	Features
Bayesian Network / DBN/ PGM	AEDvBN[Hsueh UMEDIA'15] GM-DNN[Le et al. MASS'19] KBAR[Zeng eccv'10] [Anitha CC'19]	• Probabilistic graphical model that represents the probabilistic relationship among random variables. • Structure learning or configured by field expert.
Logic Programming	ProLog, ProbLog[De Raedt, ijcai'07], Pyro[Bingham et al. jmlr'18], DeepProbLog[Manhaeve '18]	• Program written in logic from, expressing facts and rules about some problem domain • Recent framework incorporates deep learning by means of neural predicates
Complex Event Processing (CEP)	SASE[Gyllstrom, Arxiv'06] Cayuga[Demers, cidr'07] Siddhi[Suhothayan, gpe'11] IoMT[Aslam, IEEE Access'18]	• Takes user's definition of patterns to process streaming events. • Use finite automata to capture the temporal dependencies of infinite range.

Problem: work well with structured data only, don't handle data quality issues well

CE Detection with High-Dimensionality Unstructured Sensor Data

Current CE detection for **high-dimensional unstructured** data:

- Relies on **human effort.**
- **Not performed in real-time.**

CE definition:

“ A nurse forgets to wash their hands between processing different patients ”

Detected complex events:

“ Unsanitary Operation at 2:10pm in Room 279 ”

Current State of Practice

Deep-Learning based Models:

- Process high-dim sensory data
- Excel on perception task: e.g. modeling short term patterns
- Data-consuming, inefficient in capturing long-term dependencies

Logic-based Methods:

- Work with structured data in a human-understandable way
- Represent complex spatial & temporal dependencies efficiently and effectively
- Cannot handle unstructured data

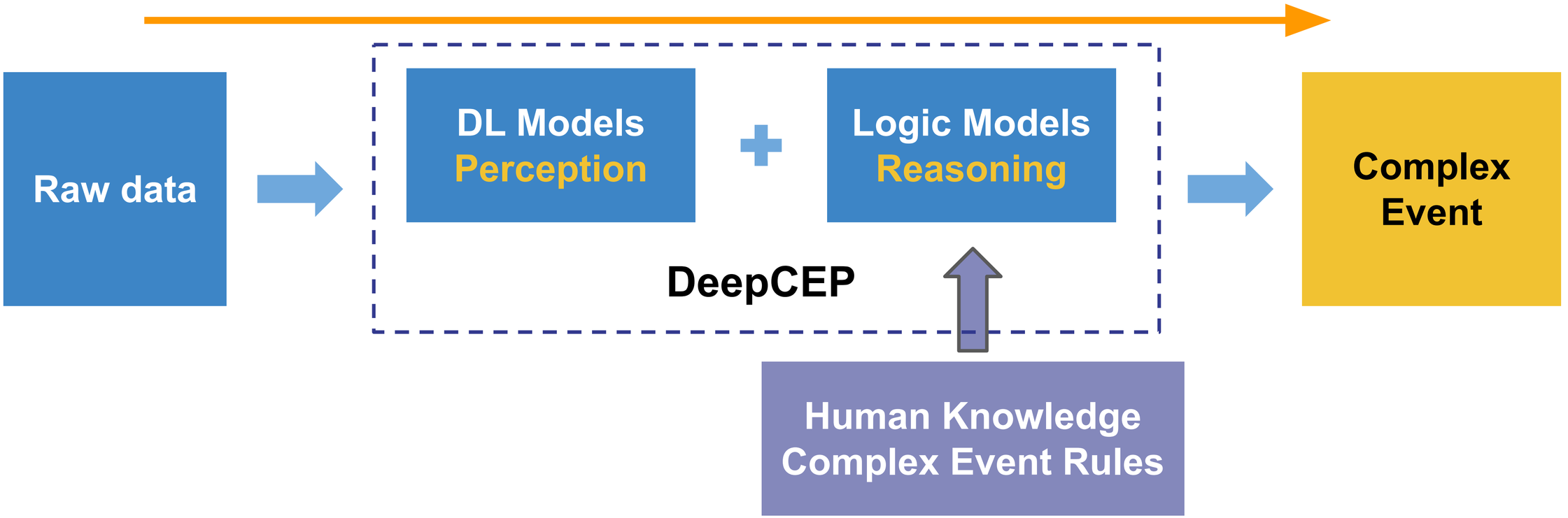
Perception

Reasoning

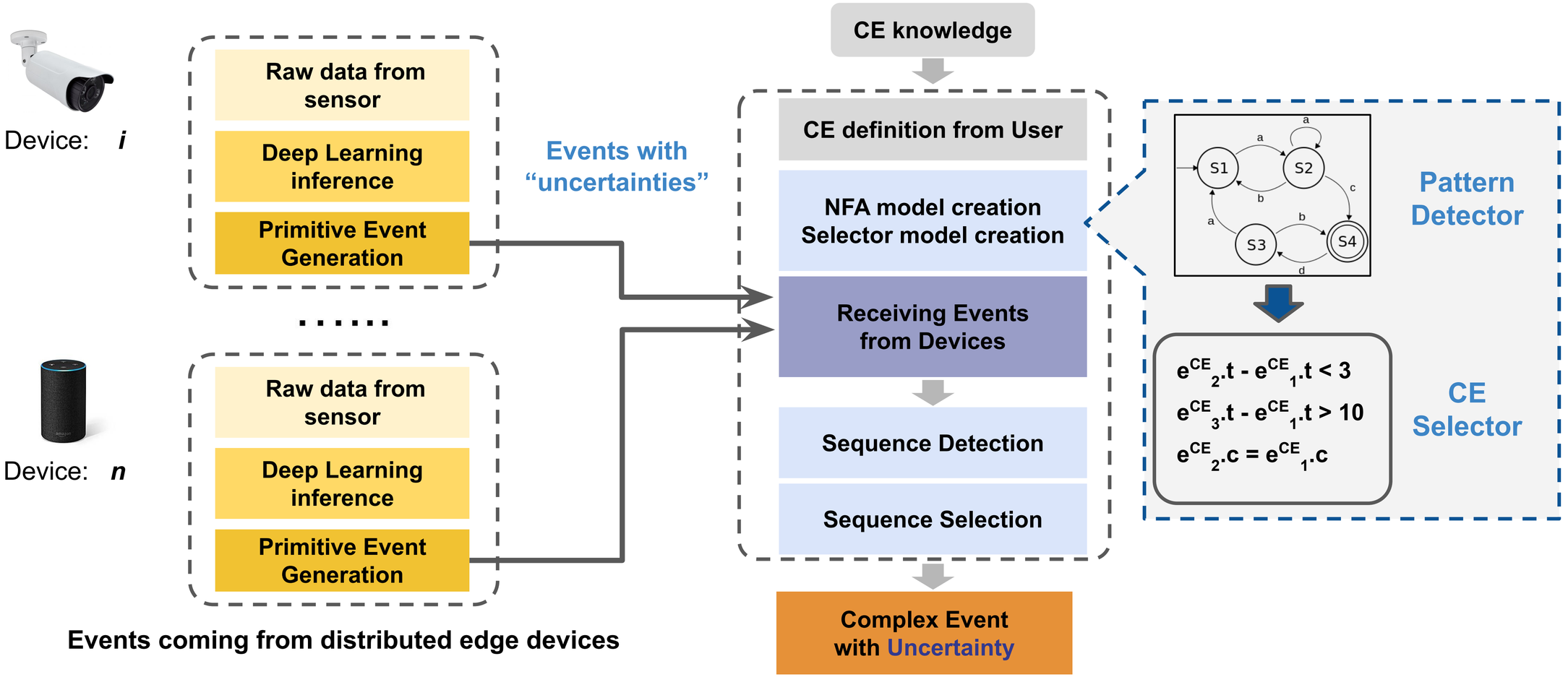
A hybrid approach?

- Inspired by how human process CE
- Combine the power of the DL and Logic approaches.

A “Neurosymbolic” Approach to CE Detection



The DeepCEP System Architecture



Complex Event Grammar for DeepCEP

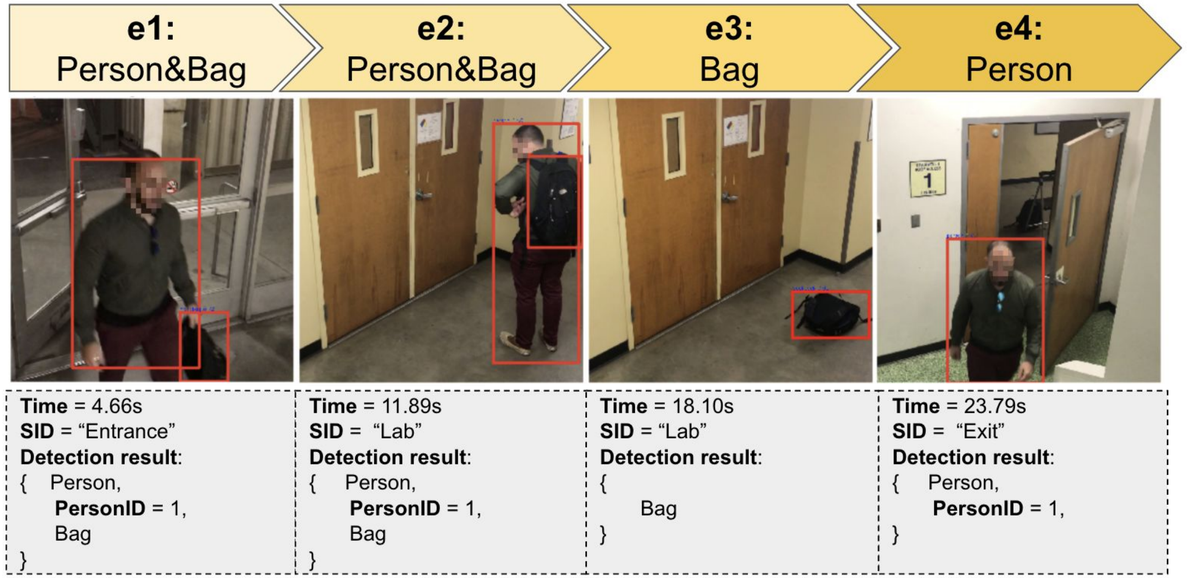
⟨*complex-event*⟩ ::= ⟨*input-title*⟩ ⟨*complex-event-title*⟩ ⟨*format-pattern*⟩ ⟨*constraint*⟩? **EOF**;
⟨*input-title*⟩ ::= **INPUT :** ⟨*event-stream-source-id*⟩;
⟨*complex-event-title*⟩ ::= **CE :** ⟨*complex-event-stream-id*⟩;
⟨*format-pattern*⟩ ::= ⟨*combo-format*⟩ **: {** ⟨*event-list*⟩+ **};**
⟨*combo-format*⟩ ::= **FORMAT-SEQ** | **FORMAT-PATTERN** | **FORMAT-PATTERN-WITHOUT**;
⟨*event-list*⟩ ::= ⟨*event*⟩ (**,** ⟨*event*⟩)*;
⟨*event*⟩ ::= ⟨*event-id*⟩ **:** ⟨*event-type-id*⟩;
⟨*constraint*⟩ ::= **CONSTRAINT : {** ⟨*logical-predicate-list*⟩ **};**
⟨*logical-predicate-list*⟩ ::= logic-expression (**,** logic-expression)*;

INPUT: camera-feed

CE: unattended-bag

PATTERN: {e1:person bag e2:person bag e3:bag e4:person}

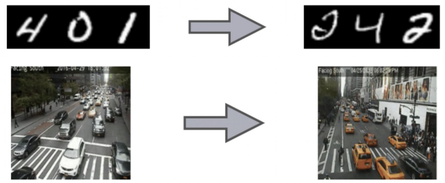
Constraints:{
e1.sid = 'Entrance',
e2.sid = e3.sid = 'Laboratory', e4.sid = 'Exit',
e1.person id = e2.person id = e4.person id
}



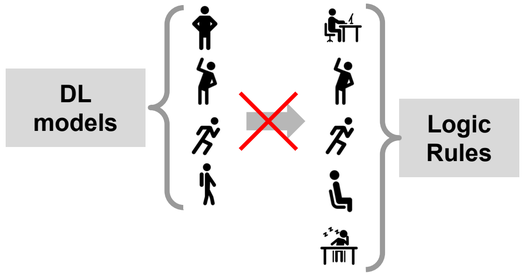
Training Neurosymbolic Models

- Assumption that pre-trained DL models for detecting simple events are available

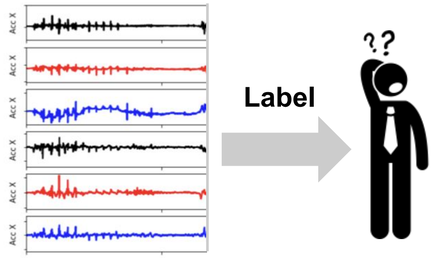
Pre-trained **perception models** may not perform well in **personalized environments**.



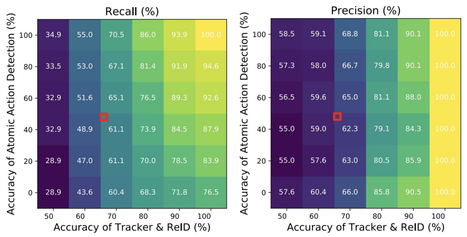
The **output directory** of pre-trained DL models **may not include** the **necessary events** for defining complex events.



Annotating the sensory data at **event-level** is challenging.

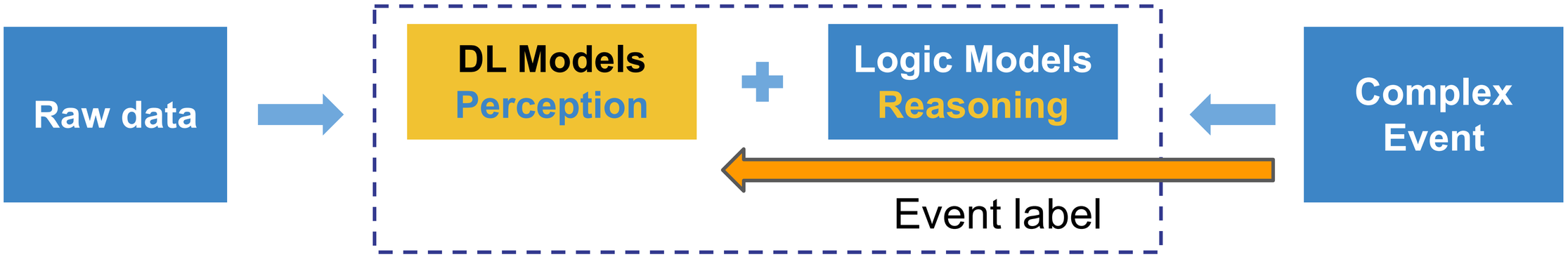


The performance of **hybrid systems relies on** the of **perception models** for simple events.



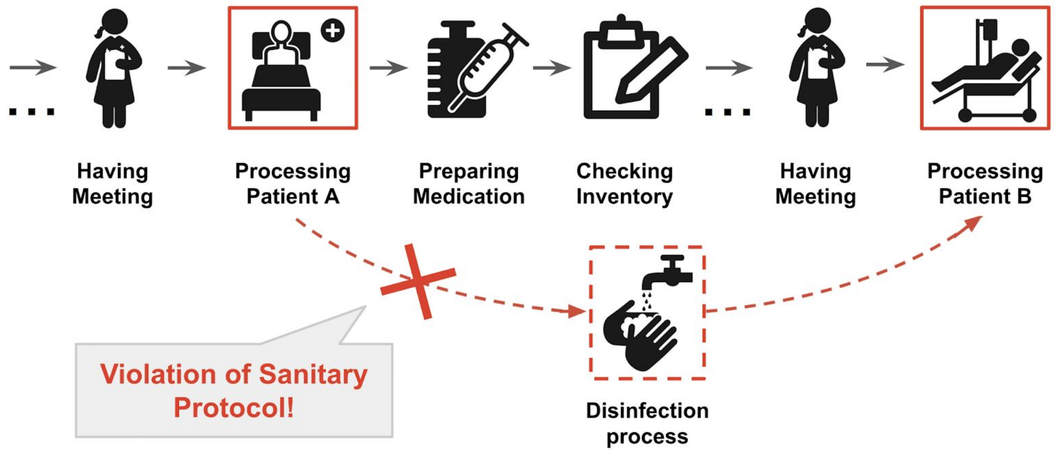
Liu et al. CAESAR SenSys'19

Approach 1: Backpropagate Across the Neurosymbolic Model



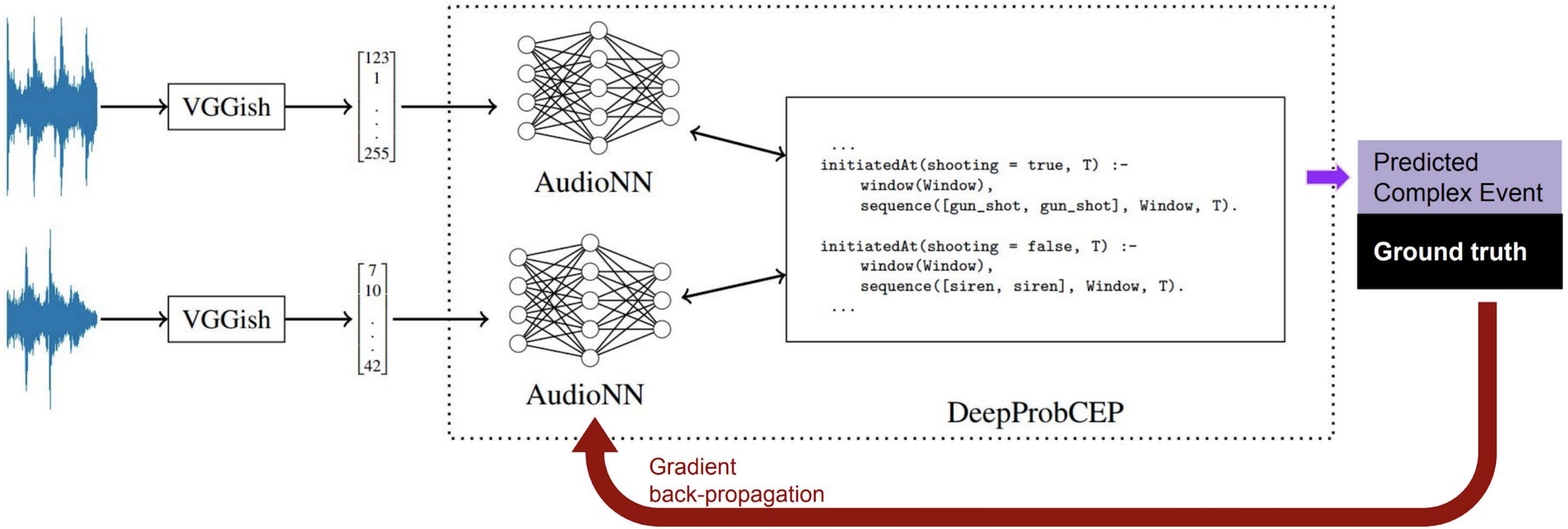
Propagate CE annotation directly to the event level? Difficult!

- Logic rules of CE can be arbitrarily complex
- Relevant events sparsely distributed in the event streams

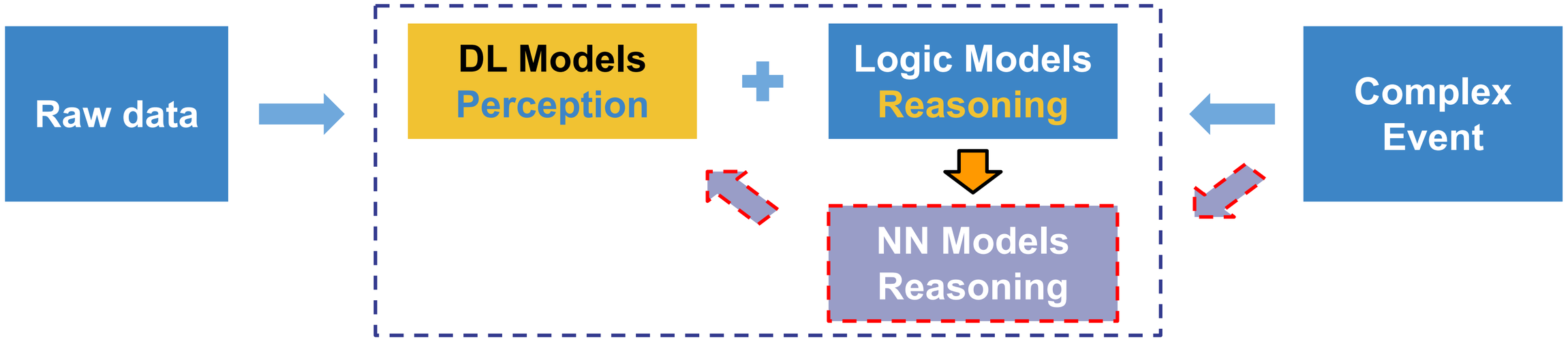


DeepProbCEP: Exploits Logic that is Differentiable

- Augments DeepProbLog to support event processing in temporal dimension
- Trains the neural networks as part of the system in an end-to-end manner



Approach 2: Neural Proxy of the Symbolic Part



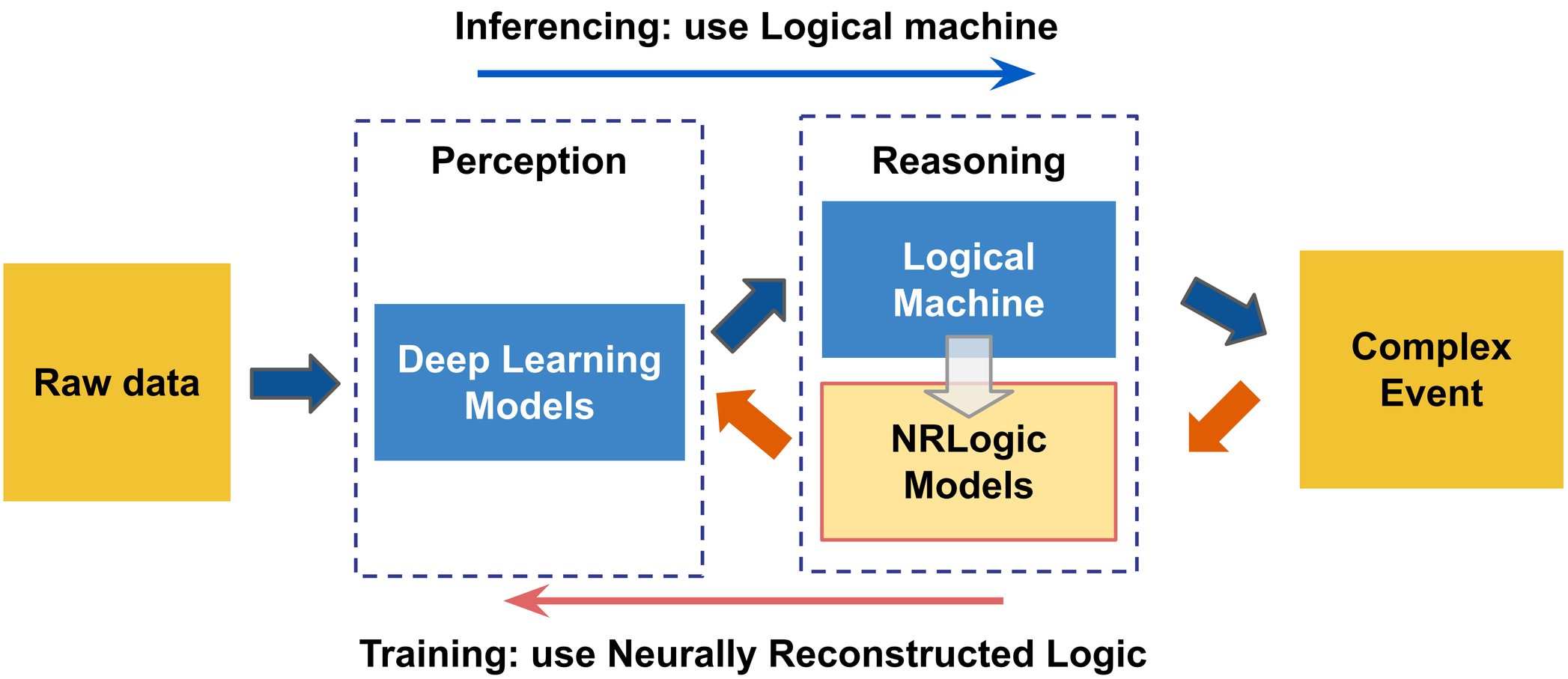
Propagate the gradient to the DL models?

- Make the logic model differentiable.
- Easy if we have DL model on logical layer

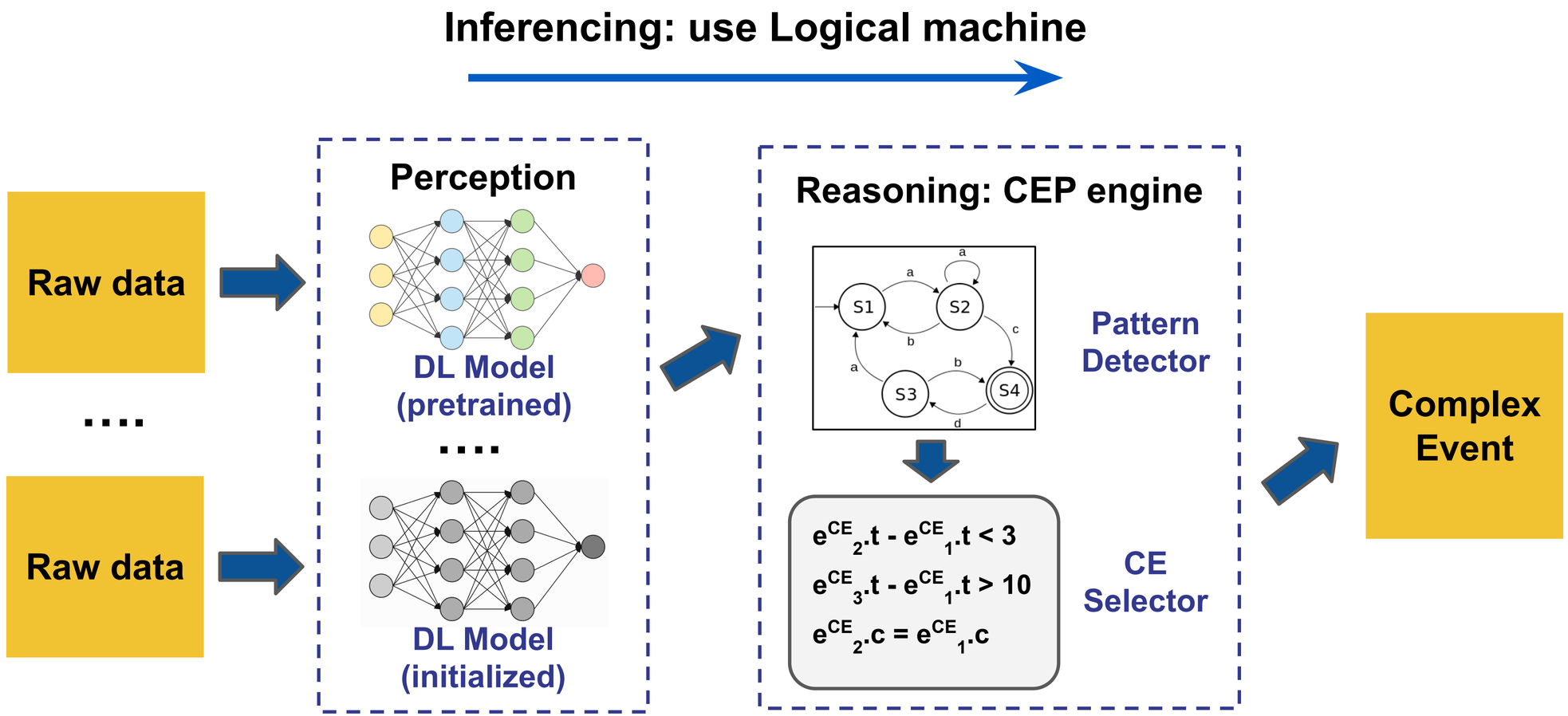
Approximate the Logical Models using Neural Networks?

- Logical models transfer human knowledge to Neural Network models.

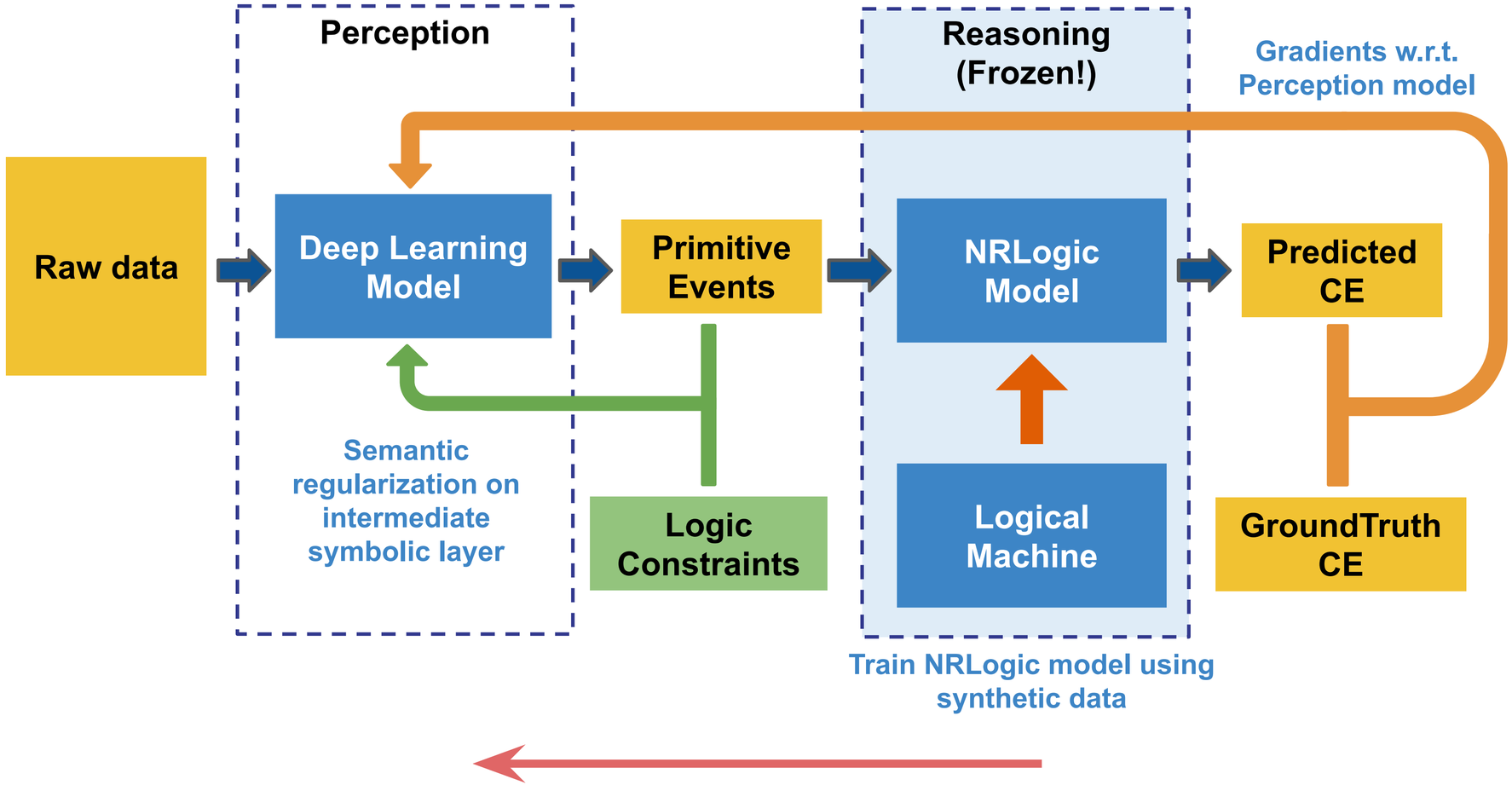
The Neuroplex System



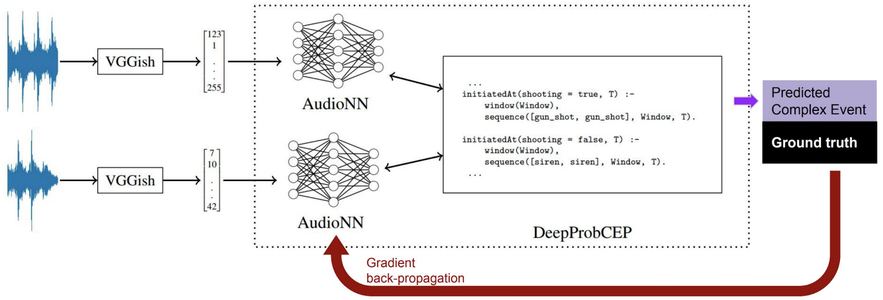
The Neuroplex System: Inferencing



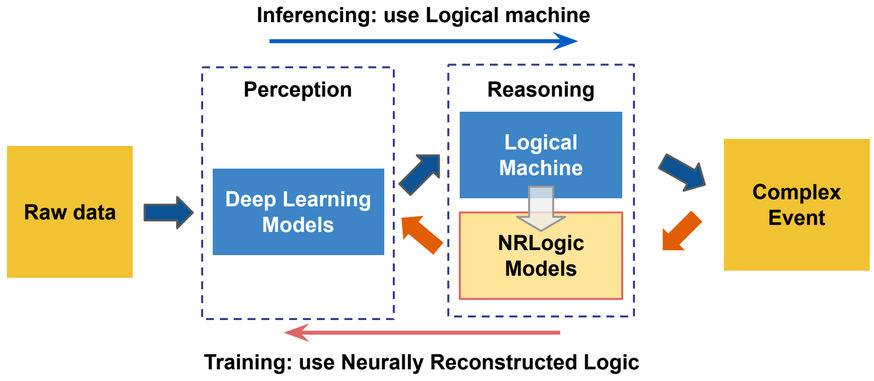
The Neuroplex System: Training



Comparing End-to-End and Neural Proxy Approaches



vs.



	Inference Time (single sequence)	Training Time	CE Detection Accuracy	Perception Accuracy
DeepProbCEP	100ms	1200ms	N/A	N/A
Neuroplex (cpu)	**5.4ms**	**18ms**	**99.39%** (Train for 20 epochs)	**98.87%**
Neuroplex (gpu)	11.54ms	31ms		
Neuroplex (cpu)	4.6s / 1280 = 3.6ms	15s / 256 = 58.6ms		
Neuroplex (gpu)	**0.96s / 1280 = 0.75ms**	**3.33s / 256 = 13.0ms**		

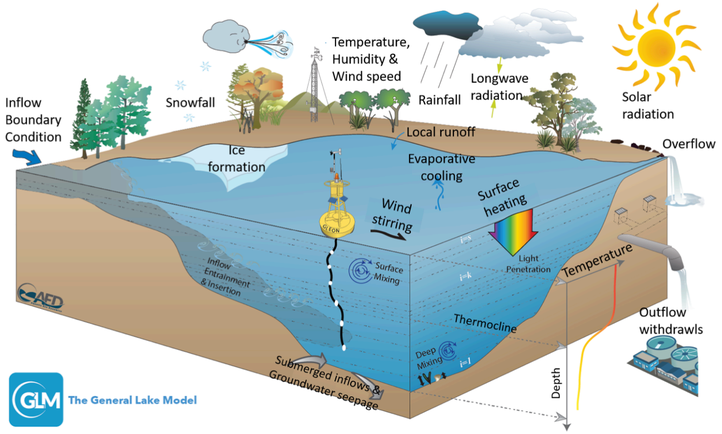
Comparison:

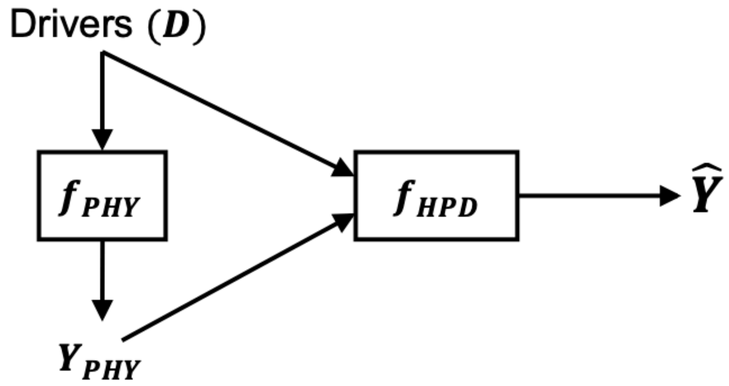
- DeepProbLog instance is around **three orders of magnitude slower** than Neuroplex.
- Probabilistic logic programming makes it quite inefficient in terms of training time.
- DeepProbCEP has a human-understandable and easily manipulable logical regularisation.

Main Ways of Composing “Neural” and “Symbolic”

- Symbolic programs used to compute variables which are then processed by a neural network
- Differentiable symbolic layer in a larger end-to-end trainable neural network
- Neural network called as a function by an outer symbolic program

Example: Physics-Guided Neural Networks (PGNN)





Data science model:

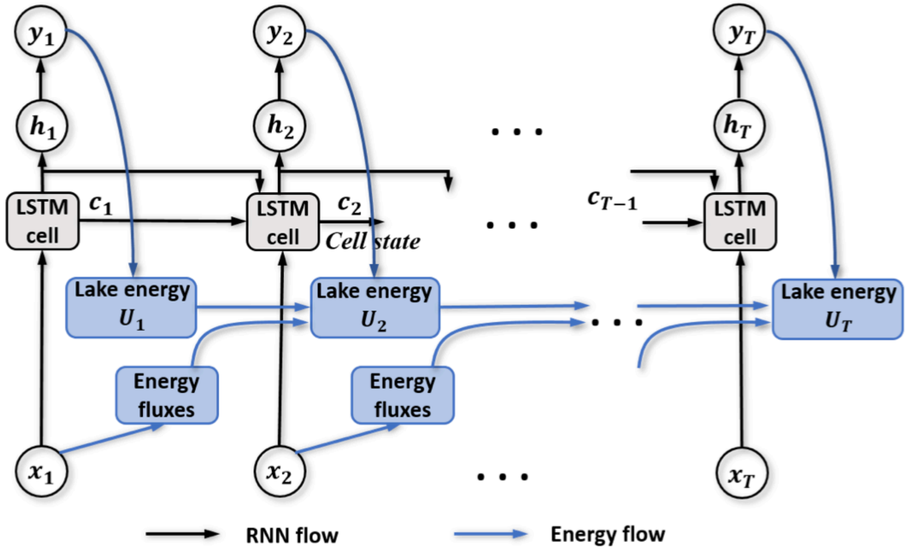
$$f_{NN} : \mathbf{D} \rightarrow Y$$

Physics-based model:

$$f_{PHY} : \mathbf{D} \rightarrow Y$$

Hybrid physics-data model:

$$f_{HPD} : \mathbf{X} = [\mathbf{D}, Y_{PHY}] \rightarrow Y$$



Physics-based loss function:

$$\arg\min_f \quad \underbrace{Loss(\hat{Y}, Y)}_{\text{Empirical Error}} + \underbrace{\lambda\, R(f)}_{\text{Structural Error}} + \underbrace{\lambda_{PHY}\; Loss.PHY(\hat{Y})}_{\text{Physical Inconsistency}},$$

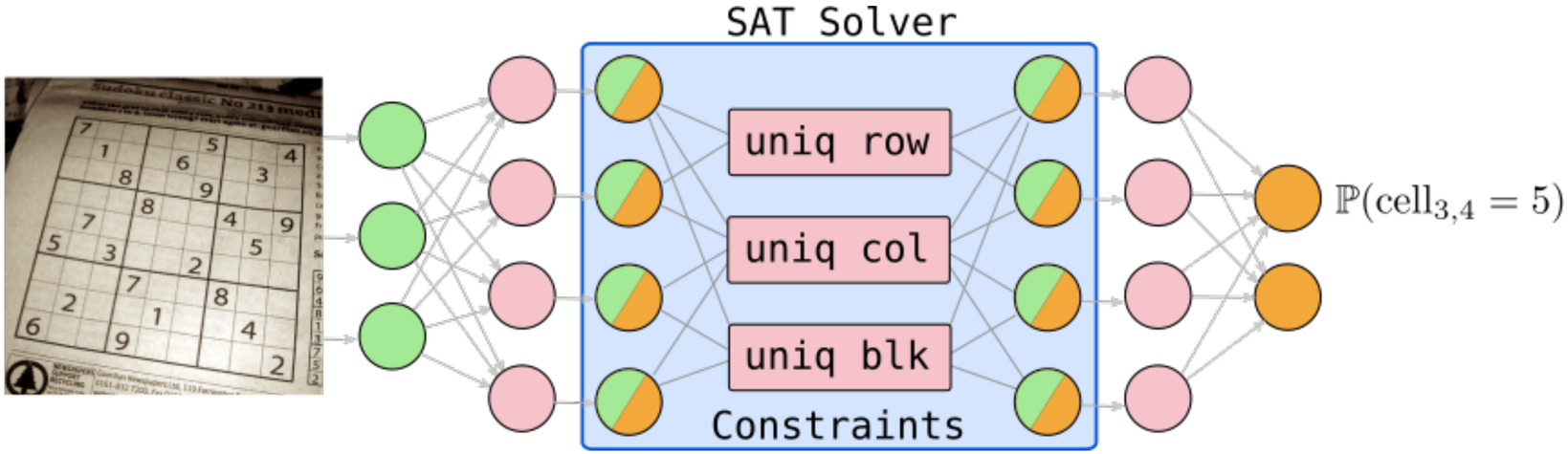
https://arxiv.org/pdf/2001.11086.pdf

Main Ways of Composing “Neural” and “Symbolic”

- Symbolic programs used to compute variables which are then processed by a neural network

- Differentiable symbolic program layer in a larger end-to-end trainable neural network

Example: Convex Optimization and SAT Solver as a Layer



Deep Learning

No constraints on output

Differentiable

Solved via gradient optimizers

+

Logical Inference

Rich constraints on output

Discrete input/output

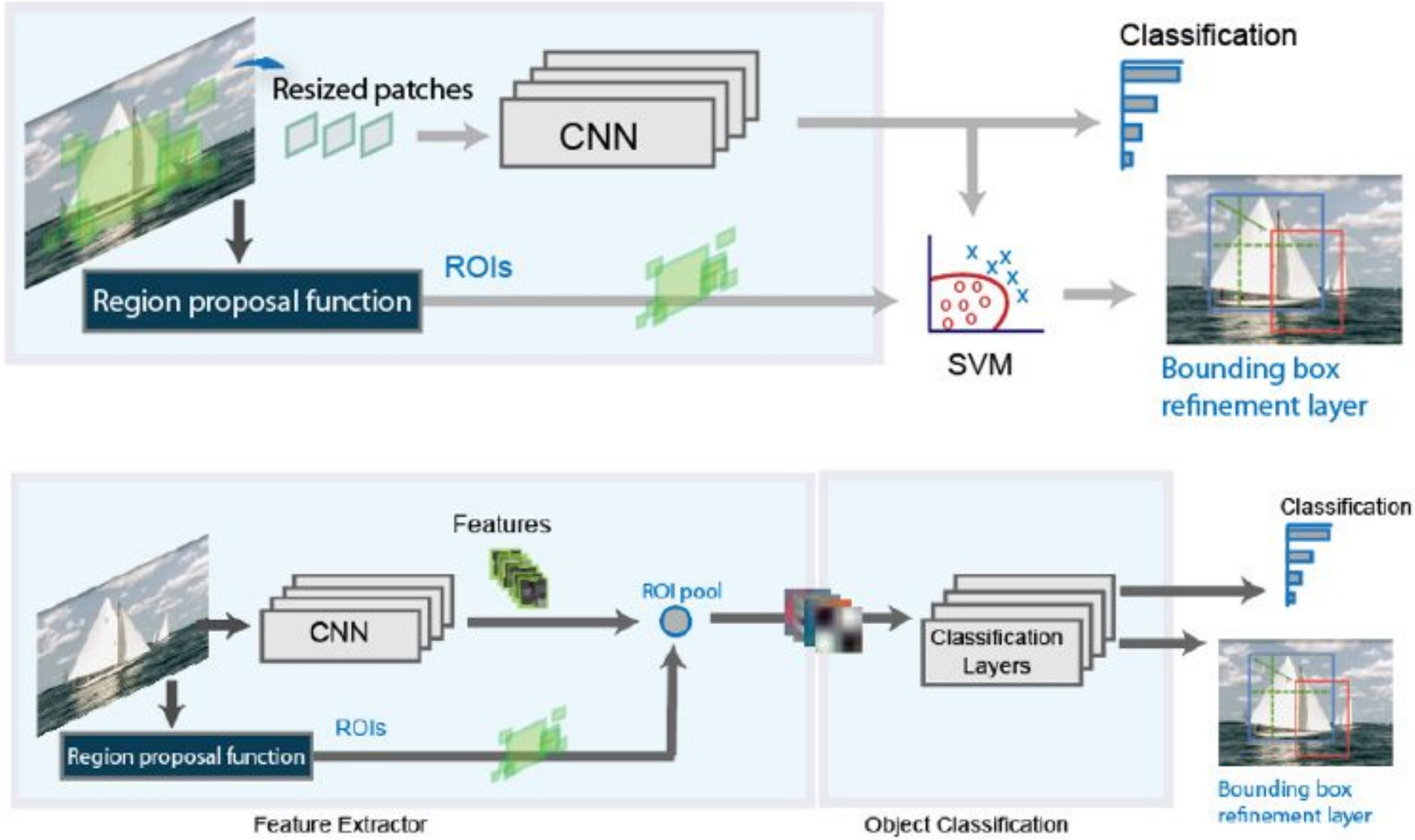
Solved via tree search

- SATNet introduces a layer that enables end-to-end learning of both the constraints and solutions within deep networks where neurons have logical constraints

Main Ways of Composing “Neural” and “Symbolic”

- Symbolic programs used to compute variables which are then processed by a neural network
- Differentiable symbolic program layer in a larger end-to-end trainable neural network
- Neural network called as a function by an outer symbolic program

Example: Object Detection



https://www.mathworks.com/discovery/object-detection.html

The End