# Neuroplex: Learning to Detect Complex Events in Sensor Networks through Knowledge Injection

Tianwei Xing[1], Luis Garcia[2], Marc Roig Vilamala[3], Federico Cerutti[4], Lance Kaplan[5], Alun Preece[6], and Mani Srivastava[7]

[1]University of California, Los Angeles
[2]University of Southern California ISI
[3]Cardiff University
[4]University of Brescia
[5]CCDC Army Research Lab, Adelphi
[6]Cardiff University
[7]University of California, Los Angeles

**Abstract**

Despite the remarkable success in a broad set of sensing applications, state-of-the-art deep learning techniques struggle with complex reasoning tasks across a distributed set of sensors. Unlike recognizing transient complex activities (e.g., human activities such as walking or running) from a single sensor, detecting more complex events with larger spatial and temporal dependencies across multiple sensors is extremely difficult, e.g., utilizing a hospital's sensor network to detect whether a nurse is following a sanitary protocol as they traverse from patient to patient. Training a more complicated model requires a larger amount of data–which is unrealistic considering complex events rarely happen in nature. Moreover, neural networks struggle with reasoning about serial, aperiodic events separated by large quantities in the spatial-temporal dimensions.

We propose NEUROPLEX, a neural-symbolic framework that learns to perform complex reasoning on raw sensory data with the help of high-level, injected human knowledge. NEUROPLEX decomposes the entire complex learning space into explicit perception and reasoning layers, i.e., by maintaining neural networks to perform low-level perception tasks and neurally reconstructed reasoning models to perform high-level, explainable reasoning. After training the neurally reconstructed reasoning model using human knowledge, NEUROPLEX allows effective end-to-end training of perception models with an additional semantic loss using only sparse, high-level annotations. Our experiments and evaluation show that NEUROPLEX is capable of learning to efficiently and effectively detect complex events–which cannot be handled by state-of-the-art neural network models. During the training, NEUROPLEX not only reduces data annotation

requirements by 100x, but also significantly speeds up the learning process for complex event detection by 4x.

# 1 Introduction

Temporal and spatial relationships are natural elements that occur in human learning tasks such as language, motion, and vision [18]. During the past decade, deep learning researchers have achieved great success simulating human cognitive processes using sensors for associated tasks such as object detection, activity classification, and autonomous driving. While demonstrating excellent performance on different sensing tasks, deep neural networks rely on large volumes of training data. Complex spatial-temporal classification tasks suffer from data scarcity, thus making it challenging to design robust learning frameworks. Further, state-of-the-art frameworks are currently limited to a few thousand-time steps while sacrificing interpretability[7]. For instance, deep learning models can currently be trained to detect whether a nurse is sitting or standing in a video stream. However, it would be intractable to train a model that can detect a nurse who does not follow a sanitary protocol before moving between patients—a task that spans arbitrarily long periods, an arbitrary combination of spaces, and time-dependent constraints, and for which a large dataset would not be available. Intuitively, if a set of deep learning classifiers are utilized to detect the simpler activities that happen on the order of seconds, e.g., the nurse entering a room or washing his hands, a human would be able to identify a logical sequence of events that correspond to a violation.

In this paper, we introduce NEUROPLEX, a neural-symbolic framework that splits the entire learning space into a *perception* space and a *reasoning* space. For the perception task, it trains deep neural networks to get low-level symbolic concepts; while accepting the injection of symbolic human knowledge for high-level reasoning. The entire model can be trained end-to-end with only high-level annotations, which also alleviates the burden of data annotation. In comparison to prior work that combines symbolic and neural reasoning [22, 34, 36], we focus on injecting symbolic knowledge expressed as finite state machines and logical rules, which capture the complex temporal and spatial dependencies for these complex tasks. Given the hierarchical reasoning approach, we formulate the problem as a complex event processing (CEP) problem as was done in previous works [42]. In this work, we provide end-to-end training as opposed to just the forward path.

To summarize, NEUROPLEX makes these important contributions:

- NEUROPLEX adapts neural-symbolic approaches that combine deep learning perception with semantic logical models to enable end-to-end learning for detecting complex events from raw sensor data streams. To enable learning, NEUROPLEX leverages a differentiable Neurally Reconstructed Logic (NRLogic) model, which is a neural network trained by a logical machine through a knowledge distillation[11].

- NEUROPLEX is able to train itself using only data with sparse, high-level, complex event labels. By propagating gradients through the differentiable NRLogic model, *perception modules* receive feedback from complex events labels and are trained accordingly. The training could happen both at the initialization stage, where a perception module is untrained, as well as during the fine-tuning stage, where a perception module is a pre-trained off-the-shelf model and needs to be fine-tuned to a specific environment. NEUROPLEX also applies a semantic loss on the intermediate symbolic layer, forcing the perception module to generate a reasonable symbolic output to improve the learning performance.

- We evaluate the NEUROPLEX framework on three complex event datasets, and compare its performance with state-of-the-art neural network models and neural-symbolic methods. Results show that guided by injected human knowledge, NEUROPLEX can effectively and efficiently learn to detect complex events that other approaches cannot handle. It not only improves the training speed by 4X times, but can also achieve superb performance while using only 2 orders of magnitude less training data.

This paper is organized as follow: In section 2 we formally define what complex event detection is, and discuss the challenge of learning in complex event detection; Section 3 formulates the problem we are solving and gives an overview of NEUROPLEX system; Section 4 details the inference and training pipeline of NEUROPLEX, and describes how models are trained in both perception and reasoning module[1]; In section 5, we perform a set of experiments on three different complex event dataset, and demonstrate the effectiveness and efficiency of NEUROPLEX, and discuss the current limitations and future directions in section 6; Section 7 lists a set of related works, and section 8 concludes the paper.

# 2 Background and Motivation

NEUROPLEX aims to detect complex events with intricate spatial and temporal dependencies across multiple sensors. In this section, we formally define what complex events are and discuss the motivation of NEUROPLEX.

## 2.1 Simple and Complex Events

A *simple event* is an event that happens over a short period of time, which usually can be succinctly described by a single word label or a short phrase [22], and can be captured by a single sensor. Modern deep learning models have shown remarkable performance in detecting simple events. For example, a simple event can be the occurrence of a particular object in a given image (e.g., a truck or a suitcase), an audio segment in a waveform (e.g., a siren or a gunshot), or a specific action/activity performed by the subject in the camera

[1]The data and codes of NEUROPLEX are available at https://github.com/NESL/Neuroplex

feed or IMU sensor trace (e.g., walking or opening a door). Simple events are the atoms composing *complex events*.

**Definition 1** (**Complex event**). *In this paper, a complex event is strictly defined as a particular pattern or sequence of* $\geq 2$ *instances of simple events that have spatial and/or temporal dependencies.*

Under this definition, a *complex event* must be composed of multiple *simple events* that may evolve over long periods of time in different spatial contexts with various participants. One important distinction between a *complex event* and a *complex activity* is that, although both of them can be decomposed to a set of atomic events, the composing events of a *complex event* may or may NOT be consecutive in time and space. For example, the "long-jump" sporting event is considered a complex activity that consists of five sub-activities: "standing still", "running", "jumping", "landing", and "standing up". These activities are consecutive and can be captured from a single sensor. An example of *complex event* is a sanitary protocol violation event in a hospital scenario: a nurse could violate the sanitary protocol if she or he processes one patient, and then processes another patient without proper sanitation. In this case, the related events (processing patients and hand-washing) need to be detected and analyzed over a broad temporal and spatial range.

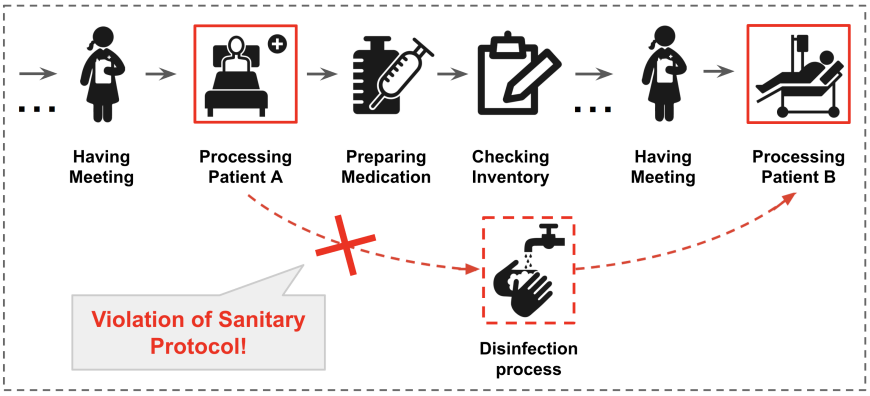


Figure 1: Complex Event: Violation of sanitary protocol.

As illustrated in Figure 1, these essential events are separated by intermediate, unrelated events such as medication preparation and inventory checking. Although deep learning models can excel at identifying each composing event, developing a robust inferencing mechanism for the associated complex event is challenging.

**Complex Event Detection Systems** Although the complex event detection task is challenging, it is of great importance and spans numerous applications in distributed sensor networks, e.g., detecting suspicious activities for security surveillance or the aforementioned protocol violations in hospitals. Recent works have proposed to use neural-symbolic systems [42, 22, 34] to detect complex events. With a hybrid approach, pre-trained neural network models are applied

to recognize simple events from multimodal sensory data, and rule-based logic models are used to perform high-level reasoning over a sequence of simple events extracted by deep learning models. The logic models are defined via a set of logic rules, specifying the temporal and spatial relationship between simple events. The logic rules are given by users based on human knowledge.

### 2.2 Learning for Complex Event Detection

Although hybrid neural-symbolic systems have shown to work well on complex event detection tasks, the existing works only focus on the inference stage, where it is assumed that pre-trained deep learning models for detecting simple events are available.

However, this assumption is not practical in real scenarios. First, off-the-shelf deep learning models are trained on standard population-scale datasets and may not perform well when deployed without fine-tuning in a personalized environment. Second, the definition of complex events involves a set of simple events, which may not be included in the output directory of pre-trained deep learning models. For example, when detecting an unsanitary nursing event, if the pre-trained activity classification model can only have a label set of ["washing hands", "walking", "running"] without a "processing patient" label, then the complex event detection system would not work unless a new activity classification model supporting "processing patient" is trained. Third, the performance of neural-symbolic systems relies primarily on the performance of deep learning models for simple events. Since the complex event definition fixes the logic, the final detection result can be totally different if the deep learning model outputs change. Also, the system's performance degrades when the accuracy of the detection models decreases[22].

Thus, enabling training in complex event detection systems is of great importance. A training pipeline allows the system to fine-tune itself during runtime when deployed in a new environment. Further, the system can train newly added, untrained deep learning models from scratch using only high-level annotations, i.e., it is possible to train an activity classification model supporting a "processing patient" activity only with the high-level "unsanitary nursing event" annotations. High-level annotation significantly reduces the simple event labeling effort for users.

## 3 Problem Formulation and Overview

In this section, we first formalize the complex event detection problem, and then describe NEUROPLEX's design overview.

### 3.1 Complex Event Problem Formulation

Without loss of generality, we consider a sensor streaming data continuously.[2] At every time step $i$, the sensor generates a triple of raw data information, $d_i = \{x_i, t_i, c_i\}$, where: $x_i$ is raw data of a certain modality captured by the sensor; $t_i$ is the timestamp of the raw data sample; and $c_i$ corresponds to any domain-specific metadata or attributes of the data sample. The attributes can represent any physical features or context related to the piece of data, e.g. the location of the sensor. Depending on the usage scenario, a machine learning model $f_\theta$ maps raw information $x_i$ to a set of symbolic classes or values $f_\theta(x_i)$, which we refer to as a simple event, $w_i$, and where $\theta$ is the parameter of model $f$. A primitive event denoted as $e_i = \{w_i, t_i, c_i\}$, is the abstraction of the raw data sample $d_i$.

The problem that NEUROPLEX addresses is learning and reasoning about a data stream of length K. We assume that the length is either user-defined or determined by the limitations of the implemented system. At any given sample point $i \geq$ K, NEUROPLEX should be able to determine if a complex event exists for the previous K samples, i.e., $\{e_{i-K+1},...,e_{i-1},\ e_i\}$. Further, we assume that a set of logical rules, $\phi$, can be utilized from prior knowledge, e.g., human knowledge, to describe the logical dependencies between primitive events to compose complex events. The logical rules, $\phi$, are comprised of both a pattern definition, $\omega$, as well as any additional logical constraints, $\psi$, for the primitive events, i.e., $\phi = \{\omega,\psi\}$. Formally, we define a complex event as a sequence or pattern of primitive events $\omega = e_1^{CE}, e_2^{CE}, ...e_k^{CE}$ that may have an additional set of logical constraints, $\psi$. The primitive events $e_i^{CE}, (i = 1, 2, ..., k)$ correspond to the composite primitive events for a complex event, and $k$ is the number of primitive events in this complex event. Each rule in $\psi$ specifies a logical relationship between primitive events composing the complex event. For a complex event to be detected, all of the logical constraints in $\psi$ must be satisfied. Based on this definition, it is possible to have different complex events happening at the same time when they have the same terminal primitive event, i.e., $e_{k_1}^{CE_1} = e_{k_2}^{CE_2}$. It is also possible to have multiple complex events of the same type happening at the same time if more than one set of satisfying primitive events are detected within the time window K.

At training time, the goal of NEUROPLEX is to train a perception model $f$ for each raw sensor data stream, i.e., to learn the optimal set of parameters $\theta$ using only streams of raw sensory data $d_i = \{x_i, t_i, c_i\}$ and a complex event label.

### 3.2 Neuroplex Overview

NEUROPLEX uses a hybrid neural-symbolic framework to detect complex events. The deep learning models for detecting simple events compose the *perception module*, and a high-level, human-input logical reasoning machine is referred to

[2] Although we are discussing learning of NEUROPLEX in a single sensor scenario, it can be generalized to cases where there exist multiple sensors with different modalities.

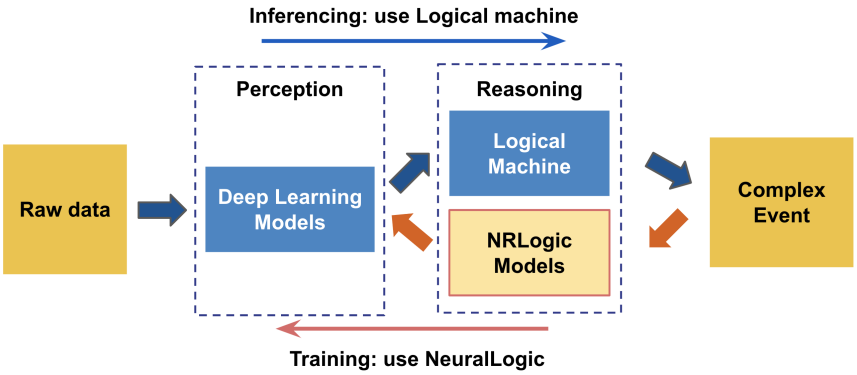


Figure 2: NEUROPLEX system with Perception module and dual form Reasoning module.

as the *reasoning module*, as illustrated in figure 2. Given human knowledge injected into the system in the form of logic rules defining complex events, NEUROPLEX enables both a forward inference path and a backward training path. In order to support training, NEUROPLEX tackles the challenge of how to make the reasoning module differentiable while maintaining its function, and how to perform effective training over perception module with only high-level complex event annotations. The details of our methodology are described in the next section.

# 4 Neuroplex Design

In this section we detail the design of the NEUROPLEX framework. We first discuss how NEUROPLEX is initialized before describing how the framework is trained.

## 4.1 Neural-Symbolic Initialization of Neuroplex

Before NEUROPLEX can be trained, we first need to provide a mechanism to initialize the learning framework. The initialized structure *without training* would be semantically equivalent to an inference-only, hybrid neural-symbolic framework such as DeepCEP [42]–depicted in Figure 3.

### 4.1.1 Reasoning Module Initialization

The reasoning module is initialized by a logical machine in the form of a complex event processing (CEP) engine, which takes user-defined complex event patterns and generates the corresponding machinery for detection. We previously formulated a complex event as a pattern or sequence, $\omega$, of primitive events, together with some logical constraints, $\psi$, that impose fine-grained limits on temporal and event attributes. Thus, we decompose the set of logical rules $\phi = \{\omega, \psi\}$ to a finite-state-machine (FSM) model $\omega$ and additional logical constraints $\psi$. The FSM model $\omega$ describes the temporal ordering of primitive events of a complex

event, and the logical constraints $\psi$ describe the arithmetic logical relations that are not covered by $\omega$. For example, for the complex event of a nurse violating a sanitary protocol (Figure 1), the FSM pattern detector, $\omega$, would model the pattern of the nurse's activities to detect two instances of a nurse processing a patient that was not separated by a disinfection process activity. The logical constraints, $\psi$, would correspond to checking whether the pattern happened within a particular time frame as well as to ensure the primitive events correspond to the appropriate sensors (e.g., the two instances of a patient being processed happened at different locations).

To formalize these definitions, we present a complex event grammar in Backus-Naur Form (BNF) that enables the injection of human knowledge into NEUROPLEX.

**CEP reasoning module grammar.** The Backus-Naur Form grammar for NEUROPLEX's CEP engine is defined in Figure 4 and builds upon the grammar of a previous work [42]. We utilize the <input-title> and <complex-event-title> clauses to label the source of a primitive event stream and the associated complex event, respectively. The <format-pattern> clause defines the aforementioned FSM by describing what the candidate primitive event patterns are. The <constraint> clause defines the logical constraints $\psi$.

Unlike the prior work [42], our grammar's semantics carry a significantly different meaning at training time. The goal of the NEUROPLEX design is to utilize these definitions to bootstrap the training for the reasoning layer of a CEP engine. Without training enabled, the framework would be semantically equivalent to the previous hybrid neural-symbolic inferencing frameworks [42, 24]–as shown in Figure 3. In this context, the human-defined logical machine will be utilized as a ground truth at training time. Thus, we define the semantics for how a defined pattern detector processes primitive events as well as how logical constraints are enforced.

**Semantics of primitive event pattern detection.** The primitive event pattern specified by the <format-pattern> clause is a derivative of the CEP language for SASE [37]. The formats supported by our CEP grammar are:

$$SEQ(A_1, A_2, ..., A_n)(t) \equiv \{A_1 A_2 ... A_n\}$$
$$PATTERN(A_1, A_k, ..., A_n)(t) \equiv \{A_1(.*)A_k(.*)...A_n\}$$

Where the right-hand side of the equations are the associated regular expressions. The $SEQ$ format represents a sequence of consecutive primitive events that occur in a strict order, while the $PATTERN$ format represents a sequence of nonconsecutive primitive events. At runtime, the defined logical machine will utilize finite state machines (FSMs) to represent each regular expression and maintain the current state of the pattern detector.

When NEUROPLEX initializes the logical machine, the CEP engine first reads a CE definition and creates an FSM model. It then creates and initializes both the event buffer and the event state stack with a size K. Currently, the size number K is a fixed number, where the memory only stores the latest K

events with a sliding window for each subsequent event.[3] When a new event arrives, the CEP engine first updates the event buffer by pushing in the new event information and popping out the oldest event. It then updates the event state stack by removing the active states associated with the popped event and updating all the dependencies. If the final state is activated, it means a complex pattern is detected and the CEP engine will output the sequence of events that triggered the final state. This sequence of events will then be fed into the *selector model* to be checked against the associated logical constraints.

**Semantics of logical constraints.** As discussed, logical predicates can be defined to *filter* or *select* candidate complex events as shown in Figure 3. Event attribute constraints are used to describe the spatial dependencies as well as additional temporal constraints of complex events. They are expressed as a set of logical predicates in the CEP language. NEUROPLEX is intended to support any logic language that can express spatial-temporal properties and has an associated theorem prover. Currently, NEUROPLEX utilizes ProbLog [2] to express the <logic-expression> clause. This choice is due to the fact that deep learning typically outputs a probabilistic result for a primitive event. The reasoning framework aims to propagate the associated probability of a detected primitive event to the probability of a composed complex event. The overall constraint is a Boolean combination (using logical connectives $\vee$ and $\wedge$) of the associated predicates. The complex event is valid only when both the *combination format* and *attribute constraints* are satisfied.

When the logical machine is initialized, the CEP Engine creates the selector model in ProbLog based on the user's complex event definition. The associated parser identifies which events to be taken into account as well as the constraints to be applied to those events. No code transformation is needed as the user defines the logical constraints in ProbLog. At run time, when a complex pattern is detected, the list of events that can form the complex event is passed to ProbLog to check if all of the constraints are satisfied and to calculate the probability of the event happening.

#### 4.1.2 Perception Module Initialization

As previously discussed, the perception module should have a set of deep learning models that can abstract the raw data for each sensor in the network. The event symbols that are used by the human to define complex event patterns and constraints should be a subset of the label set for the associated perception module, i.e., users can only define reasoning rules based on event labels generated by the deep learning models. We assume that each provided model is initialized either with random weights or with pre-trained weights, i.e., NEUROPLEX can be initialized with off-the-shelf models trained on population-scale data–as depicted in Figure 3. However, the pre-trained weights are not necessary for NEUROPLEX's training process.

[3] It is possible to have a memory with a variable size for a fixed time length.

## 4.2 Neuroplex Training Framework Design

The training pipeline for NEUROPLEX decomposes the original end-to-end learning problem into two sub-problems of perception and reasoning. NEUROPLEX can learn each part separately with the aid of injected semantic knowledge to significantly reduce the learning space.

When the logic rules of complex events are known, we need to train the perception module to connect the path between raw data and complex events. However, because logic is not differentiable, we first propose a deep learning model called NRLogic, which approximates the function of the original logical machine in the reasoning module. Depending on the application, this NRLogic model can be trained using knowledge distillation [11] with only synthetic data labeled by the logical machine. During this process, only the reasoning model is updated. Once trained, the reasoning module has a dual form of representation: a logical representation and a neural representation. This enables the training of the perception module to be performed in a supervised manner. With the annotation of complex events, we freeze the parameters of the NRLogic model and calculate the gradient of the loss with respect to the deep learning models in the perception module. Additionally, logical constraints can be added to the intermediate symbolic layer between perception and reasoning module, imposing another regularization term for training. In this phase, the reasoning module is only used to propagate the gradients, and its parameters are kept intact to preserve its functionality. Figure 5 shows the training pipeline of NEUROPLEX's perception module.

**Training of the NRLogic network.** In NEUROPLEX, the intermediate layer between the perception module and the reasoning module is symbolic. For example, if we want to detect a set of composed complex activities, then the perception module could be an activity classifier, and the reasoning module is the logical machine expressing the pattern and constraints between atomic activities. The corresponding primitive event in the intermediate layer is a classified activity result with the associated timestamp and attributes, $e_t = \{f_\theta(x_i), t_i, c_i\}$. The reasoning module here has both structured input space and structured output space, so we can easily generate data sequence samples, and use logical machine $\phi$ to get the ground truth annotations $y$.

Because the logical machine $\phi$ used in complex event detection contains arithmetic logical constraints $\psi$ and finite state machine $\omega$, we use a recurrent neural network structure $p(E)$ to mimic the reasoning module. The input $E_i = \{e_{i-K+1}, .., e_{i-1}, e_i\}$ denotes all the primitive events happening in the past time window K. The training procedure of the NRLogic model can be regarded as a knowledge distillation process [11]. The logical machine is the teacher—which provides ground truth values to the generated primitive events, and the primitive event sequence and label pairs $\{E_i, y_i\}$ are then used to train the student NRLogic network.

Since the ground truth annotation $y_i$ describes instances of multiple complex events, it is possible to have different complex events occurring multiple times in a given time window. Therefore, we formulate this problem as a multi-label

regression problem. The annotation $y_i$ and prediction $p(E_i)$ would be an $m$-dimensional vector, where $m$ is the number of complex events we are detecting. Each entry $y_i^j$ represents the number of complex events $j$ happening at the current time sample $i$.

The NRLogic model is trained using a number of synthetic data as a preparation step for the perception module training. Since the dimension of the input primitive event data is not large, the RNN model employed here can be relatively simple, and thus, the training process would not introduce significant overhead. We keep training the NRLogic until it converges.

**Training of the perception module.** Once the NRLogic model $p$ is well-trained, we can integrate it as a layer in the training pipeline,i.e., we concatenate it with the perception module we need to train. The new integrated model still complies with the structure of NEUROPLEX, where the reasoning module follows the perception module, but the reasoning module is expressed in its neural network form. In this way, we successfully build a differentiable path between raw data and complex event labels.

Since the NRLogic network is pre-trained to mimic the logical machine, we freeze its weights to preserve its functionality. We then calculate the gradient of the complex event prediction loss with respect to the perception model only, $\partial L(y_t, p(E_t))/\partial\theta$, and use this gradient to update the model. The loss $L$ for the regression task is the mean squared error:

$$L_{MSE} = \frac{1}{N}\sum_{t=1}^{N}(p(E_t) - y_t)^2 \tag{1}$$

Because of the frozen NRLogic, the perception model is forced to learn to predict the corresponding event types. After the training finishes, the perception model ideally performs as if trained with fully-annotated event-level data.

**Semantic Loss on an Intermediate Layer.** One of the most significant features of NEUROPLEX is that it has a symbolic intermediate layer. Since the primitive event $e_i$ in the intermediate layer contains the event type information $w_i$, which is usually expressed in the form of a softmax score, we can impose an additional semantic loss on it to further facilitate training.

We use the idea from [45]: in a multi-class classification task, a well-trained model should give output with exactly one of the classes being true, and the others being false. With this idea, a semantic loss function is introduced to force the mass of the softmax vector to accumulate for a single class, i.e.,

$$L_{Semantic} = -\log\sum_{i=1}^{m} p_i \prod_{j\neq i}(1 - p_j) \tag{2}$$

where $m$ refers to the number of classes in a softmax vector. The intuition here is to minimize the negative log probability of generating a state that satisfies the logical constraints with sampling probabilities equal to the softmax values. When the probabilities of classes are evenly distributed, the loss value would be

a large value close to 1, and if only one of the classes has a probability of 1 and the rest being 0, the loss value is equal to 0.
Therefore, training the perception network entails optimizing a combined loss:

$$L = \lambda \times L_{Semantic} + (1 - \lambda) \times L_{MSE} \tag{3}$$

and the gradient used to update the network becomes:

$$\frac{(1 - \lambda) \cdot \partial L_{MSE}(y_t, p(E_t)) + \lambda \cdot \partial L_{semantic}(f_\theta(X_t))}{\partial \theta}$$

The $\lambda$ is the hyper-parameter that controls the strength of semantic regularization.

# 5 Evaluation

In this section, we empirically evaluate the proposed training method for NEUROPLEX on three complex event datasets. The goal is to evaluate whether NEUROPLEX is capable of learning and detecting complex events, and how it performs compared with state-of-the-art deep learning methods and neuro-symbolic methods. We further perform an ablation study to evaluate how it improves training by adding a semantic loss on the intermediate symbolic layer.

NEUROPLEX is implemented using Tensorflow and Keras frameworks, and evaluated on a desktop machine with an Nvidia RTX Titan GPU. We first conduct an evaluation on a synthesized MNIST sequence data to thoroughly analyze the performance of NEUROPLEX's method, while comparing it with a set of strong baselines. Then we test NEUROPLEX on a complex audio event dataset and a complex nursing event dataset, both of which are constructed using real sensory data.

## 5.1 MNIST Sequence Complex Events

To explicitly control the complexity of the logic in complex events, we synthesize our dataset using MNIST [20] digit images which we refer to as the MNIST Sequence Complex Event dataset. This dataset is generated by randomly creating sequences of MNIST images. Each image in an MNIST sequence is assigned with an increasing timestamp $t$ and an attribute ID $c$. To get the ground-truth label of every MNIST sequence, we randomly generate a set of logical rules $\phi = \{\omega, \psi\}$, and then apply these rules to the sequence to see if the complex events exist. For instance, a complex event in this context, $CE_1$, can be a pattern of $\{e_1^{CE_1}$: "1" $\Rightarrow$ $e_2^{CE_1}$: "3" $\Rightarrow$ $e_3^{CE_1}$: "9"$\}$. The primitive events $e_1^{CE_1}$, $e_2^{CE_1}$, and $e_3^{CE_1}$, must happen in a sequential order with the corresponding digits 1, 3, and 9 as the event types. Additionally, temporal and spatial constraints can be added to the complex event definition, e.g., the set of temporal constraints $\psi \equiv e_2^{CE_1}.time() - e_1^{CE_1}.time() < 10s$ and $e_3^{CE_1}.time() - e_1^{CE_1}.time() > 3s$ and spatial constraints $dist(e_1^{CE_1}.location(), e_2^{CE_1}.location(), e_3^{CE_1}.location()) < 100$ limit the complex events to those that satisfy the generated rules.

We say that a complex event $CE_1$ is happening if a sequence contains the pattern of primitive events $[e_1^{CE_1}, e_2^{CE_1}, e_3^{CE_1}]$—with primitive event $e_3^{CE_1}$ being the last event in the sequence—and if all the logical constraints $\psi$ are satisfied. Each sequence of primitive events is one sample fed to the complex event detection system. The goal of this dataset is to detect the occurrence of different complex events for each raw image sequence.

The complex event detection system has a maximum time window of length K, which means that at most $K$ recent primitive events are considered when making a detection. Thus, we generate the MNIST sequence data with length equals to $K$. Apparently, as window length $K$ increases, the difficulty of complex event detection increases as well. It is because that not only the system requires a larger memory to store and process the latest events, but also the input space and complex events arrangement increase exponentially. In our experiment, we change the size of window length $K$ in different simulations, to test the robustness of NEUROPLEX as the complex event detection task becomes more difficult.

**Experimental setup.** In our experiment, we used a LeNet convolution network [20] as the perception model for digit classification, and an LSTM network as the NRLogic model. Specifically, the LeNet is a CNN architecture with two convolution modules (convolution layer + Relu activation + maxpooling) with a 5-by-5 kernel, followed by two fully-connected layers. It achieves about 99.2% testing accuracy when trained on MNIST training data directly and tested on MNIST testing set.

In the NRLogic model, we use a simple network with one LSTM layer with 64 hidden units plus one fully-connected layer to capture the logic of complex events. Since it is a regression model, the output layer has $m$ nodes and linear activations, where $m$ is the number of complex events types. The optimal $\lambda$ value in function (3) is set to $1e-4$, and we use a grid search to find reasonable parameters. The experiments here use an Adam regularizer with a 0.001 learning rate for training both the NRLogic model and the perception model, and the batch size is set to 256. The NRLogic model is pre-trained on randomly generated data until convergence within 200 epochs.

**Performance measure.** We evaluated the learning performance by looking at the Mean Absolute Error (MAE) of the complex event prediction. However, this term sometimes cannot reflect the model performance directly. We therefore compare the predicted scores with the ground truth values and calculate the prediction accuracy (Acc) by rounding the prediction numbers:

$$Acc = \frac{\sum_{i=1}^{N}\sum_{j=1}^{m}\mathbb{1}(round(p(E_i)^j) == y_i^j)}{m \times N}$$

The superscript $j$ refers to the $j$-th entry of the prediction, and $m$ represents the types of complex events. The accuracy is calculated as the average correct prediction rate for all types of complex events on a testing dataset with $N$ samples.

Additionally, for models with a similar structure as NEUROPLEX, we also

evaluate the performance of the LeNet on the MNIST testing dataset after training to measure the actual learning performance on the perception model.

**Learning comparison with strong baseline methods.** We first test the learning performance of NEUROPLEX on an MNIST sequence dataset, as shown in Table 2, simulation 1. Here we are considering 4 different complex events happening in a window of 10 primitive events. The logic rules of complex events are randomly generated, while each complex event is composed of two to three primitive events, and the average length of the complex events is 2.8. The number of unique primitive event types is 10, which means that these four complex events cover all the 10 digits in MNIST.

We compared NEUROPLEX with mainstream deep learning baselines:

- **(1) CRNN network:** It has exactly the same structure as the model we are using in NEUROPLEX: CNN+LSTM structure, inspired by [30]. The only difference is that in the CRNN model, human knowledge is not injected, so the LSTM layer is not pre-trained.
- **(2) C3D model:** The C3D network has a similar 3D convolutional structure as [21] to capture temporal dependencies between image frames. The total number of parameters is around 4.1 million, which is much more complex compared with our model with 1.3 million parameters.
- **(3) Neuroplex w/o semantic loss:** We also perform an ablation study by removing the semantic loss imposed on the intermediate layer of NEUROPLEX to see the extent to which it helps training.
- **(4) Oracle:** The oracle method uses a neural-symbolic approach with perfectly pre-trained perception models and 100% correct human-defined logical rules. This method only represents the theoretically best performance we can get, since pre-trained networks tuned to specific environments are usually not available because of the heterogeneity of different domains and different task requirements.

Although NEUROPLEX has human knowledge injected and distilled to NRLogic model, the pre-training overhead is pretty small. This is because the NRLogic model is not complex, and it has symbolic input and output space. The model converges to optimum within a few minutes during the pre-training process.

As shown in Figure 6, we train different models on a training dataset of 10K sequences for 200 epochs, and plot the learning curves on 2K validation dataset in terms of MAE. Clearly, NEUROPLEX learns faster than other baselines, and it trains the best final model with a performance close to the oracle approach. The C3D model learns slowly, and it fails to converge to the optimum in 200 epochs. Though having the same structure as NEUROPLEX, the CRNN model struggles to capture the dependencies between raw data and complex events due to the high task complexity. In the ablation study, the model without semantic loss has an inferior performance compared with the original NEUROPLEX, proving that the semantic loss indeed helps training and regularizing the model.

| | Oracle | NEUROPLEX | NEUROPLEX (w/o) | CRNN | C3D |
|---|---|---|---|---|---|
| Perception Acc | 99.19% | **98.87%** | 70.55% | 10.09% | NA |
| Validation MAE | 0.002 | **0.013** | 0.065 | 0.523 | 0.176 |
| Converted Acc | 99.85 | **99.39%** | 96.02% | 69.98% | 88.47% |

Table 1: Performance comparison on MNIST Sequence

We also measure the performance of different methods in Table 1. Clearly, NEUROPLEX could achieve the lowest MAE on the complex event prediction, and the converted accuracy is higher than 99%. The LeNet in the perception module is also well-trained to get an accuracy of 98.87%, which is pretty close to the model trained on the original MNIST training data (99.2%). The CRNN model which has the same network architecture but without human knowledge injected does not capture the complex events well, and the LeNet on the CRNN could not perform image recognition effectively.

Figure 7 shows that, as training progresses, the performance of LeNet is improving with the entire model. Also, since the NRLogic model is frozen and not trainable, it perfectly maintains the functionality of the original logic models. As we expected, the overall complex detection performance is totally dependent on the ability of the perception module.

**Comparison with neural-symbolic methods.** We performed a preliminary investigation using the state-of-the-art complex event detection method described in [35], which is a neuro-symbolic architecture that combines a neural network—which processes raw data—and logic programming—to express the patterns that define a complex event. The system allows for end-to-end learning using DeepProbLog[24]. However, the probabilistic logic programming aspect of this system makes it quite inefficient in terms of training time. In a preliminary investigation in this direction, while NEUROPLEX takes 5.4 milliseconds in training over a CPU—and even less on GPUs—a DeepProbLog instance is around four orders of magnitude slower. We believe that the flexibility of having a human-understandable and easily manipulable logical regularisation will be valuable for articulated complex event detection rules, but that first requires a coordinated effort of the neuro-symbolic community to improve the engineering of DeepProbLog.

**Learning with a limited amount of training data.** To evaluate how NEUROPLEX performs in the scarce data scenario, we synthesize a new complex event setting, as per Table 2, simulation 4. In this dataset, the complexity of a complex event is greatly reduced to ensure that all the baseline models can learn. The length of the event window is 3, and three unique events are considered in 5 different complex events which have an average length of 2. We adjust the number of available training data, from only 10 samples to 21K samples, and train all models for 200 epochs.

As shown in Figure 8, regardless of the training dataset size, the proposed NEUROPLEX method steadily shows the best performance, especially in the case when training data is very limited. Removing the semantic loss on NEUROPLEX would incur a small performance drop and it again proves the benefits of a symbolic layer and the corresponding logical constraints for training. The CRNN trained from scratch cannot learn and reason about complex events effectively

even with 21K data, and the complex C3D model begins to show acceptable performance (greater than 85%) only when the amount of training data is greater than 10K. NEUROPLEX is shown to be robust to the data scarcity problem as it can achieve over 90% accuracy with only 20 data samples.

| | Simulation 1 | Simulation 2 | Simulation 3 | Simulation 4 | Simulation 5 |
|---|---|---|---|---|---|
| Window Length | 10 | 20 | 30 | 3 | 2 |
| Number of Unique Events | 10 | 10 | 10 | 3 | 3 |
| Number of CE | 4 | 4 | 7 | 5 | 4 |
| Avg. CE Length | 2.8 | 2.8 | 3.43 | 2 | 2 |
| NEUROPLEX/ Perception | **99.39%** / 98.87% | **99.56%** / 99.17% | **98.65%** / 98.91% | **100.00%** / 99.84% | 99.98% / 99.78% |
| CRNN model | 69.98% | 7.79% | 1.83% | 86.37% | **99.99%** |
| C3D model | 88.47% | 83.73% | 86.91% | 98.56% | 99.72% |

Table 2: Summary of CE datasets and training performance of different methods. Simulation 1, 2 and 3 are complex event tasks with normal complexity. Simulation 4 and 5 are the simple complex event scenarios that all the systems can train on.

**Performance on different tasks.** To further test whether NEUROPLEX scales well for complex event detection, we conduct a set of different experiments with datasets of various complexity. As shown in Table 2, simulations 1, 2, and 3 are tasks with increasing complexity, and simulations 4 and 5 are two simple tasks that all the testing models are able to learn.

From our formulation, the complexity of complex event detection would increase when the window length increases. A longer time window with more primitive events implies that the input space increases exponentially. The number of unique events and the length of a complex event, on the other hand, control the complexity of the complex event itself.

In simulation 3, the task is detecting 7 complex events composed of 3 or 4 primitive events in a window of 30 primitive events. Even though the CRNN shows poor performance in this setting, NEUROPLEX could still train the perception model well and achieve high accuracy for the complex event detection task. The fifth row of learning performance in Table 2 shows both the converted validation accuracy of NEUROPLEX on the complex detection task as well as the testing accuracy of the perception module. The performance of the C3D model in simulation 3 is better than simulation 2, even though the complexity of the task increases. One possible reason is that in simulation 3, the number of complex events we detect grows to 7, meaning that more feedback information is provided for a single event sequence sample since the annotation increases from 4 to 7.

In the simple task simulation 5, we notice that the CRNN model's performance is unnoticeably better than the proposed NEUROPLEX. This is because both the CRNN baseline and NEUROPLEX use the same network structure, and NEUROPLEX keeps part of its parameters frozen so that NEUROPLEX could be less flexible when finding the optimal solution. Besides, in NEUROPLEX, we decompose the entire learning space into the perception space and the reasoning space and learn them separately. This leads to a simplified problem and a im-

| | Event types | Length | Num |
|---|---|---|---|
| CE 1 | cooking ⇒ eating ⇒ dishwashing | 3 | 1213 |
| CE 2 | social_activity ⇒ cooking ⇒ eating | 3 | 1198 |
| CE 3 | working ⇒ other | 2 | 2898 |
| CE 4 | watching_tv ⇒ vacuum_cleaner | 2 | 2904 |
| CE 5 | absence ⇒ eating | 2 | 2844 |
| CE 6 | dishwashing ⇒ cooking | 2 | 2888 |
| CE 7 | absence ⇒ social_activity | 2 | 2919 |
| Event types: 9 . Avg length: 2.29. Dataset size: 16162 | | | |

Table 3: Summary of Complex Audio Event Dataset

proved learning speed, but the summation of optimalities in two sub-spaces is not necessarily the optimality of the entire space. However, this does not affect the efficacy of the proposed method since NEUROPLEX can get near-optimal performance with great compute, data efficiency, and speed.

## 5.2 Synthetic Complex Audio Events

In this experiment, we show how NEUROPLEX can be applied to real audio event data. We construct a complex audio event dataset by sampling a subset from the DCASE 2018 challenge task 5 [3] and synthesizing complex events. The audio event data has 9 different classes: absence, cooking, dish-washing, eating, other, social activity, vacuum cleaning, watching TV and working. Each data sample is an audio recording of 10 seconds.

When creating the complex audio event data, we define the rules of complex audio events and build our logical machine based on it to provide annotations. The pattern is defined arbitrarily, as specified in Table 3. Some of them are defined based on regular human activities. For example, the complex event No.1 (CE 1) defines a regular dinner activity with a pattern of ["cooking" ⇒ "eating" ⇒ "dish-washing"].

For generating raw complex audio event waveforms, we first create an empty long audio data sample with a length of 100 seconds and then overlay random audio samples selected from the audio dataset. Since the labels of the audio samples are known, we use the logical machine to get the ground-truth complex event annotation for each generated long audio waveform. Table 3 gives a summary of the complex audio event dataset we generate.

With a given large audio file, the system is expected to extract the audio features from raw waveforms and make predictions about the occurrence of complex audio events inside this long period of time. In this experiment, we used the CNN model from DCASE 2018 [4] as our perception module. This CNN model contains two convolution blocks followed by two fully-connected layers with Relu activation. It has 17.8K parameters in total. Batch normalization and dropout are used to add robustness to the model.

The input to the model is the log mel-band energies for audio in a 10-second window. Therefore, to analyze the 100-second-long audio waveform, our system first used a sliding window to extract the mel-band features for every 10 seconds in a non-overlapping manner. The features are extracted from the raw data and

| | NEUROPLEX | NEUROPLEX (w/o) | CRNN | ConvNet |
|---|---|---|---|---|
| Perception Acc | **89.44%** | 84.93% | 0.22% | NA |
| Validation MAE | **0.1015** | 0.1032 | 0.1671 | 0.3884 |
| Converted Acc | **92.53%** | 92.39% | 89.50% | 85.09% |

Table 4: Model performance on complex audio event data

then processed by the perception module to generate primitive events. The NRLogic model is also an LSTM network with one hidden layer analyzing the logic between audio events.

Similar to the MNIST sequence simulation, our first baseline model is also a CRNN network using the exact same structure as NEUROPLEX but without semantic knowledge injected. The second baseline model has a similar structure as the baseline CNN, but with twice the dimensions and a total of 55.8K parameters. We use this model to test if the deep learning model that works well on short time-series data would scale well on a much longer time series. These strong baseline models represent the performance of modern deep learning approaches well.

We train all the models for 200 epochs using the Adam optimizer with a learning rate equal to 0.001 and a batch size of 256. In Figure 9 we can see the learning curves of different methods in terms of validation MAE. Both the CRNN and NEUROPLEX models can capture the complex audio event, but the baseline CNN model does not converge to optimal. The CNN baseline model fails to learn in this task with a longer time series, which shows the limitation of mainstream deep learning models on complex event tasks.

In Table 4, we can see that the NEUROPLEX method could learn complex audio events with the lowest validation MAE and the highest accuracy of 92.53%. Additionally, the perception module performs well (with an accuracy of 89.44%) on the testing audio dataset—given that the same network trained using fully-annotated audio event data can only get an accuracy of 91.14%. The ablation study shows that NEUROPLEX without semantic loss also shows good performance but a little bit inferior to NEUROPLEX, which further proves that the semantic loss helps training.

## 5.3 Complex Nursing Events Detection

The third experiment is conducted on a complex nursing event dataset based on a public dataset from Nursing Activity Recognition Challenge[17]. The dataset contains nurse activity data collected from three sources: Motion Capture, Meditag, and Accelerometer sensors, and it includes six different activities performed by eight subjects (nurses). These are C1: Vital signs measurements, C2: Blood collection, C3: Blood glucose measurement, C4: Indwelling drip retention and connection, C5: Oral care, and C6: Diaper exchange and cleaning of area. Each of these activities is performed 5 times by each nurse. The data is divided into 1-minute segments.

Because of the noisy and missing data problem in Motion capture and Meditag data, we only use the accelerometer data in our experiment. The data

| | Complex Nursing Event Name | Complex Nursing Event logic |
|---|---|---|
| **Complex Event** | Physiological Measurement | Vital sign ⇒ blood glucose measure ⇒ blood collection |
| | Indwelling Drip | Vital sign ⇒ Indwelling drip |
| | Patient Cleaning | Oral care ⇒ Diaper exchange |
| **Protocol Violation** | Unsanitary Operation No.1 | Diaper exchange ⇒ blood collection |
| | Unsanitary Operation No.2 | Area cleaning ⇒ blood glucose measure |
| | Unsanitary Operation No.3 | Diaper exchange ⇒ indwelling drip |

Table 5: Logic of Complex Nursing Events

segments with less than 50 seconds are removed from the dataset. To extract features from the variable-length, un-uniformly sampled accelerometer data, we first divide the segment into 30 non-overlapping windows where the duration of each window is 2 seconds. For the data segments with a length smaller than 60 seconds, we perform imputation by filling the missing features with feature values of the last time window. We split the original dataset into a training set and a validation set. The data in the training set is used to generate complex nursing events data, and the validation set is used for evaluating the performance of the perception module.

To construct complex nursing events, we follow a similar approach as previous experiments. In the first simulation(Sim 1 in table 7), we randomly sample 10 data segments from the Nursing Activity dataset and concatenate them together representing a nursing activity sequence that takes over 10 minutes. The logic of our complex nursing events can be categorized into two groups as shown in table 5: complex nursing events, and violations of sanitary protocol. The constructed complex nursing events dataset has 2319 samples in total, each of which is an accelerometer sequence of ten minutes.

In this experiment, we use a convolutional LSTM structure [41] that is usually used to analyze IMU data. The model contains two 2-D convolution blocks with one LSTM layer and one fully-connected layer. A 0.5 dropout is applied to every layer. As the high-level reasoning logic have the same level of complexity, we keep using the one-layer LSTM network as the NRLogic model. We compare NEUROPLEX with three different baseline models:

- **ConvLSTM:** Like the previous experiments, this baseline has exactly the same network architecture as NEUROPLEX, but without human knowledge injected.
- **ConvLSTM-2:** This model has a similar structure to ConvLSTM. Instead of adding another LSTM network after the Conv-LSTM network, it learns to output the label of the complex event directly.
- **LSTM-Attention:** This model is inspired by [10], which demonstrates

| | NEUROPLEX | ConvLSTM | ConvLSTM-2 | LSTM-Attention |
|---|---|---|---|---|
| Perception Acc | **77.59%** | 1.72% | NA | NA |
| Validation MAE | **0.0027** | 0.1430 | 0.1860 | 0.6245 |
| R-Square | **1.000** | 0.882 | 0.807 | 0.002 |
| Converted Acc | **100%** | 93.67% | 89.28% | 78.81% |

Table6: Model performance on complex nursing event data

good performance in the Nursing Activity Challenge. Instead of using the GRU layer, this model uses two layers of LSTM, and the sequence of hidden states are aggregated using the attention mechanism to get a score vector. Two score vectors from two LSTM layers are concatenated, and a fully-connected layer is added to get the softmax result.

All the models above are trained for 400 epochs with Adam optimizer and 0.001 learning rate. The batch size is set to 256. In addition to the converted accuracy, we use the metric R-square to evaluate the performance of the regression task. R-Square measures how well the model fits the dependent variables. The value is usually between 0 to 1, and a bigger value indicates a better fit between the predicted and actual value. R-Square is calculated as follows:

$$R^2 = 1 - (\sum_i (y_i - f_i)^2)/(\sum_i (y_i - \overline{y})^2)$$

Table 6 shows that the proposed NEUROPLEX performs well on complex nursing event detection with raw accelerometer data, and the perception model also gets near-native accuracy. (80% accuracy when perception network trained directly on Nursing activity dataset). The other deep learning baselines fail to show comparable performance, and the NEUROPLEX shows the best performance in all different metrics.

**Detection over long period of time.** To further test whether NEUROPLEX scales well on complex nursing event detection, we conduct a set of experiments with increasing length of the time window, and keep the definition of the complex nursing event the same as that in the simulation 1. We test the NEUROPLEX with the other three baseline models, and measure their performance based on R-square and converted accuracy. As shown in table7, as the length of time window increases, the complex event detection becomes harder, and the model performance degrades. We notice that when the time window length is less than 30 minutes, NEUROPLEX can successfully learn to detect complex event at high accuracy. As the time window length grows, it takes more time for the model to converge, and 400 epochs are not sufficient to get the optimum, although it already performs pretty well and beats other baselines by a large margin. In simulation 6, we can see that the NEUROPLEX model can still get about 80% detection accuracy in a one-hour-long time window, proving that NEUROPLEX is robust to long-term reasoning.

The other neural-network-based models all suffer from the long temporal range. As we can see, in simulation 1, the ConvLSTM models is able to detect complex events with acceptable performance. However, as the length of time window increases, their performance drops quickly. The ConvLSTM-2 model

performs better than ConvLSTM, suggesting that the added LSTM network in ConvLSTM does not help capture temporal dependencies without human knowledge injected. Although it demonstrates good performance on other tasks, the AttentionNet can not learn complex event on accelerometer data. It basically gets an approximate zero R-square value in all the simulations (the R-square is negative in some cases because the model performs worse than the null hypothesis). The converted accuracy in sim 1 is 78.81% because it outputs small fraction numbers which are rounded to zeros.

| Methods | Sim 1 | Sim 2 | Sim 3 | Sim 4 | Sim 5 | Sim 6 |
|---|---|---|---|---|---|---|
| Time window (minutes) | 10 | 20 | 30 | 40 | 50 | 60 |
| R-square | | | | | | |
| Neuroplex | **1.00** | **0.99** | **1.00** | **0.90** | **0.88** | **0.85** |
| ConvLSTM | 0.88 | 0.90 | 0.66 | 0.32 | 0.33 | 0.35 |
| ConvLSTM-2 | 0.81 | 0.76 | 0.80 | 0.76 | 0.75 | 0.70 |
| AttentionNet | 0.02 | 0 | 0 | 0 | -0.01 | -0.02 |
| Converted Accuracy | | | | | | |
| Neuroplex | **100%** | **98.90%** | **100%** | **83.59%** | **79.00%** | **79.63%** |
| ConvLSTM | 93.67% | 83.29% | 67.75% | 40.79% | 39.03% | 37.47% |
| ConvLSTM-2 | 89.28% | 80.08% | 75.70% | 60.30% | 45.83% | 39.48% |
| AttentionNet | 78.81% | 2.60% | 0.62% | 0.50% | 0.11% | 0.02% |

Table 7: Experiment result of complex nursing activity detection as the length of time window increases.

# 6 Discussion

The design of NEUROPLEX framework is inspired by the human learning process: the perception ability is trained in a data-driven approach, and the mid- to high-level reasoning ability can be taught in an efficient manner—the knowledge is passed in a condensed form of logic rules. Human can know what complex event is even without seeing any examples before. On the other hand, human store knowledge in neurons inside the brain, so high-level logic should be expressed as a neural network as well.

In NEUROPLEX, we create a dual form of the reasoning module so that the system learns by back-propagation in a standard supervised manner. During the inference stage, even though the NRLogic is also available for the forward propagation, we use the logical machine instead. This is because the logical machine is more reliable and explainable than the NRLogic model. Additionally, the logical rules are often more compact than deep neural networks, so they can usually be executed quickly over sensor networks with minimal computation overhead. We enumerate the limitations of the current approach and future research directions as follows:

**Distribution of complex sensor reasoning.** The hybrid architecture of NEUROPLEX is designed to thrive in emerging distributed computation architectures that push sensor inferencing towards edge devices. However, primitive events that stem from edge device inferences are currently fused at a single

CEP node. Future work can focus on distributing the reasoning module across heterogeneous sensor networks with dynamic computation placement.

**Prior knowledge of complex event reasoning logic.** We have assumed that the higher-level logic of a complex event is provided by the user. This is a reasonable assumption, and it is broadly used not only in neural-symbolic systems [22, 42, 34], but also in earlier works on rule-based activity recognition [26, 33], activity decomposition using sensory grammar[23], and complex event processing [9, 8, 28]. During training, the use of prior knowledge reduces the burden of data annotation and accelerates the learning process significantly. Additionally, we use the CEP language with a BNF grammar like DeepCEP[42] to formally define the logic rules, which standardizes and simplifies the coding of human knowledge.

However, the system may be deployed in a new or evolving sensor network environment–where the definition of complex events provided by the user may not present robust detection. Future efforts can focus on learning or fine-tuning the reasoning module by freezing the trained perception module and updating the NRLogic module. After learning, the FSM can be extracted from the trained RNN network to provide human-understandable logic.

**Complexity of logical reasoning.** In the context of complex event detection, the NRLogic model is trained to capture the logic of a complex event, which could be arbitrarily complex. The NEUROPLEX framework can be generalized to a wide range of scenarios, and NRLogic is able to capture not only temporal logic, but also spatial logic over sensory networks. For all experiments in this paper, the complex event logic can be captured effectively by an LSTM network with one hidden layer. However, as the complexity of logic increases, deeper and more sophisticated networks need to be used to approximate the logical function. Future research will investigate what the most efficient structure is to capture the reasoning logic of different complexity in different settings.

**Annotation of complex event sensor data.** In the problem of complex event detection, we only care about the occurrence of complex events at the current time. Thus, the user would only make annotations when complex events are actually happening and do not need to care about simple events. This greatly reduces the labeling burden for numerous applications. In this work, we formulate the complex event detection as a regression problem, which requires annotations of complex events happening times. This is sometimes not a trivial task because sensors with events happening over long periods could generate multiple primitive events of the same type, leading to an increase in complex event instances. Also, in real-world scenarios, the complex event labels may not have accurate spatial-temporal metadata, e.g., inaccurate timestamps. Future research will investigate how robust the proposed NEUROPLEX is when the annotation is noisy, and how to aggregate and learn from events of the same type.

**Generalization to real multimodal scenarios.** Although NEUROPLEX is evaluated on complex events from a single modality in this paper, we believe that it can be extended to multimodal complex event learning tasks as well. We envision a framework similar to [27, 25], with multimodal data streams input

as different channels. While the performance of NEUROPLEX in multimodal scenarios is not presented in this paper as the on-going pandemic conditions prevent us from conducting the requisite experimental study, we plan to address it in future work.

# 7 Related Work

A body of earlier work studied complex events in sensor networks [23, 48, 47]; however, we focus on detecting complex events over unstructured data using deep learning models. We first provide an overview of the state-of-the-art deep learning methods related to complex activity detection. After highlighting the scalability and data efficiency issues of these approaches, we describe how prior works have attempted to integrate human logic directly into the deep learning models. Although these approaches help to regularize and bootstrap the associated learning processes, they fail to address the notions of scalability in the spatial-temporal domains. We then highlight a recent class of neuro-symbolic approaches that combine deep learning with explicit symbolic reasoning, and discuss modern approaches to complex event detection using hybrid systems.

**Deep learning for complex activities.** Prior work on deep learning methods has explored learning and analyzing time-series data such as human activities. In particular, several works in video classification have proposed solutions for detecting *complex activities* over short periods [39, 15]. Images [43] and audio [6] are also considered to contain complex events, and several studies [38, 14] use multimodal information to perform classification. Similarly, anomalous event detection [44] utilizes motion features to extract temporal-spatial localization features for complex event detection.

Although the aforementioned works have shown promising results in their respective domains, they do not have a clear definition of *complex events*. Generally, they use the term to describe events that contain interactions between different elements. Furthermore, these works typically only consider processing information from a single input instead of a distributed set of heterogeneous sensors. Although multimodal data fusion has been explored, they fail to fuse the information at a semantic level so as to provide a clear explanation of the result. Additionally, these learning-based models alone cannot learn extremely long temporal dependencies well, even with the help of the LSTM [12] structure and the Attention mechanism[1]. They typically reason about events on the order of seconds. Finally, in order to have effective models that generalize well, learning-based methods necessitate the consumption of large amounts of data with an expensive annotation process [7].

Intuitively, integration of human logic would address these issues in an interpretable way. We next review how prior works have integrated logic reasoning with machine learning (ML).

**Combining logic reasoning with ML.** Combining reasoning with learning is a popular topic in the AI field, and one interesting direction is integrating symbolic human knowledge with ML models. One approach to integra-

tion is to instrument logical formulae into an embedding space while preserving the logical meaning and the relationship between formulae. ConvNet Encoder [16] and TreeLSTM [32, 19, 46] embed formulae using different network structures. LENSR [40] first converts logical formulae into d-DNNF DAGs and use a Graph Convolutional Network to perform an embedding. Another approach uses knowledge to impose additional logic loss to help augment the original training objective [45, 31, 5, 29]. In [13], a distillation method is used to transfer knowledge from a rule-regularized teacher network to a standard student network. [45] imposes a semantic loss on predicted probability by quantifying the probability of generating a satisfying assignment by randomly sampling from the predictive distribution.

However, the preceding methods are not designed for reasoning about spatial and temporal events at scale. Therefore, we next discuss neural-symbolic frameworks that allow a hierarchy of reasoning between deep learning and human logic.

**Hybrid Neural-symbolic frameworks for complex tasks.** Building hybrid systems that utilize the power of both human logic and deep learning is becoming a hot trend. Caesar [22] proposes a system that uses both deep learning models and rule-defined complex activity graphs to recognize complex activities in a multi-camera video surveillance setting. [34] shows that it is possible to fuse proxy deep learning models and use ProbLog [2] defined rules to perform crime detection. In [42], simple events are first captured by deep data abstractors and then reasoned about by a complex event processing (CEP) engine that takes human definitions as logic rules. All of these works use either existing or newly defined languages to inject human knowledge into the system to perform reasoning to detect complex events. They utilize pre-trained neural network models and only focus on the inference path of the problem as learning is beyond their scope.

The ability to *learn* is of great importance, especially when the system is deployed in a new environment. SATNet [36] sits at the boundary between end-to-end deep learning models and neural-symbolic hybrid approaches. It introduced a differentiable SAT solver that can be integrated into deep learning models as a MAXSAT layer. This logical structure can be learned using a supervised end-to-end approach. However, the SATNet does not have symbolic representation, nor is the MAXSAT layer explainable.

DeepProbLog [24] provides a generalized probabilistic logic programming language that incorporates deep learning into ProbLog. The parameters of both neural networks and logical rules are learned in an end-to-end manner using $\alpha$ProbLog while supporting symbolic representation. An important feature for DeepProbLog is that it supports symbolic representation inside the system. Therefore, unlike black-box deep learning models, the results of DeepProbLog are explainable. While DeepProbLog is designed to handle complex problems where the logic can be expressed as combinational logic, it struggles to represent the sequential logic in ProbLog as the number of nodes grows exponentially.

Unlike prior works, in our approach, logical knowledge is not used to augment training objectives, but rather perform individual reasoning tasks. Trained to

mimic the logic rules, the NRLogic is plugged into the system as a layer to perform learning tasks.

# 8 Conclusion

In this work, we presented NEUROPLEX, a neural-symbolic framework for detecting complex events. Using semantic knowledge to guide the learning, NEUROPLEX can learn to detect complex events with much fewer and sparse annotations. Results on different datasets proved the effectiveness, reliability, and robustness of NEUROPLEX. Future work will focus on the deployment of NEUROPLEX in real-world scenarios with multimodal sensory data.

# Acknowledgments

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement # W911NF-16-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

# References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[2] Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. Problog: A probabilistic prolog and its application in link discovery. In *IJCAI*, volume 7, pages 2462–2467. Hyderabad, 2007.

[3] Gert Dekkers, Steven Lauwereins, Bart Thoen, Mulu Weldegebreal Adhana, Henk Brouckxon, Toon van Waterschoot, Bart Vanrumste, Marian Verhelst, and Peter Karsmakers. The SINS database for detection of daily activities in a home environment using an acoustic sensor network. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, pages 32–36, November 2017.

[4] Gert Dekkers, Lode Vuegen, Toon van Waterschoot, Bart Vanrumste, and Peter Karsmakers. Dcase 2018 challenge-task 5: Monitoring of domestic activities based on multi-channel acoustics. *arXiv preprint arXiv:1807.11246*, 2018.

[5] Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. Lifted rule injection for relation embeddings. *arXiv preprint arXiv:1606.08359*, 2016.

[6] Miquel Espi, Masakiyo Fujimoto, Keisuke Kinoshita, and Tomohiro Nakatani. Exploiting spectro-temporal locality in deep learning based acoustic event detection. *EURASIP Journal on Audio, Speech, and Music Processing*, 2015(1):26, 2015.

[7] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33(4):917–963, 2019.

[8] Ioannis Flouris, Nikos Giatrakos, Antonios Deligiannakis, Minos Garofalakis, Michael Kamp, and Michael Mock. Issues in complex event processing: Status and prospects in the big data era. *Journal of Systems and Software*, 127:217–236, 2017.

[9] Lajos Jenő Fülöp, Gabriella Tóth, Róbert Rácz, János Pánczél, Tamás Gergely, Arpád Beszédes, and Lóránt Farkas. Survey on complex event processing and predictive analytics. In *Proceedings of the Fifth Balkan Conference in Informatics*, pages 26–31. Citeseer, 2010.

[10] Md Nazmul Haque, Mahir Mahbub, Md Hasan Tarek, Lutfun Nahar Lota, and Amin Ahsan Ali. Nurse care activity recognition: A gru-based approach with attention mechanism. In *Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers*, pages 719–723, 2019.

[11] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[12] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[13] Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. Harnessing deep neural networks with logic rules. *arXiv preprint arXiv:1603.06318*, 2016.

[14] I-Hong Jhuo and DT Lee. Video event detection via multi-modality deep learning. In *2014 22nd International Conference on Pattern Recognition*, pages 666–671. IEEE, 2014.

[15] Yu-Gang Jiang, Zuxuan Wu, Jun Wang, Xiangyang Xue, and Shih-Fu Chang. Exploiting feature and class relationships in video categorization with regularized deep neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 40(2):352–364, 2018.

[16] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. Character-aware neural language models. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[17] Paula Lago, Sayeda Shamma Alia, Shingo Takeda, Tittaya Mairittha, Nattaya Mairittha, Farina Faiz, Yusuke Nishimura, Kohei Adachi, Tsuyoshi Okita, François Charpillet, et al. Nurse care activity recognition challenge: summary and results. In *Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers*, pages 746–751, 2019.

[18] Martin Längkvist, Lars Karlsson, and Amy Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42:11–24, 2014.

[19] Phong Le and Willem Zuidema. Compositional distributional semantics with long short term memory. *arXiv preprint arXiv:1503.02510*, 2015.

[20] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[21] Kun Liu, Wu Liu, Chuang Gan, Mingkui Tan, and Huadong Ma. T-c3d: temporal convolutional 3d network for real-time action recognition. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[22] Xiaochen Liu, Pradipta Ghosh, Ulutan Oytun, B.S. Manjunath, Kevin Chan, and Ramesh Govindan. Caesar: Cross-camera complex activity recognition. In *17th ACM Conference on Embedded Networked Sensor Systems, SENSYS 2019*. Association for Computing Machinery, Inc, 2019.

[23] Dimitrios Lymberopoulos, Abhijit S Ogale, Andreas Savvides, and Yiannis Aloimonos. A sensory grammar for inferring behaviors in sensor networks. In *Proceedings of the 5th international conference on Information processing in sensor networks*, pages 251–259, 2006.

[24] Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Deepproblog: Neural probabilistic logic programming. In *Advances in Neural Information Processing Systems*, pages 3749–3759, 2018.

[25] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. Multimodal deep learning. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 689–696, 2011.

[26] Siyuan Qi, Siyuan Huang, Ping Wei, and Song-Chun Zhu. Predicting human activities using stochastic grammar. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1164–1172, 2017.

[27] Valentin Radu, Catherine Tong, Sourav Bhattacharya, Nicholas D Lane, Cecilia Mascolo, Mahesh K Marina, and Fahim Kawsar. Multimodal deep learning for activity and context recognition. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(4):1–27, 2018.

[28] D Robins. Complex event processing. In *Second International Workshop on Education Technology and Computer Science. Wuhan*, pages 1–10. Citeseer, 2010.

[29] Tim Rocktäschel, Sameer Singh, and Sebastian Riedel. Injecting logical background knowledge into embeddings for relation extraction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1119–1129, 2015.

[30] Baoguang Shi, Xiang Bai, and Cong Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(11):2298–2304, 2016.

[31] Russell Stewart and Stefano Ermon. Label-free supervision of neural networks with physics and domain knowledge. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[32] Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015.

[33] Mohammed Yassine Kazi Tani, Adel Lablack, Abdelghani Ghomari, and Ioan Marius Bilasco. Events detection using a video-surveillance ontology and a rule-based approach. In *European Conference on Computer Vision*, pages 299–308. Springer, 2014.

[34] Marc Roig Vilamala, Liam Hiley, Yulia Hicks, Alun Preece, and Federico Cerutti. A pilot study on detecting violence in videos fusing proxy models. In *Fusion*, 2019.

[35] Marc Roig Vilamala, Harrison Taylor, Tianwei Xing, Luis Garcia, Mani Srivastava, Lance Kaplan, Alun Preece, Angelika Kimming, and Federico Cerutti. A hybrid neuro-symbolic approach for complex event processing (extended abstract). In *EPTCS proceedings of ICLP*, 2020.

[36] Po-Wei Wang, Priya L Donti, Bryan Wilder, and Zico Kolter. Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. *arXiv preprint arXiv:1905.12149*, 2019.

[37] Eugene Wu, Yanlei Diao, and Shariq Rizvi. High-performance complex event processing over streams. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 407–418. ACM, 2006.

[38] Zuxuan Wu, Yu-Gang Jiang, Xi Wang, Hao Ye, and Xiangyang Xue. Multi-stream multi-class fusion of deep networks for video classification. In *Proceedings of the 24th ACM international conference on Multimedia*, pages 791–800. ACM, 2016.

[39] Zuxuan Wu, Xi Wang, Yu-Gang Jiang, Hao Ye, and Xiangyang Xue. Modeling spatial-temporal clues in a hybrid deep learning framework for video classification. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 461–470. ACM, 2015.

[40] Yaqi Xie, Ziwei Xu, Kuldeep Meel, Mohan S Kankanhalli, and Harold Soh. Semantically-regularized logic graph embeddings. *arXiv preprint arXiv:1909.01161*, 2019.

[41] Tianwei Xing, Sandeep Singh Sandha, Bharathan Balaji, Supriyo Chakraborty, and Mani Srivastava. Enabling edge devices that learn from each other: Cross modal training for activity recognition. In *Proceedings of the 1st International Workshop on Edge Systems, Analytics and Networking*, pages 37–42. ACM, 2018.

[42] Tianwei Xing, Marc Roig Vilamala, Luis Garcia, Federico Cerutti, Lance Kaplan, Alun Preece, and Mani Srivastava. Deepcep: Deep complex event processing using distributed multimodal information. In *2019 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 87–92. IEEE, 2019.

[43] Yuanjun Xiong, Kai Zhu, Dahua Lin, and Xiaoou Tang. Recognize complex events from static images by fusing deep channels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1600–1609, 2015.

[44] Dan Xu, Elisa Ricci, Yan Yan, Jingkuan Song, and Nicu Sebe. Learning deep representations of appearance and motion for anomalous event detection. *arXiv preprint arXiv:1510.01553*, 2015.

[45] Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck. A semantic loss function for deep learning with symbolic knowledge. *arXiv preprint arXiv:1711.11157*, 2017.

[46] Xiaodan Zhu, Parinaz Sobihani, and Hongyu Guo. Long short-term memory over recursive structures. In *International Conference on Machine Learning*, pages 1604–1612, 2015.

[47] Michael Zoumboulakis and George Roussos. Escalation: Complex event detection in wireless sensor networks. In *European Conference on Smart Sensing and Context*, pages 270–285. Springer, 2007.

[48] Michael Zoumboulakis and George Roussos. Complex event detection in extremely resource-constrained wireless sensor networks. *Mobile Networks and Applications*, 16(2):194–213, 2011.

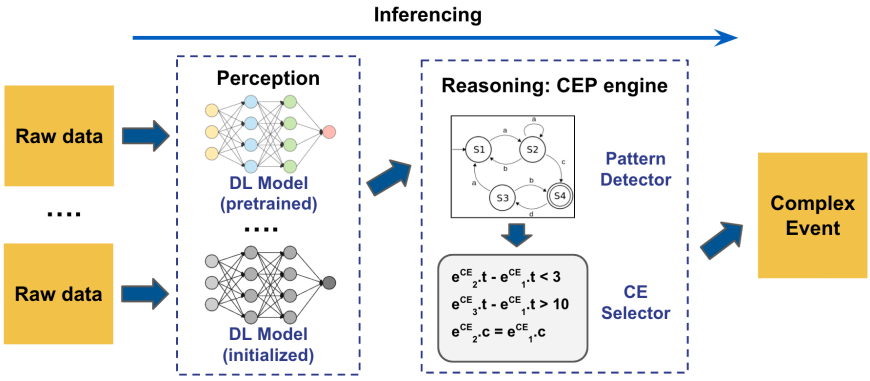


Figure 3: NEUROPLEX neural-symbolic initialization. The reasoning module is initialized with human-defined pattern detectors and logical constraints, while the perception module is initialized with a set of deep learning models that may or may not be pre-trained for each raw data source.

⟨*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)*;

Figure 4: Simplified CEP Grammar, whose syntax semantics are significantly adapted from [42]. In this context, the grammar is used to initialize the training framework as opposed to define an explicit CEP engine.

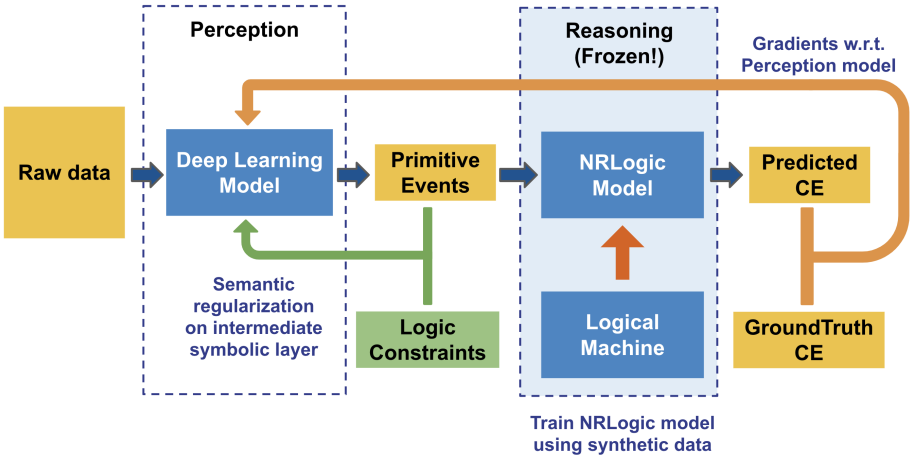


Figure 5: Training on NEUROPLEX's perception module.

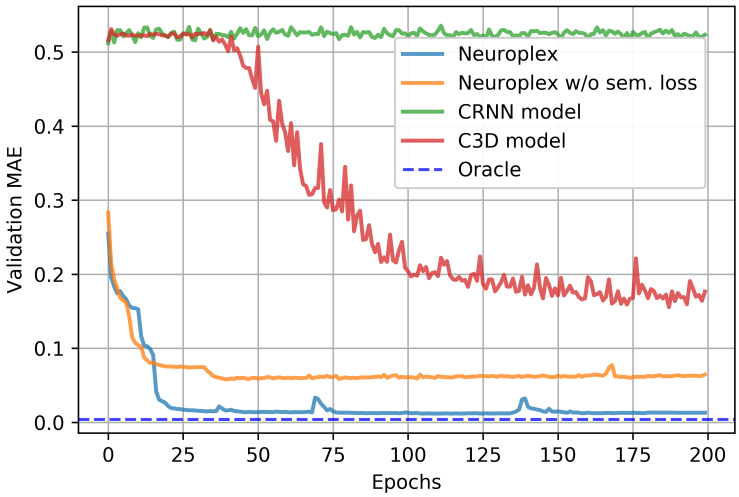


Figure 6: Prediction MAE (on the validation set) changes as training progressed in simulation-1

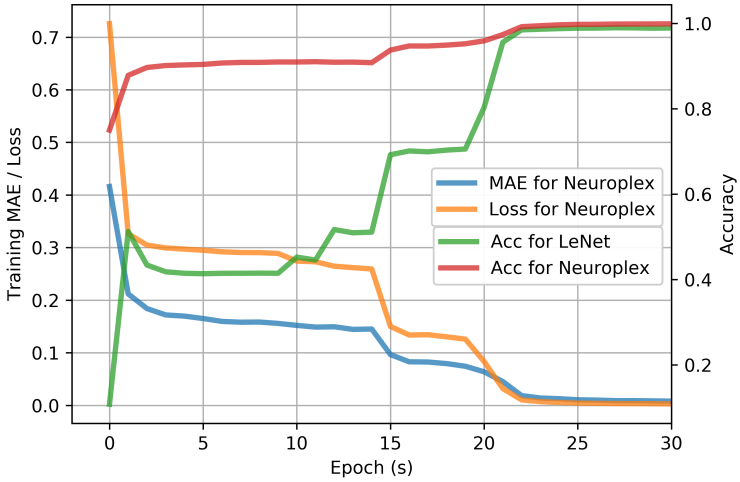


Figure 7: The performance of perception model increases as training processed in NEUROPLEX

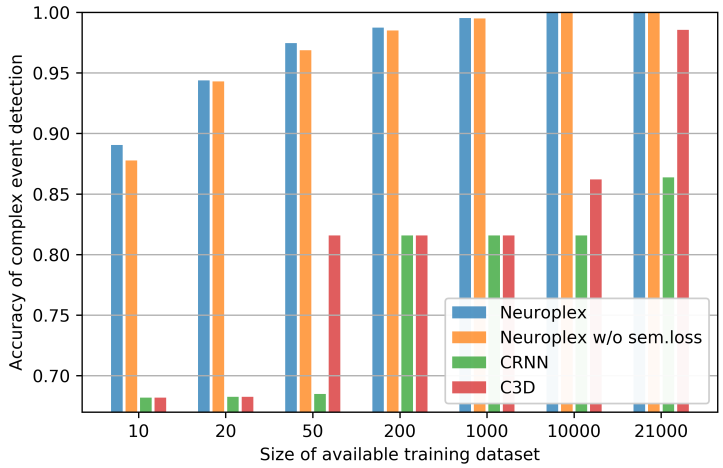


Figure 8: The performance changes with size of training data

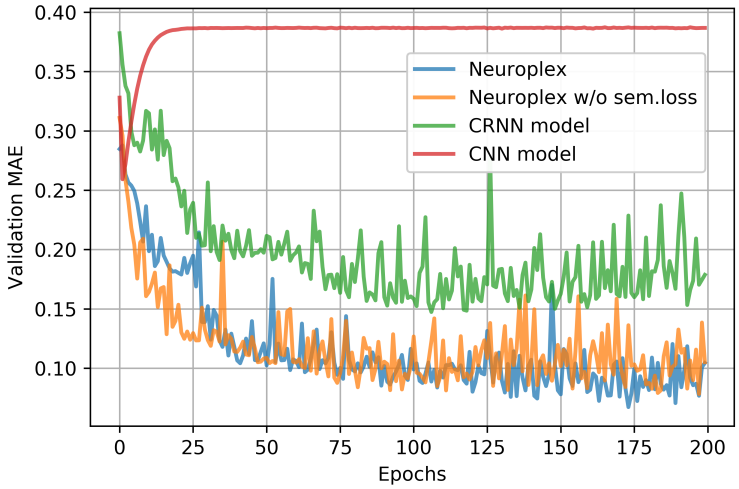


Figure 9: Learning Curves on Complex Audio Events