Temporal and Object Quantification Networks

Jiayuan Mao[1*] , **Zhezheng Luo**[1*] , **Chuang Gan**[2] , **Joshua B. Tenenbaum**[1] , **Jiajun Wu**[3] , **Leslie Pack Kaelbling**[1] and **Tomer D. Ullman**[4]

[1] MIT [2] MIT-IBM Watson AI Lab [3] Stanford University [4] Harvard University

Abstract

We present *Temporal and Object Quantification Networks* (TOQ-Nets), a new class of neuro-symbolic networks with a structural bias that enables them to learn to recognize complex relational-temporal events. This is done by including reasoning layers that implement finite-domain quantification over objects and time. The structure allows them to generalize directly to input instances with varying numbers of objects in temporal sequences of varying lengths. We evaluate TOQ-Nets on input domains that require recognizing event-types in terms of complex temporal relational patterns. We demonstrate that TOQ-Nets can generalize from small amounts of data to scenarios containing more objects than were present during training and to temporal warpings of input sequences.

1 Introduction

Every day, people interpret events and actions in terms of concepts, defined over temporally evolving relations among agents and objects (Zacks *et al.*, 2007; Stränger and Hommel, 1996). For example, in a soccer game, people can easily recognize when one player has control of the ball, when a player *passes* the ball to another player, or when a player is *offsides*. Although it requires reasoning about intricate relationships among sets of objects over time, this cognitive act is effortless, intuitive, and fast. It also generalizes directly over different numbers and arrangements of players, and detailed timings and trajectories. In contrast, most computational representations of sequential concepts are based on fixed windows of space and time, and lack the ability to perform relational generalization.

In this paper, we develop generalizable representations for learning complex activities in time sequences from realistic data. As illustrated in Fig. 1, we can describe complex events with a first-order linear temporal logic (FO-LTL; Pnueli, 1977) formula, which allows us to flexibly decompose an input sequence into stages that satisfy different criteria over time. Object quantifiers ($\forall$ and $\exists$) are used to specify conditions on sets of objects that define each stage. Such representations immediately support generalization to situations with a varying number of objects, and sequences with different time warpings.

More concretely, the variety of complicated spatio-temporal trajectories that *high pass* can refer to in a soccer game can be described in these terms: in a high pass from player A to teammate B, A is *close* to the ball ($\mathit{distance}(A, \mathit{ball}) < \theta_1$) and moving ($\mathit{velocity}(A) > \theta_2$) *until* the ball moves over the ground ($z_{\mathit{position}}(\mathit{ball}) > \theta_3$), which is in turn *until* teammate B gets control of the ball ($\mathit{teammate}(A, B) \wedge \mathit{distance}(B, \mathit{ball}) < \theta_1$). Beyond modeling human actions in physical environments, these structures can be applied to events in any time sequence of relational states, e.g., characterizing particular offensive or defensive maneuvers in board games such as chess or in actual conflicts, or detecting a process of money-laundering amidst financial transaction records.

In this paper, we propose a neuro-symbolic approach to learning to recognize temporal relational patterns, called *Temporal and Object Quantification Networks* (TOQ-Nets), in which we design structured neural networks with an explicit bias that represents finite-domain quantification over both entities and time. A TOQ-Net is a multi-layer neural network whose inputs are the properties of agents and objects and their relationships, which may change over time. Each layer in the TOQ-Net performs either *object* or *temporal* quantification.

The key idea of TOQ-Nets is to use *tensors* to represent relations between objects, and to use tensor pooling operations over different dimensions to realize temporal and object quantifiers ($\boldsymbol{G}$ and $\forall$). For example, the colorful matrix in Fig. 1(a) can be understood as a matrix representing a unary color property of a set of 8 entities over a sequence of 8 time steps. Representing a sequence of relations among objects over time would require a 3-dimensional tensor. Crucially, the design of TOQ-Nets allows *the same network weights* to be applied to domains with different numbers of objects and time sequences of different lengths. By stacking object and temporal quantification operations, TOQ-Nets can easily learn to represent higher-level sequential concepts based on the relations between entities over time, starting from low-level sensory input and supervised with only high-level class labels.

There are traditional symbolic learning or logic synthesis methods that construct first-order or linear temporal logic expressions from accurate symbolic data (Neider and Gavran,

*: equal contribution; order determined by a coin toss.
Email: {jiayuanm,ezzluo}@mit.edu.

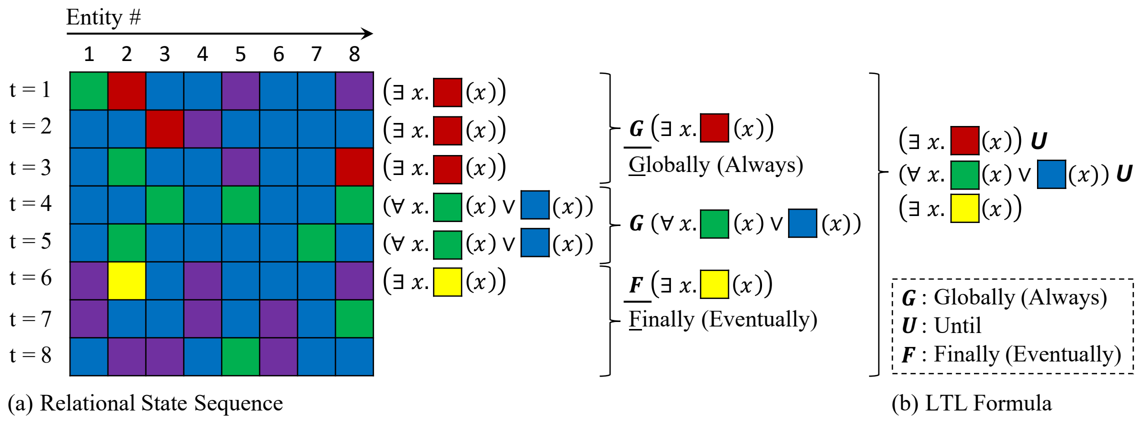


Figure 1: (a) An input sequence composed of relational states: each column represents the state of an entity that changes over time. A logic formula describes a complex concept or feature that is true of this temporal sequence using object and temporal quantification. The sequence is segmented into three stages: throughout the first stage, 🟥 holds for at least one entity, until the second stage, in which each entity is always either 🟦 or 🟩, until the third stage, in which 🟨 eventually becomes true for at least one of the entities. (b) Such events can be described using first-order linear temporal logic expressions.

2018; Camacho *et al.*, 2018; Chou *et al.*, 2020). TOQ-Nets take a different approach and can learn from noisy data by backpropagating gradients, which allows them to start with a general perceptual processing layer that is directly fed into logical layers for further processing.

We evaluate TOQ-Nets on two perceptually and conceptually different benchmarks: trajectory-based sport event detection and human activity recognition, demonstrating several important contributions. First, TOQ-Nets outperform both convolutional and recurrent baselines for modeling temporal-relational concepts across benchmarks. Second, by exploiting temporal-relational features learned through supervised learning, TOQ-Nets achieve strong few-shot generalization to novel actions. Finally, TOQ-Nets exhibit strong generalization to scenarios with more entities than were present during training and are robust w.r.t. time warped input trajectories. These results illustrate the power of combining symbolic representational structures with learned continuous-parameter representations to achieve robust, generalizable interpretation of complex relational-temporal events.

2 TOQ-Nets

The input to a TOQ-Net is a tensor representation of the properties of all entities at each moment in time. For example, in a soccer game, the input encodes the position of each player and the ball, as well as their team membership at each step of an extended time interval. The output is a label of the category of the sequence, such as the type of soccer play it contains.

The first layer of a TOQ-Net (Fig. 2 (i)) extracts temporal features for each entity with an *input feature extractor* that focuses on entity features within a fixed and local time window. These features may be computed, e.g., by a convolutional neural network or a bank of parametric feature templates. The output of this step is a collection of nullary, unary, and binary relational features over time for all entities. Throughout the paper we will assume that all output tensors of this layer are binary-valued, but it can be extended directly to real-valued functions. This input feature extractor is task-specific and is not the focus of this paper.

Second, these temporal-relational features go through several *relational reasoning* layers (RRLs), detailed in Section 2.2, each of which performs linear transformations, sigmoid activation, and *object quantification* operations. The linear and sigmoid functions allow the network to realize learned Boolean logical functions, and the object quantification operators can realize quantifiers. Additional RRLs enable deeper nesting of quantified expressions, as illustrated in Fig. 2. All operations in these layers are performed for all time steps in parallel.

Next, the RRLs perform a final quantification, computing for each time step a set of a nullary features that are passed to the *temporal reasoning layers* (TRLs), as detailed in Section 2.3. Each TRL performs linear transformations, sigmoid activation, and *temporal quantification*, allowing the model to realize a subset of linear temporal logic (Pnueli, 1977). As with RRLs, adding more TRLs enables the network to realize logical forms with more deeply nested temporal quantifiers.

In the last layer, all object and time information is projected into a set of features of the initial time step, which summarize the temporal-relational properties of the entire trajectory (e.g., "the kicker eventually scores"), and fed into a final softmax unit to obtain classification probabilities for the sequence.

It is important to understand the representational power of this model. The *input transformation layer* learns basic predicates and relations that will be useful for defining more complex concepts, but no specific predicates or relations are built into the network in advance. The relational reasoning layers build quantified expressions over these basic properties and relations, and might construct expressions that could be interpretable as "the player is close to the ball." Finally, the temporal reasoning layer applies temporal operations to these complex expressions, such as "the player is close to the ball until the ball moves with high speed." Critically, *none* of the symbolic properties or predicates are hand defined—they are all constructed by the initial layer in order to enable the network to express the concept it is being trained on.

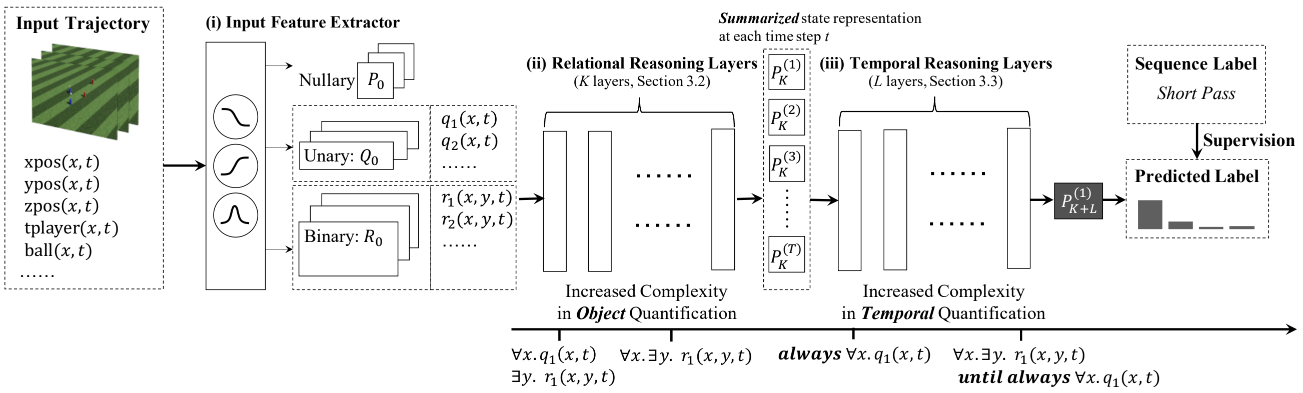

Input Trajectory
xpos(x, t)
ypos(x, t)
zpos(x, t)
tplayer(x, t)
ball(x, t)
……
(i) Input Feature Extractor
Nullary P_0
Unary: Q_0
$q_1(x,t)$
$q_2(x,t)$
……
Binary: R_0
$r_1(x,y,t)$
$r_2(x,y,t)$
……
Summarized state representation at each time step *t*
(ii) Relational Reasoning Layers
(*K* layers, Section 3.2)
$P_K^{(1)}$ $P_K^{(2)}$ $P_K^{(3)}$ $P_K^{(T)}$
(iii) Temporal Reasoning Layers
(*L* layers, Section 3.3)
$P_{K+L}^{(1)}$
Sequence Label
Short Pass
Supervision
Predicted Label
Increased Complexity in *Object* Quantification
Increased Complexity in *Temporal* Quantification
$\forall x.\, q_1(x,t)$
$\exists y.\ r_1(x,y,t)$
$\forall x. \exists y.\ r_1(x,y,t)$
always $\forall x.\, q_1(x,t)$
$\forall x. \exists y.\ r_1(x,y,t)$
until always $\forall x.\, q_1(x,t)$

Figure 2: A TOQ-Net contains three modules: (i) an input feature extractor, (ii) relational reasoning layers, and (iii) temporal reasoning layers. To illustrate the model's representational power, we show that logical forms of increasing complexity can be realized by stacking multiple layers.

TOQ-Nets are not fully first-order: all quantifiers operate only over the finite domain of the input instance, and can be seen as "short-hand" for finite disjunctions or conjunctions over objects or time points. In addition, the depth of the logical forms it can learn is determined by the fixed depth of the network. However, our goal is not to fully replicate temporal logic, but to bring ideas of object and temporal quantification into neural networks, and to use them as structural inductive biases to build models that generalize better from small amounts of data to situations with varying numbers of objects and time courses.

2.1 Temporal-Relational Feature Representation

TOQ-Nets use tensors as internal representations between layers; they represent, all at once, the values of all predicates and relations grounded on all objects at all time points. The operations in a TOQ-Net are vectorized, operating in parallel on all objects and times, sometimes expanding the dimensionality via outer products, and then re-projecting into smaller dimensions via max-pooling. This processing style is analogous to representing an entire graph using an adjacency matrix and using matrix operations to compute properties of the nodes or of the entire graph. In TOQ-Nets, the input to the network, as well as the feature output of intermediate layers, is represented as a tuple of three tensors.

Specifically, we use a vector of dimension D_0 to represent aspects of the state that are global and do not depend on any specific object at each time t. We use a matrix of shape $N \times D_1$ to represent the unary properties of each entity at time t, where N is the number of entities and D_1 is the hidden dimension size. Similarly, we use a tensor of shape $N \times N \times D_2$ to represent the relations between each pair of entities at time step t. As a concrete example, illustrated in Fig. 2, the number of entities N is the total number of players plus one (the ball). For each entity x and each time step, the inputs are their 3D position, type (ball or player) and team membership. The TOQ-Net outputs the action performed by the target player. Since there are only entity features, the input trajectory is encoded with a "unary" tensor of shape $T \times N \times D_1$, where T is the length of the trajectory. That is, there are no nullary or binary inputs in this case.

2.2 Relational Reasoning Layers

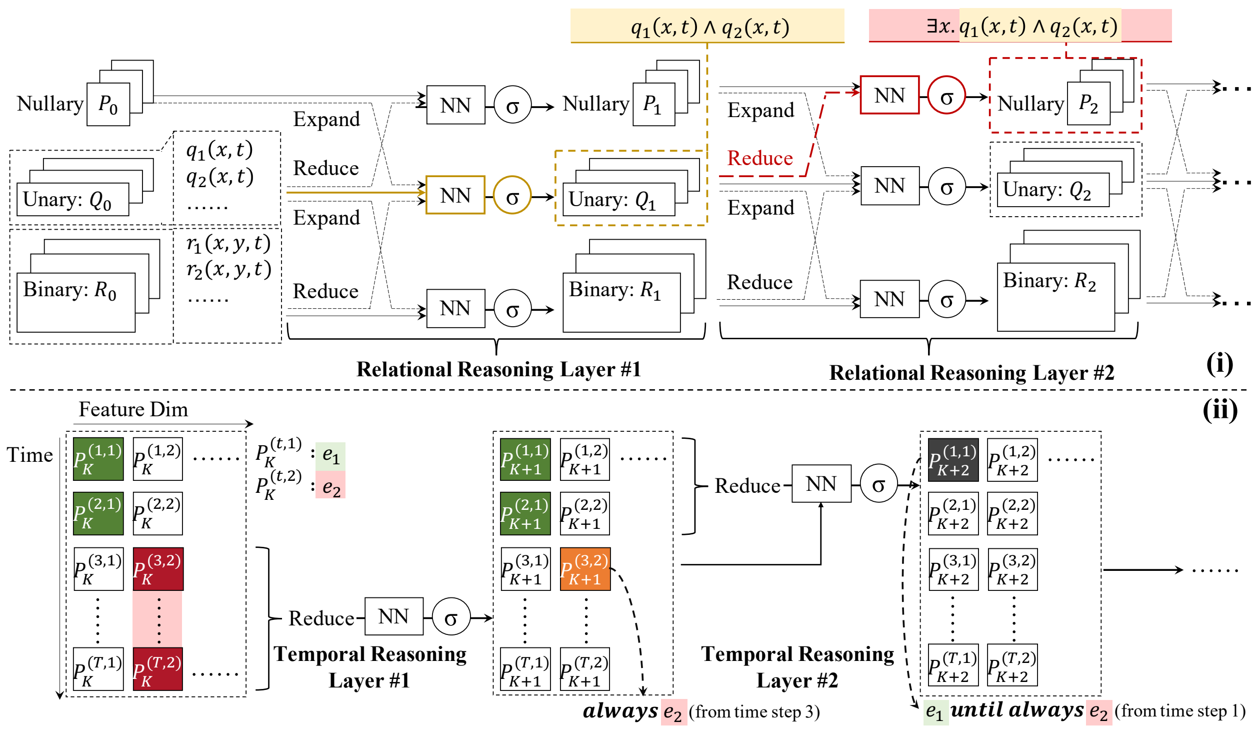
Our Relational reasoning layers (RRLs) follow prior work on Neural Logic Machines (Dong *et al.*, 2019), illustrated in Fig. 3 (i). Consider a specific time step t. At each layer l, the input to a neural logic layer is a 3-tuple $(P_{l-1}, Q_{l-1}, R_{l-1})$, which corresponds to nullary, unary, and binary features respectively. Their shapes are D_0, $N \times D_1$, and $N \times N \times D_2$. The output is another 3-tuple (P_l, Q_l, R_l), given by

$$\begin{aligned}
P_l &= \mathrm{NN}_P\left(\mathrm{Concat}\left[P_{l-1}; \max(Q_{l-1}, \mathrm{dim}=0)\right]\right), \\
Q_l &= \mathrm{NN}_Q\left(\mathrm{Concat}\left[Q_{l-1}; \max(R_{l-1}, \mathrm{dim}=0);\right.\right. \\
&\qquad \left.\left.\max(R_{l-1}, \mathrm{dim}=1); \mathrm{expand}(P_{l-1}, \mathrm{dim}=1)\right]\right), \\
R_l &= \mathrm{NN}_R\left(\mathrm{Concat}\left[R_{l-1}; \mathrm{expand}(Q_{l-1}, \mathrm{dim}=0);\right.\right. \\
&\qquad \left.\left.\mathrm{expand}(Q_{l-1}, \mathrm{dim}=1)\right]\right).
\end{aligned}$$

where NN_* are single fully-connected layers with sigmoid activations. For unary and binary features, NN_Q and NN_R are applied along the feature dimension. That is, we apply the same linear transformation to the unary features of all entities. A different linear transformation is applied to the binary features of all entity pairs. $\mathrm{Concat}[\cdot\, ;\, \cdot]$ is the concatenation operation, applied to the last dimension of the tensors (the feature dimension). $\max$, also called the "reduced" max operation, takes the maximum value along the given axis of a tensor. The expand operation, also called "broadcast," will duplicate the input tensor N times and stack them together along the given axis. RRLs are applied identically to the input features at every time step t. That is, we use the same neural network weights in a RRL for all time steps in parallel.

RRLs are motivated by relational logic rules in a finite and fully grounded universe. The $\max$ reduction operations implement a differentiable version of an existential quantifier over the finite universe of individuals, given that the truth values of the propositions are represented as values in $(0.0, 1.0)$. Because preceding and subsequent RRLs can negate propositions as needed, we omit explicit implementation of finite-domain universal quantification, although it could be added by including analogous $\min$ reductions. Thus, as illustrated in Fig. 3 (i), given input features $q_1(x,t)$ and $q_2(x,t)$, we can realize the formula $\exists x.\ q_1(x,t) \wedge q_2(x,t)$ by stacking two such layers.

Throughout the paper we have been using only nullary, unary, and binary features, but the proposed framework it-



$q_1(x,t) \wedge q_2(x,t)$
$\exists x.\, q_1(x,t) \wedge q_2(x,t)$
Nullary P_0
Unary: Q_0
$q_1(x,t)$
$q_2(x,t)$
Binary: R_0
$r_1(x,y,t)$
$r_2(x,y,t)$
Expand
Reduce
NN
σ
Nullary P_1
Unary: Q_1
Binary: R_1
Nullary P_2
Unary: Q_2
Binary: R_2
Relational Reasoning Layer #1
Relational Reasoning Layer #2
(i)
(ii)
Feature Dim
Time
$P_K^{(t,1)}$: e_1
$P_K^{(t,2)}$: e_2
Temporal Reasoning Layer #1
Temporal Reasoning Layer #2
always e_2 (from time step 3)
e_1 until always e_2 (from time step 1)

Figure 3: Illustration of (i) relational reasoning layers and (ii) temporal reasoning layers. We provide two illustrative running traces. (i) The first relational reasoning layer takes unary predicates q_1 and q_2 as input and its output Q_1 is able to represent $q_1 \wedge q_2$. The $\max(Q_1, dim = 0)$ in layer 2 can represent $\exists x.\ q_1(x,t) \wedge q_2(x,t)$. (ii) Assume P_K encodes the occurance of events e_1 and e_2 at each time step. The first temporal reasoning layer can realize *always* e_2 with a temporal pooling from time step 3 to time step T. In the second temporal reasoning layer, the temporal pooling summarizes that e_1 holds true from time step 1 to 2. Thus, the NN should be able to realize e_1 *until* (*always* e_2).

self can be easily extended to higher-order relational features. From a graph network (Bruna *et al.*, 2014; Kipf and Welling, 2017; Battaglia *et al.*, 2018) point of view, one can treat these features as the node and edge embeddings of a fully-connected graph and the relational reasoning layers as specialized graph neural network layers for realizing object quantifiers.

2.3 Temporal Reasoning Layers

Temporal reasoning layers (TRLs) perform *quantification* operations similar to relational reasoning layers, but along the *temporal* rather than the object dimension. The first TRL takes as input the summarized event representation produced by the K-th relational reasoning layer, $P_K^{(t)}$ for at all time steps t, as a matrix of shape $T \times D$. Each TRL is computed as

$$P_{K+l}^{(t)} = \max_{t'>t} \mathrm{NN}_l \left(\mathrm{Concat} \left[P_{K+l-1}^{(t')} ; \max_{t \le t'' < t'} P_{K+l-1}^{(t'')} \right] \right). \quad (1)$$

We start from inspecting each element in this formula.

1. P_{K+l-1} is the output tensor of the previous temporal reasoning layer, of shape $T \times C$, where T is the number of time steps, and C is the number of feature channels. Each entry in this tensor $P_{K+l-1}^{(t)}[i]$ can be interpreted as: event i happens at time t.
2. $\left(\max_{t \le t'' < t'} P_{K+l-1}^{(t'')}\right)$, abbreviated as $Q_{k+l-1}^{(t,t')}$ in the following text, is a vector, of shape C. Its entry $Q_{k+l-1}^{(t,t')}[i]$, where $t \le t'$, represents the concept that event i happens some time between t and t'. **Importantly,** together with the preceding and subsequent neural network operations, which can realize negation operations, it also allows us to describe the event that i holds true for all time steps between t and t'.
3. NN_l is a fully connected neural network with sigmoidal activation, which gets uniformly applied to all time steps t and a future time step $t' > t$. Its input is composed of two parts: the events that happen at t', i.e., $P_{K+l-1}^{(t)}$, and the events that happen between t and t', summarized with temporal quantification operations, i.e., $\left(\max_{t \le t'' < t'} P_{K+l-1}^{(t'')}\right)$.
4. The outer-most max pooling operation $\max_{t'>t}$ enumerates over all $t' > t$, and test whether the condition specified by NN_l holds for at least one such t'.

A special case is the first temporal reasoning layer. It takes P_K as its input, which is the output of last relational reasoning layer. Thus, the first temporal reasoning layer implements:

$$P_{K+1}^{(t)} = \mathrm{NN}_1 \left(\max_{t \le t'' < T} P_K^{(t'')} \right),$$

where T is the sequence length. Note that, there is no enumeration for a future time step $t' > t$ involved. In addition, for all

	Input		1-st Layer				2-nd Layer
t	$p(t)$	$q(t)$	$\boldsymbol{G}p$	$\boldsymbol{F}p$	$\boldsymbol{G}q$	$\boldsymbol{F}q$	$p\,\boldsymbol{U}\,(\boldsymbol{G}q)$
1	T	T	F	T	F	T	T
2	T	F	F	T	F	T	T
3	F	T	F	F	T	T	F
4	F	T	F	F	T	T	F

Table 1: A running example of different temporal quantification formulas that TOQ-Nets can realize. For clarity, we use T and F for True/False. In the actual computation, they are "soft" Boolean values ranges in $[0, 1]$. **G** means ***always***; **F** means ***Eventually***; **U** means ***until***.

Algorithm 1 An example temporal structure that the second temporal reasoning layer can recognize.

```
Input: $p(t)$, $q(t)$, $(\boldsymbol{G}p)(t)$, $(\boldsymbol{F}p)(t)$, $(\boldsymbol{G}q)(t)$, and $(\boldsymbol{F}q)(t)$
Output: $(p\boldsymbol{U}(\boldsymbol{G}q)))\,(t)$, which is true if $p$ holds true from
    time step $t$ until $q$ becomes always true.
1: for $t \leftarrow 1$ to $T$ do
2:     for $t' \leftarrow t+1$ to $T$ do
3:         if $\forall t'' \in [t, t').\ p(t'')$ and $(\boldsymbol{G}q)(t')$ then
4:             $(p\boldsymbol{U}(\boldsymbol{G}q)))\,(t) \leftarrow true$
5:         end if
6:     end for
7: end for
```

temporal layers, we add residual connections by concatenating their inputs with the outputs.

Next, let's consider a running example for the computation: how a TOQ-Net can recognize the event that: event p holds true until event q becomes always true. In LTL, this can be written as $p\boldsymbol{U}(\boldsymbol{G}q)$. Using the plain first-order logic (FOL) language, we can describe it as: $\exists\ t. [\forall t'. (0 \le t' < t) \implies p(t')) \wedge (\forall t'. (t' \ge t) \implies q(t')]$.

For simplicity, we consider a tensor representation for two events $p(t)$ and $q(t)$, where $p(t) = 1$ if it happens at time step t and $p(t) = 0$ otherwise. Given the input sequence of length 4 in Table 1 ($p(t)$ and $q(t)$), the first layer is capable of computing the following four properties for each time step t: $\boldsymbol{G}p$(***always*** p), which is true if p holds true for all future time steps starting from t, $\boldsymbol{F}p$(***eventually*** p), which is true if p is true for at least one time step starting from t, and similarly, $\boldsymbol{G}q$(***always*** q) and $\boldsymbol{F}q$(***eventually*** q). Overall, together with residual connections, the first layer can recognize six useful events: $p(t)$, $q(t)$ (from residual connection), $\boldsymbol{G}p$, $\boldsymbol{F}p$, $\boldsymbol{G}q$, and $\boldsymbol{F}q$ (by temporal quantification, i.e. pooling operations along the temporal dimension).

The second layer can realize the computation depicted in Algo 1. For every time step t, it enumerates all $t' > t$ and computes the output based on

1. the events at t' (represented as $P_{K+l-1}^{(t')}$ in Equation 1, concretely the $(\boldsymbol{G}q)(t')$ in the Algo 1 example), and
2. the state of another event between t and t' (represented as $\left(\max_{t \le t'' < t'} P_{K+l-1}^{(t'')}\right)$ in Equation 1, concretely the $\forall t'' \in [t, t').\ p(t'')$ in the Algo 1 example).

From the perspective of First-Order Linear Temporal Logic (FO-LTL), stacking multiple temporal reasoning layers enables us to realize FO-LTL formulas such as:

$$p_1\,\boldsymbol{U}\,p_2\,\boldsymbol{U}\,p_3\,\boldsymbol{U}\,\cdots\,\boldsymbol{U}\,p_k,$$

which is interpreted as p_1 holds true until p_2 becomes true and p_2 holds true from that until p_3 becomes true $\cdots$, and

$$\boldsymbol{F}p_1\,\boldsymbol{XF}\,p_2\,\boldsymbol{XF}\,p_3\,\boldsymbol{XF}\,\cdots\,\boldsymbol{XF}\,p_k,$$

which is interpreted as p_1 eventually becomes true and after that p_2 eventually becomes true and after that $\cdots$, and in addition, formulas with interleaved until and eventually quantifiers. Here, ***XF*** is a composition of the ne***X***t operator and the ***F***inally operator in LTL. Meanwhile, as described so far, TOQ-Nets can only nest object quantification inside temporal quantification, so it can represent ***always*** $\exists x.\ q_1(x) \wedge q_2(x)$, but not $\exists x.$ ***always*** $q_1(x) \wedge q_2(x)$. This can be solved by interleaving relational and temporal reasoning layers.

It is important to notice that, by using object and temporal pooling operations together with trainable neural networks to realize logic formulas with object and temporal quantifiers, the idea itself generalizes to a broader set of FO-LTL formulas. We design TOQ-Nets to model only a subset of FO-LTL formulas, because they can be computed efficiently (with only $O(T^2)$ space) and they are expressive enough for the type of data we are trying to model.

3 Experiments

We compare our model with other approaches to object-centric temporal event detection in this section, and include an application of TOQ-Nets to concept learning over robot object-manipulation trajectories in the supplementary material. The setups and metrics focus on data efficiency and generalization.

3.1 Baseline approaches

We compare TOQ-Nets against five baselines. The first two are spatial-temporal graph convolutional neural networks (STGAN; Yan *et al.*, 2018) and its variant STGCN-MAX, which models entity relationships with graph neural networks and models temporal structure with temporal-domain convolutions. The third is STGCN-LSTM, which uses STGCN layers for entity relations but LSTM (Hochreiter and Schmidhuber, 1997) for temporal structures. The last two baselines are based on space-time graphs: Space-Time Graph (Wang and Gupta, 2018) and Non-Local networks (Wang *et al.*, 2018). We provide details about our implementation and how we choose the model configurations in the supplementary material.

3.2 Trajectory-Based Soccer Event Detection

We start our evaluation with an event-detection task in soccer games. The task is to recognize the action performed by a specific player at specific time step in a soccer game trajectory.

Dataset and setup. We collect training and evaluation datasets based on the gfootball simulator[*], which provides

[*]https://research-football.dev/

Model	Reg.	Few-Shot	Full
STGCN	$73.2_{\pm 1.6}$	$26.0_{\pm 5.7}$	$62.8_{\pm 0.6}$
STGCN-MAX	$73.6_{\pm 1.5}$	$28.6_{\pm 5.0}$	$63.6_{\pm 0.7}$
STGCN-LSTM	$72.7_{\pm 1.4}$	$23.8_{\pm 5.9}$	$61.9_{\pm 0.6}$
Space-Time	$74.8_{\pm 1.5}$	$31.7_{\pm 6.1}$	$65.2_{\pm 0.6}$
Non-Local	$76.5_{\pm 2.4}$	$45.0_{\pm 6.3}$	$69.5_{\pm 2.4}$
TOQ-Net (ours)	**87.7**$_{\pm 1.3}$	**52.2**$_{\pm 6.3}$	**79.8**$_{\pm 0.8}$

Table 2: Results on the soccer event dataset. Different columns correspond to different action sets (the regular, few-shot, and full action sets). The performance is measured by per-action (macro) accuracy, averaged over nine few-shot splits. The ± values indicate standard errors. TOQ-Net significantly outperforms all baseline methods on the few-shot action set.

a physics-based 3D football simulation. It also provides AI agents that can be used to generate random plays. The simulator provides the 3D coordinates of the ball and the players as well as the action each player is performing at each time step. There are in total 13 actions defined in the simulator, including *movement, ball_control, trap, short_pass, long_pass, high_pass, header, shot, deflect, catch, interfere, trip* and *sliding*. We exclude *header* and *catch* actions, as they never appear in AI games. We also exclude *ball_control* and *movement*, since they just mean the agent is moving (with or without the ball). Thus, in total, we have nine action categories. We run the simulator with AI-controlled players to generate plays, and formulate the task as classifying the action (9-way classification) of a specific player at a specific time step given a temporal context (25 frames). For each action, we have generated 5,000 videos, expect for *sliding*, for which we generated 4,000 videos because it is rare in the AI games. Among the generated examples, 62% (2,462 or 3,077) are used for training, 15% are used for validation, and 23% are used for testing. Each trajectory is an 8-fps replay clip that contains 17 frames (about two seconds). There is a single "target" player in each trajectory. The action label of the trajectory is the action performed by this target player at frame #9. We randomly split all actions into two categories: seven "regular" actions, for which all game plays are available, and two "few-shot" actions, for which only 50 clips are available during training.

Input features. Each trajectory is represented with 7 time-varying unary predicates, including the 3D coordinate of each player and the ball and four extra predicates defining the type of each entity x: $\mathrm{IsBall}(x)$, $\mathrm{IsTargetPlayer}(x)$, $\mathrm{SameTeam}(x)$, $\mathrm{OpponentTeam}(x)$, where $\mathrm{SameTeam}(x)$ and $\mathrm{OpponentTeam}(x)$ indicates whether x is of the same team as the target player. We also add a temporal indicator function which is a Gaussian function centered at frame of interest with variance $\sigma^2 = 25$.

Results. Table 2 shows the result. Our model significantly outperforms all the baselines in all three action settings, suggesting that our model is able to discover a set of useful features at both input and intermediate levels and use them to compose new action classifiers from only a few examples.

Generalization to more players. Due to their object-centric design, TOQ-Nets can generalize to soccer games with a varying number of agents. After training on 6v6 soccer games (i.e., 6 players on each team), we evaluate the performance of different models on games with different numbers of players: 3v3, 4v4, 8v8, and 11v11. For each action we have generated, on average, 1,500 examples for testing. Table 3 summarizes the result and the full results are provided in the supplementary material. Comparing the columns highlighted in yellow, we notice a significant performance drop for all baselines while TOQ-Net performs the best. By visualizing data and predictions, we found that misclassifications of instances of *shot* as *short pass* contribute most to the performance degradation of our model when we have more players. Specifically, the recall of *shot* drops from 97% to 60%. In soccer plays with many agents, a shot is usually unsuccessful and a player from another team steals the ball in the end. In such scenarios, TOQ-Net tends to misclassify such trajectories as a *short pass*. Ideally, this issue should be addressed by understanding actions based on agents' goals instead of the actual outcome (Intille and Bobick, 2001). We leave this extension as a future direction.

Generalization to temporally warped trajectories. Another crucial property of TOQ-Nets is to recognize actions based on their sequential order in the input trajectory, instead of binding features to specific time steps in the trajectory. To show this, we test the performance of different models on time warped trajectories. Each test trajectory has a length of 25, and each trajectory is labeled by the action performed by the target player at any time step between the 6-th and 19-th frame. We ensure that the target player performs only one action during the entire input trajectory. Thus, the label is unambiguous. The results are shown in Table 3. Specifically, our test set consists of 25-frame trajectories, and the action may occur at anytime between the 6th and the 19th frame. By comparing rows with and without time warping, we notice a 60% performance drop for STGCN, STGCN-MAX, and STGCN-LSTM. In contrast, TOQ-Nets still have reasonable performance. Note that Space-Time and Non-Local model have almost no performance drop against time warping because they are completely agnostic to temporal ordering.

3.3 Extension to Real-World Datasets

The proposed TOQ-Net can also be extended to other real-world datasets. These examples further illustrate the robustness of TOQ-Net to temporal variations in activities.

Toyota Smarthome (Das *et al.*, 2019). Toyota Smarthome is a dataset that contains videos of humans performing everyday activities such as "walk", "take pills", and "use laptop". It also comes with 3D-skeleton detections. There are around 16.1k videos in the dataset, and 19 activity classes. The videos' length varies between a few seconds to 3 minutes. We subsample 30 frames for each video. We split frames into training (9.9k), validation (2.5k), and testing (3.6k). We treat human joints as entities. The input is then the position of joints, the velocity of joints, limb lengths, and joint angles. We evaluated our model and STGCN on a 19-way classification task. We also test model performance on time-warped sequences by accelerating the trajectories by two times.

Our model achieves a comparable accuracy to STGCN on the standard classification task: (42.0% vs. 43.0%). Impor-

Model	3v3	4v4	6v6	6v6 (Time Warp)	8v8	11v11
STGCN	$40.7_{\pm 1.0}$ (-40.4%)	$63.2_{\pm 4.9}$ (-7.4%)	$68.2_{\pm 2.8}$	$52.8_{\pm 7.0}$ (-22.6%)	$55.4_{\pm 3.3}$ (-18.8%)	$44.4_{\pm 2.1}$ (-34.9%)
STGCN-MAX	$47.4_{\pm 3.2}$ (-33.7%)	$68.8_{\pm 2.0}$ (-3.8%)	$71.5_{\pm 1.9}$	$56.5_{\pm 4.5}$ (-21.0%)	$59.1_{\pm 0.7}$ (-17.3%)	$45.6_{\pm 2.5}$ (-36.2%)
STGCN-LSTM	$39.7_{\pm 1.1}$ (-43.1%)	$60.4_{\pm 0.2}$ (-13.5%)	$69.8_{\pm 0.1}$	$30.6_{\pm 0.6}$ (-56.1%)	$55.8_{\pm 2.0}$ (-20.0%)	$44.1_{\pm 0.7}$ (-36.8%)
Space-Time	$29.0_{\pm 1.6}$ (-60.4%)	$53.5_{\pm 3.2}$ (-27.0%)	$73.3_{\pm 0.3}$	$70.7_{\pm 0.3}$ (-3.5%)	$33.9_{\pm 2.8}$ (-53.7%)	$15.2_{\pm 1.8}$ (-79.3%)
Non-Local	$45.9_{\pm 5.1}$ (-41.2%)	$70.7_{\pm 5.3}$ (-9.5%)	$78.1_{\pm 5.8}$	$77.7_{\pm 5.0}$ (-0.5%)	$58.5_{\pm 10.8}$ (-25.1%)	$41.8_{\pm 13.6}$ (-46.5%)
TOQ-Net	**77.4**$_{\pm 3.5}$ **(-12.4%)**	**88.3**$_{\pm 0.7}$ **(-0.0%)**	**88.4**$_{\pm 0.6}$	**86.9**$_{\pm 0.4}$ **(-1.7%)**	**81.3**$_{\pm 1.7}$ **(-8.0%)**	**77.1**$_{\pm 1.7}$ **(-12.8%)**

Table 3: Results on generalization to scenarios with more agents and temporally warped trajectories on the soccer event dataset. All models are trained only on 6v6 games. The standard errors indicated by the ± signs are computed with three random seeds.

tantly, on the generalization test to time-warped sequences, our model has only a 0.8% performance drop (41.2%), while STGCN drops 10.7% (32.3%). This indicates that the temporal structures learned by TOQ-Net improve model generalization to varying time courses.

Volleyball Activity (Ibrahim *et al.*, 2016). The volleyball dataset contains 4830 video clips collected from 55 youtube volleyball videos. They are labeled with 8 group activities (e.g. "left spike" and "right pass"). Each video contains 20 frames with the labeled group activity performed at the 10-th frame. The dataset also includes annotations for players, including the bounding box, the indicator of whether the player is involved in the group activity, and the individual action such as "setting", "digging", and "spiking". We use the manual annotations (processed by an MLP) as the input features. We train models to classify the video into one of the eight group activities, following the original split, i.e., 24, 15, and 16 of 55 videos are used for training, validation, and testing.

On the standard classification task, TOQ-Net achieves a comparable performance with STGCN (73.3% vs. 73.6%). When we perform time warping on the input sequences, STGCN's performance drops by more than 25.0% (39.5% on temporally shifted trajectories and 48.6% on 2× quick motion trajectories), while our model drops only 3% (70.3% on temporally shifted trajectories and 70.7% on quick motion trajectories). This again shows the generalization ability of TOQ-Net w.r.t. varying time courses, and the robustness of learned temporal structures.

4 Related Work

Action concept representations and learning. First-order and linear temporal logics (LTL; Pnueli, 1977) have been used for analyzing sporting events (Intille and Bobick, 1999, 2001) and activities of daily living (Tran and Davis, 2008; Brendel *et al.*, 2011) in logic-based reasoning frameworks. However, these frameworks require extra knowledge to annotate relationships between low-level, primitive actions and complex ones, or performing search in a large combinatorial space for candidate temporal logic rules (Penning *et al.*, 2011; Lamb *et al.*, 2007). By contrast, TOQ-Nets enable end-to-end learning of complex action descriptions from sensory input with only high-level action-class labels.

Temporal and relational reasoning. This paper is also related to work on using data-driven models for modeling relational and temporal structure, such as LTL (Neider and Gavran, 2018; Camacho *et al.*, 2018; Chou *et al.*, 2020), Logical Neural Networks (Riegel *et al.*, 2020), ADL description languages (Intille and Bobick, 1999), and hidden Markov models (Tang *et al.*, 2012). These models need human-annotated action descriptions and symbolic state variables (e.g., *pick up* x means a state transition from not *holding* x to *holding* x), and dedicated inference algorithms such as graph structure learning. In contrast, TOQ-Nets have an end-to-end design, and can be integrated with other neural networks. People have also used structural representations to model object-centric temporal concepts with graph neural networks (GNNs; Yan *et al.*, 2018), recurrent neural networks (RNNs; Ibrahim *et al.*, 2016), and integrated GNN-RNN architectures (Deng *et al.*, 2016; Qi *et al.*, 2018). TOQ-Nets use a similar relational representation, but different models for temporal structures.

5 Conclusion and Discussion

The design of TOQ-Nets suggests multiple research directions. For example, the generalization of the acquired action concepts to novel object kinds, such as from *opening fridges* to *opening bottles*, needs further exploration. Meanwhile, TOQ-Nets are based on physical properties, e.g. 6D poses. Incorporating representations of mental variables such as goals, intentions, and beliefs can aid in action and event recognition (Baker *et al.*, 2017; Zacks *et al.*, 2001; Vallacher and Wegner, 1987).

In summary, we have presented TOQ-Nets, a neuro-symbolic architecture for learning to describe complex state sequences with quantification over both entities and time. TOQ-Nets use tensors to represent the time-varying properties and relations of different entities, and use tensor pooling operations over different dimensions to realize temporal and object quantifiers. TOQ-Nets generalize well to scenarios with varying numbers of entities and time courses.

Acknowledgements

We thank Tomás Lozano-Pérez for helpful discussions and suggestions. We gratefully acknowledge support from the Center for Brains, Minds and Machines (NSF STC award CCF1231216), ONR MURI N00014-16-1-2007, NSF grant 1723381, AFOSR grant FA9550-17-1-0165, ONR grant N00014-18-1-2847, the Honda Research Institute, MIT-IBM Watson Lab, SUTD Temasek Laboratories, the Samsung Global Research Outreach (GRO) Program, and the Stanford Institute for Human-Centered AI (HAI). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of our sponsors.

References

Chris L Baker, Julian Jara-Ettinger, Rebecca Saxe, and Joshua B Tenenbaum. Rational quantitative attribution of beliefs, desires and percepts in human mentalizing. *Nature Human Behaviour*, 1(4):1–10, 2017.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv:1806.01261*, 2018.

William Brendel, Alan Fern, and Sinisa Todorovic. Probabilistic event logic for interval-based event recognition. In *CVPR*, 2011.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. In *ICLR*, 2014.

Alberto Camacho, Jorge Baier, Christian Muise, and Sheila McIlraith. Finite ltl synthesis as planning. In *ICASP*, 2018.

Glen Chou, Necmiye Ozay, and Dmitry Berenson. Explaining multi-stage tasks by learning temporal logic formulas from suboptimal demonstrations. In *RSS*, 2020.

Srijan Das, Rui Dai, Michal Koperski, Luca Minciullo, Lorenzo Garattoni, Francois Bremond, and Gianpiero Francesca. Toyota smarthome: Real-world activities of daily living. In *ICCV*, 2019.

Zhiwei Deng, Arash Vahdat, Hexiang Hu, and Greg Mori. Structure inference machines: Recurrent neural networks for analyzing relations in group activity recognition. In *CVPR*, 2016.

Honghua Dong, Jiayuan Mao, Tian Lin, Chong Wang, Lihong Li, and Denny Zhou. Neural logic machines. In *ICLR*, 2019.

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

Mostafa S. Ibrahim, Srikanth Muralidharan, Zhiwei Deng, Arash Vahdat, and Greg Mori. A hierarchical deep temporal model for group activity recognition. In *CVPR*, 2016.

Stephen S Intille and Aaron F Bobick. A framework for recognizing multi-agent action from visual evidence. In *AAAI*, 1999.

Stephen S Intille and Aaron F Bobick. Recognizing planned, multi-person action. *CVIU*, 81(3):414–445, 2001.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

Luis C. Lamb, Rafael V. Borges, and A. d'Avila Garcez. A connectionist cognitive model for temporal synchronisation and learning. In *AAAI*, pages 827–832, 2007.

Daniel Neider and Ivan Gavran. Learning linear temporal properties. In *FMCAD*, 2018.

Leo de Penning, A. d'Avila Garcez, Luís C. Lamb, and John-Jules C. Meyer. A neural-symbolic cognitive agent for online learning and reasoning. In *IJCAI*, pages 1653–1658, 2011.

Amir Pnueli. The temporal logic of programs. In *FOCS*, 1977.

Mengshi Qi, Jie Qin, Annan Li, Yunhong Wang, Jiebo Luo, and Luc Van Gool. stagnet: An attentive semantic rnn for group activity recognition. In *ECCV*, 2018.

Ryan Riegel, Alexander Gray, Francois Luus, Naweed Khan, Ndivhuwo Makondo, Ismail Yunus Akhalwaya, Haifeng Qian, Ronald Fagin, Francisco Barahona, Udit Sharma, Shajith Ikbal, Hima Karanam, Sumit Neelam, Ankita Likhyani, and Santosh Srivastava. Logical neural networks, 2020.

Jürgen Stränger and Bernhard Hommel. The perception of action and movement. In *Handbook of perception and action*, volume 1, pages 397–451. Elsevier, 1996.

Kevin Tang, Li Fei-Fei, and Daphne Koller. Learning latent temporal structure for complex event detection. In *CVPR*, 2012.

Son D Tran and Larry S Davis. Event modeling and recognition using markov logic networks. In *ECCV*, 2008.

Robin R Vallacher and Daniel M Wegner. What do people think they're doing? action identification and human behavior. *Psychol. Rev.*, 94(1):3, 1987.

Xiaolong Wang and Abhinav Gupta. Videos as space-time region graphs. In *ECCV*, 2018.

Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018.

Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *AAAI*, 2018.

Jeffrey M Zacks, Barbara Tversky, and Gowri Iyer. Perceiving, remembering, and communicating structure in events. *J. Exp. Psychol. Gen.*, 130(1):29, 2001.

Jeffrey M Zacks, Nicole K Speer, Khena M Swallow, Todd S Braver, and Jeremy R Reynolds. Event perception: a mind-brain perspective. *Psychol. Bull.*, 133(2):273, 2007.