Don't Pour Cereal into Coffee: Differentiable Temporal Logic for Temporal Action Segmentation

Ziwei Xu[†*] **Yogesh S Rawat**[§] **Yongkang Wong**[†] **Mohan S Kankanhalli**[†] **Mubarak Shah**[§]
† School of Computing, National University of Singapore
§ Center for Research in Computer Vision, University of Central Florida
{ziwei-xu,mohan}@comp.nus.edu.sg yongkang.wong@nus.edu.sg
{yogesh,shah}@crcv.ucf.edu

Abstract

We propose Differentiable Temporal Logic (DTL), a model-agnostic framework that introduces temporal constraints to deep networks. DTL treats the outputs of a network as a truth assignment of a temporal logic formula, and computes a temporal logic loss reflecting the consistency between the output and the constraints. We propose a comprehensive set of constraints, which are implicit in data annotations, and incorporate them with deep networks via DTL. We evaluate the effectiveness of DTL on the temporal action segmentation task and observe improved performance and reduced logical errors in the output of different task models. Furthermore, we provide an extensive analysis to visualize the desirable effects of DTL.

Figure 1: A video of activity "coffee preparation". The colored bars, from the top to the bottom, show the ground truth (GT), the predictions from a baseline [15], and the predictions from the baseline trained with DTL, respectively. Note that the baseline model erroneously predicts pour_cereal when DTL solves this problem with an exclusivity constraint between pour_coffee and pour_cereal.

1 Introduction

Recent years have witnessed significant advances in video action analysis tasks such as action recognition [29, 58], action detection [52], and temporal action segmentation [15, 1]. This can be credited to the availability of large-scale datasets [50, 23] and the development of effective deep visual backbones [4, 53, 17]. Although data-driven backbones can capture useful spatio-temporal features, learning a large number of highly diverse temporal dependencies and correlations over long time spans can be very challenging. In existing approaches, these temporal dependencies are not explicit to the model: it is possible to provide framewise annotation during training, however, temporal constraints like "event X has to occur after event Y" are still implicit in the annotations and not explicitly enforced. Even though these constraints could be statistically learned from a large amount of data, a pure vision model could still be confused by different actions with similar visual

[*]This work was done when Ziwei Xu was visiting the Center for Research in Computer Vision.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

appearances. An example of this is shown in Fig. 1, where the model confuses pour_cereal with pour_milk because both actions involve holding a carton.

In this paper, we propose a solution to this problem by employing temporal logic to apply declarative temporal constraints to the output of deep networks. Linear Temporal Logic (LTL) [45], for example, defines operators that describe necessities, possibilities, and dependencies of actions in a series of actions. Checking a network's output against those constraints (a.k.a. model checking) tells us if the predicted actions are logically correct. Logical correctness can be used as an additional training objective. According to prior works in the neuro-symbolic community [57, 55, 18], applying logic constraints improves the performance of deep networks in tasks for graph data [57] and images [55] by reducing logical errors in the output. Although one would anticipate similar benefits when using the temporal logic for action analysis, most of the focus in literature is still on non-sequential data.

In this work, we focus on the temporal aspect and consider temporal constraints in videos. Inspired by the foundational works in logic-based losses [57, 18], we propose a Differentiable Temporal Logic (DTL) framework, which uses an extended definition of linear temporal logic to constrain the output of action analysis models. At a high level, DTL treats the model output as a truth assignment of variables in LTL formulae. A differentiable evaluator performs model checking on the outputs and yields a satisfaction score, which measures the consistency between the constraints and the outputs. As we optimize the satisfaction score through standard optimization methods, the constraints are enforced on the deep network. Different from existing work [56], in DTL the relation between formula evaluation and formula satisfaction is determined, which means that DTL is logically sound. Moreover, we propose a comprehensive set of constraints covering both temporal and non-temporal dependencies between actions, and show how they are represented using DTL. We evaluate DTL on the challenging temporal action segmentation task, where modeling temporal dependencies between actions is crucial. The efficacy of our method is shown by the improved performance of different task models when constraints are enforced through DTL.

The contributions of this work are as follows:

- We present Differentiable Temporal Logic (DTL), a framework providing a model-agnostic manner of introducing temporal logic constraints to deep networks (cf. Section 3.2).
- We propose DTL constraints to describe a wide range of relations between actions, which can be automatically procured from dataset annotation (cf. Section 3.3).
- Experiments with different task models on the temporal action segmentation task show the efficacy of DTL. In addition, we provide an extensive study to show the effect of DTL constraints on task models.

2 Related Works

2.1 Temporal Action Segmentation

A temporal action segmentation model takes a set of video frames as input and predict the action category for each frame. The model needs to capture both short and long term dependencies between action categories. Earlier methods [3, 16] use sliding windows to model changes in visual appearance in a short time. Long-term dependencies are captured from those short-term information by sequential models like hidden Markov model [34] and recurrent networks [35, 47]. Despite the remarkable performance, when these models are used to process long videos they still face forgetting issues and heavy computation burden. Temporal Convolutional Network (TCN) [37, 38, 15, 39] enables efficient modelling of temporal dependency in variable time spans with its flexible receptive field. Many advancements are made based on MSTCN [15], which used stacked dilated temporal convolution layers for temporal modelling. For example, ASRF [28] uses a boundary regression task on top of MSTCN to improve the quality of segmentation. Huang *et al.* [26] proposed a graph-based dependency model to improve the modeling of relations between actions. Finally, Transformer [54] is introduced to this task and shows superior performance in [59]. Our view is that these methods can be complemented by DTL, since many declarative constraints are difficult to learn from the data. On the other hand, DTL can explicitly enforce these constraints on the task model during training.

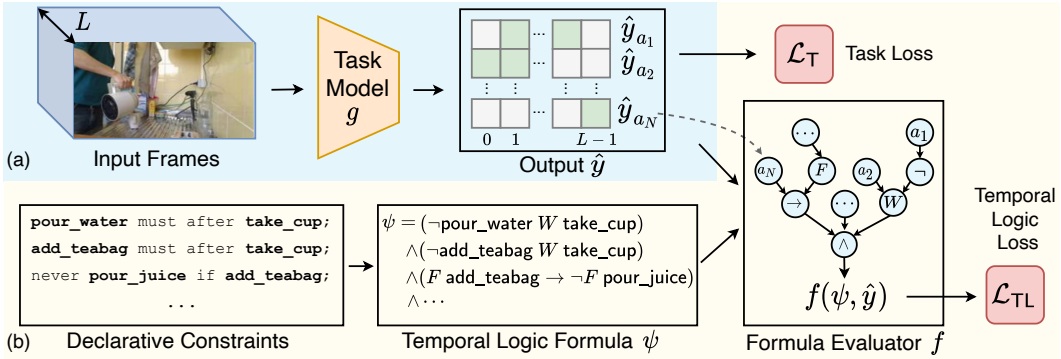

Figure 2: An overview of DTL framework. (a) shows a typical action segmentation pipeline, where a task model g takes input frames and output segmentation $\hat{\boldsymbol{y}}_*$ for each action class. A task loss $\mathcal{L}_{\mathsf{T}}$ is minimized during training. (b) shows the logic evaluation part of DTL. A set of declarative constraints on actions are converted into a temporal logic formula ψ. A formula evaluator f takes ψ and $\hat{\boldsymbol{y}}_*$ as input, and evaluates the consistency between the two, resulting in a logic loss $\mathcal{L}_{\mathsf{TL}}$.

2.2 Temporal Logic and Action Analysis

Temporal logic is a large family of logic systems that specify the relations between timed events. Linear temporal logic (LTL) [45] is one of the earliest temporal logic. It is a propositional modal logic that states necessity and possibility about events in a linear discrete succession of time steps. Later variants of LTL include Metric Temporal Logic (MTL) [41], which introduces time units as extra parameters to temporal operators, and Signal Temporal Logic [14], which specifies temporal properties for continuous signals. Allen's interval algebra [2] and Interval Temporal Logic (ITL) [22] specify the relations between intervals and their composition. Temporal logic has been used to specify constraints for program verification [45], robot control [13, 27, 46], and linguistics [42]. However, its application in action analysis remains scarce. CASE^E [21] uses ITL as a structural language to describe atomic actions for event classification. T-LEAF [56] is the first (and so far the only, to the best of our knowledge) attempt to apply temporal logic to action analysis. It uses an embedding model to incorporate LTL formulae when training deep networks for action classification, which has little guarantee on logical soundness and only handles a small set of constraints.

DTL adopts LTL for temporal constraints because LTL is: (1) more feasible than its successors with more operators and (2) expressive enough to benefit action analysis. In principle, DTL can be used with any logic system that is sound and whose formulae can be evaluated in a differentiable manner.

2.3 Logic Constraints on Neural Networks

Applying symbolic constraints to deep networks has been drawing increasing interest [25, 11, 40, 48, 20], yet it remains an open problem. An important challenge is that logic propositions generally take discrete truth values, while end-to-end differentiability is usually required to optimize a network. A workaround for this challenge is to find a continuous consistency measurement between network outputs and constraints. Existing methods can be categorized into two branches based on whether constraints are treated as data or procedures. The first branch is embedding-based [36, 30, 55, 56, 43], which treats constraints and network outputs as data and project them into a continuous space, where the distance between the two is used as a measurement. For example, LENSR [55] represents logical constraints in Deterministic Decomposable Negation Normal Form (d-DNNF) [8] and uses GCN [32] as the embedder. The second branch is procedure-based [57, 18, 19, 12, 24], which compiles constraints into computational procedures over network outputs and uses the result as consistency measurement. For example, semantic loss [57] compiles formulae into logic circuits [9], along which the probability of the output satisfying the formula is computed as the measurement. We design DTL as a procedure-based framework instead of an embedding-based one, since the latter generally lacks a soundness guarantee: high similarities in the embedding space does not guarantee logical correctness (or satisfaction).

3 Method

We consider the action segmentation task in a video clip with L frames and N candidate actions $\mathcal{A} = \{a_1, \ldots, a_N\}$. As shown in Fig. 2, we assume that there is a deep task model g parameterized by $\boldsymbol{\Theta}$, which takes frames $\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{L-1}$ as input, and outputs $\hat{\boldsymbol{y}}_{a_1}, \ldots, \hat{\boldsymbol{y}}_{a_N}$, where $\hat{\boldsymbol{y}}_{a_i} \in \mathbb{R}^L$ is the unnormalized score for the presence of action a_i throughout the L frames. We assume $\hat{\boldsymbol{y}}_{a_i,t} > 0$ indicates a_i presents at time t, and $\hat{\boldsymbol{y}}_{a_i,t} \leq 0$ indicates the opposite. Apart from the ground truth label $\boldsymbol{y}_{a_1}, \ldots, \boldsymbol{y}_{a_N}$, there is a set of constraints that describe the relation between actions. The design goal of DTL is to incorporate these constraints into the task model g, so that $\hat{\boldsymbol{y}}$ complies with both the ground truth $\boldsymbol{y}$ and the constraints. In order to do so, we introduce a temporal logic representation Ψ, and an evaluator f that enforces the constraints in formula $\psi \in \Psi$ on g, through its outputs $\hat{\boldsymbol{y}}$.

3.1 Syntax of Formulae

The definition of Ψ is an extension of Linear Temporal Logic (LTL) [45]. A formula $\psi \in \Psi$ takes any of the forms separated by "|" below:

$$\psi := \mathsf{True} \mid \mathsf{False} \mid a \mid \neg\psi_1 \mid (\psi_1 \wedge \psi_2) \mid (\psi_1 \vee \psi_2) \mid \mathrm{X}\psi_1 \mid \mathrm{F}\psi_1 \mid (\psi_1 \mathrm{W} \psi_2) \mid (\psi_1 \mathrm{S} \psi_2), \quad (1)$$

where $a \in \mathcal{A}$ is the atomic proposition, and $\psi_1, \psi_2 \in \Psi$. The connectives X (NEXT), F (EVENTUAL), W (WEAK_UNTIL), and S (SINCE) are modal operators that form the temporal relations between propositions. In our context, atomic propositions a represents action a. Note that the definition in Eqn. (A1) is recursive. For example, if $a_1 \in \Psi$, then $\mathrm{X} \ldots \mathrm{XF} a_1 \in \Psi$. Semantically, $\mathrm{X}\psi$ is satisfied when proposition ψ is satisfied in the next time step. $\mathrm{F}\psi$ is satisfied when ψ is satisfied by the end of the sequence. $\psi_1 \mathrm{W} \psi_2$ being satisfied means that ψ_1 must always be satisfied until ψ_2 is satisfied (and ψ_2 might never be satisfied). $\psi_1 \mathrm{S} \psi_2$ being satisfied means ψ_1 is always satisfied after ψ_2 is satisfied. Section D.1 provides a more formal definition of these operators.

3.2 Formula Evaluator

A formula is said to be **satisfied** by a truth value assignment if the assignment is semantically compliant with the constraints. In the context of this paper, each atomic proposition a_i is assigned $\hat{\boldsymbol{y}}_{a_i}$. Satisfiability is determined by evaluation, which is a function f of a formula ψ and $\hat{\boldsymbol{y}}$:

$$f_t(\psi, \hat{\boldsymbol{y}}) = f(\psi, \hat{\boldsymbol{y}}_{a_1:a_N, t:L-1}) : \Psi \times \underbrace{\mathbb{R}^{L-t} \times \cdots \times \mathbb{R}^{L-t}}_{N \text{ times}} \to \mathbb{R}, \quad (2)$$

where $\times$ refers to the Cartesian product. The two arguments of f are formula ψ and model output $\hat{\boldsymbol{y}}_{a_1:a_N}$. Parameter $t \in [0, L-1]$ is the start time of the evaluation. For example, $t = 2$ means that the evaluation is between ψ and the prediction starting from the third frame, i.e. $\hat{\boldsymbol{y}}_{a_1:a_N, 2:L-1}$. The result of f is a satisfaction score that measures the consistency between $\hat{\boldsymbol{y}}$ and ψ.

Eqn. (2) is abstract and must be detailed for all possible forms of a formula ψ can take in Eqn. (A1). We aim to expand the definition so that Ψ is logically sound, i.e. there is a determined relation between $f_t(\psi, \hat{\boldsymbol{y}})$ and the satisfaction of ψ given $\hat{\boldsymbol{y}}$. Specifically, we would like $f_t(\psi, \hat{\boldsymbol{y}}) > 0$ to imply that ψ is satisfied by $\hat{\boldsymbol{y}}$ at time t. Moreover, Ψ must be differentiable to be incorporated with task models. Towards these goals, we first define the evaluation for constants and atomic propositions:

$$f_t(\mathsf{True}, \hat{\boldsymbol{y}}) = +\infty, \quad f_t(\mathsf{False}, \hat{\boldsymbol{y}}) = -\infty, \quad f_t(a, \hat{\boldsymbol{y}}) = \hat{\boldsymbol{y}}_{a,t}. \quad (3)$$

Indeed, the evaluation result for True and False will always be positive and negative, because the former is always satisfied and the latter is never satisfied. Note that if $\psi = a$, it is satisfied at time t when $\hat{\boldsymbol{y}}_{a,t} > 0$, i.e. action a happens at time t.

Next, we define the evaluation for operators "$\neg$" (negation), "$\wedge$" (logical and), and "$\vee$" (logical or):

$$f_t(\neg\psi_1, \hat{\boldsymbol{y}}) = -f_t(\psi_1, \hat{\boldsymbol{y}}), \quad (4)$$

$$f_t(\psi_1 \wedge \psi_2, \hat{\boldsymbol{y}}) = \min{}^{\gamma}\big\{f_t(\psi_1, \hat{\boldsymbol{y}}), f_t(\psi_2, \hat{\boldsymbol{y}})\big\}, \quad (5)$$

$$f_t(\psi_1 \vee \psi_2, \hat{\boldsymbol{y}}) = \max{}^{\gamma}\big\{f_t(\psi_1, \hat{\boldsymbol{y}}), f_t(\psi_2, \hat{\boldsymbol{y}})\big\}, \quad (6)$$

where γ is a parameter of function $\min^{\gamma}\{x_{1:L-1}\} = -\frac{1}{\gamma} \ln \sum_{i=1}^{L-1} e^{-\gamma x_i}$, which approximates the minimum value of $\{x_{1:L-1}\}$ [6], and $\max^{\gamma}\{x_{1:L-1}\} = -\min^{\gamma}\{-x_{1:L-1}\}$. It can be shown [6]

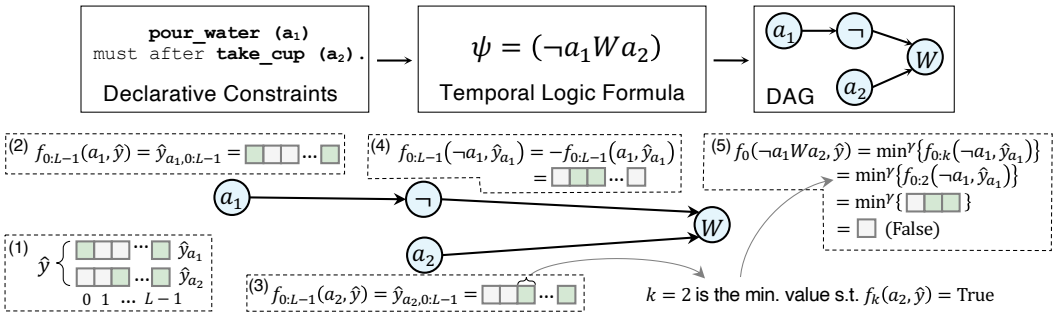

Figure 3: The evaluation of an example constraint on two actions over L time steps. The constraint is first written as formula $\psi = (\neg a_1 \mathrm{W} a_2)$, then represented as a DAG. (1) – (5) show how $f_0(\neg a_1 \mathrm{W} a_2, \hat{\boldsymbol{y}})$ is computed. (1) shows the output $\hat{\boldsymbol{y}}$ of task model, where green boxes indicate positive values (True) and grey boxes indicate negative values (False). (2) and (3) show the evaluation of leaf nodes a_1 and a_2, which is the start of evaluation. (4) takes the evaluation results for a_1 from (2) and evaluate $\neg a_1$ following Eqn. (4). In (5), the W node uses the result for $\neg a_1$ from (4) and a_2 from (3) to perform the evaluation for $(\neg a_1 \mathrm{W} a_2)$ following Eqn. (9).

that $\lim_{\gamma\to\infty} \min^\gamma\{x_{1:L-1}\} = \min\{x_{1:L-1}\}$. In Eqn. (4), the operator $\neg$ flips the sign of $f_t(\psi_1, \hat{\boldsymbol{y}})$, reflecting the negation semantics of $\neg$. In Eqn. (5), $\min^\gamma\{f_t(\psi_1, \hat{\boldsymbol{y}}), f_t(\psi_2, \hat{\boldsymbol{y}})\}$ will be negative (False) if ψ_1 or ψ_2 or both are False, and will be positive (True) iff. both ψ_1 and ψ_2 are True. This is consistent with the semantics of $\wedge$. The same rationale applies to $\max^\gamma$ for $\vee$ in Eqn. (6).

Finally, we define the modal operators X, F, W, and S that are unique in Ψ. Intuitively, evaluating $\mathrm{X}\psi_1$ is equivalent to evaluating ψ_1 at the next time step. $\mathrm{F}\psi_1$ means there is at least one time step at which ψ_1 is satisfied. $\psi_1 \mathrm{W} \psi_2$ and $\psi_1 \mathrm{S} \psi_2$ require ψ_1 to be always satisfied in the time period specified by ψ_2. We formally define them as follows:

$$f_t(\mathrm{X}\psi_1, \hat{\boldsymbol{y}}) = f_{t+1}(\psi_1, \hat{\boldsymbol{y}}), \tag{7}$$

$$f_t(\mathrm{F}\psi_1, \hat{\boldsymbol{y}}) = \max^\gamma\{f_{t:L-1}(\psi_1, \hat{\boldsymbol{y}})\}, \tag{8}$$

$$f_t(\psi_1 \mathrm{W} \psi_2, \hat{\boldsymbol{y}}) = \min^\gamma\{f_{t:k}(\psi_1, \hat{\boldsymbol{y}})\}, \text{where } k \geq t \text{ is the min. integer s.t. } f_k(\psi_2, \hat{\boldsymbol{y}}) > 0, \tag{9}$$

$$f_t(\psi_1 \mathrm{S} \psi_2, \hat{\boldsymbol{y}}) = \min^\gamma\{f_{k:L-1}(\psi_1, \hat{\boldsymbol{y}})\}, \text{where } k \geq t \text{ is the min. integer s.t. } f_k(\psi_2, \hat{\boldsymbol{y}}) > 0. \tag{10}$$

In Eqn. (9), $\min^\gamma\{f_{t:k}(\psi_1, \hat{\boldsymbol{y}})\}$ is positive (True) iff. all elements of $\{f_{t:k}(\psi_1, \hat{\boldsymbol{y}})\}$ are positive, which means that ψ_1 stays True from time t to time k. This is consistent with the semantics of W. We use $\min^\gamma$ for S in Eqn. (10) for the same reason.

The definitions in Eqn. (3)-(10) provide a soundness guarantee for Ψ, allowing us to use $f_t(\psi, \hat{\boldsymbol{y}}) > 0$ as an optimization objective to enforce the constraints in ψ on a task model. Formally, when $\gamma \to \infty$, the approximated evaluation (because of $\min^\gamma$) becomes exact, and the following theorem is true by construction (a proof is provided in Section D.2):

Theorem 1. *(Soundness) With $\psi \in \Psi, \gamma \to \infty$: if $f_t(\psi, \hat{\boldsymbol{y}}) > 0$, then ψ is satisfied by $\hat{\boldsymbol{y}}$ at time t.*

In essence, the evaluation process propagates network predictions from atomic propositions to logic operators and finally to the formula. Equivalently, we can represent a formula as a directed acyclic graph (DAG) where each leaf node is labeled with True, False, or action a; and each internal node is labeled with logic operators. The edges of the DAG point from child nodes to their parents, along which the truth value propagates following Eqn. (3)-(10). Fig. 3 illustrates how a constraint "a person cannot pour water before taking a cup" is converted to a DAG and evaluated.

3.3 Constraints

This section discusses different types of constraints we find useful for action analysis and the way they are represented using Ψ. A constraint can be categorized into either a temporal or a non-temporal constraint based on whether it states the possibility and necessity of an action in a specific time period. We propose two temporal constraints, namely Backward Dependency (BD) and Forward Cancellation (FC), and two non-temporal constraints, namely Implication (Ip), and Exclusivity (Ex). We also introduce how constraints of these types are curated from the training annotations.

Backward Dependency (BD) It is important to specify the proper order of actions because it usually determines the semantics of an activity. An order is generally related to temporal dependency: An action cannot be performed if its prerequisite actions did not occur. A way to describe this dependency is "one action must occur before another". We write this constraint as

$$\psi_{\text{BD}} = \wedge_{(a_i,a_j)\in\mathcal{B}}\text{BD}(a_i,a_j) = \wedge_{(a_i,a_j)\in\mathcal{B}}(\text{F}a_i \wedge \text{F}a_j) \rightarrow (\neg a_i \text{W} a_j),$$

where $(X \rightarrow Y)$ means "X implies Y" and is equivalent to $(\neg X \vee Y)$. This logic expression means the following: if a_i and a_j occurs in the same video, a_i must not occur until a_j occurs. Set $\mathcal{B}$ contains action pairs that satisfy this constraint. In practice, $(a_1, a_2) \in \mathcal{B}$ if a_2 always occurs before a_1 for any co-occurrence of a_1 and a_2 in the annotation.

Forward Cancellation (FC) Apart from backward dependence, actions can make some other actions impossible in all future time steps. For example, in a video of salad preparation, the action serve_salad_to_plate marks the end of the preparation. Once this action occurs, actions like cut_lettuce or cut_tomato should not occur thereafter. We write this constraint as

$$\psi_{\text{FC}} = \wedge_{(a_i,a_j)\in\mathcal{F}}\text{FC}(a_j,a_i) = \wedge_{(a_i,a_j)\in\mathcal{F}}(\text{F}a_i \wedge \text{F}a_j) \rightarrow (\neg a_i \text{S} a_j),$$

which reads "if a_i and a_j occur in the same video, a_i cannot occur after the occurrence of a_j". Set $\mathcal{F}$ contains action pairs that satisfy this constraint.

Implication (Ip) There could be some actions that are semantically dependent but temporally independent: these actions are necessary to complete an activity, but the order is not crucial. For example, in a video on juice preparation, the person must take_cup and peel_orange, but these two actions are not temporally correlated. We write this constraint as

$$\psi_{\text{Ip}} = \wedge_{(a_i,a_j)\in\mathcal{I}}\text{Ip}(a_i,a_j) = \wedge_{(a_i,a_j)\in\mathcal{I}}(\text{F}a_i \rightarrow \text{F}a_j).$$

Set $\mathcal{I}$ contains action pairs that satisfy this constraint. In practice, $(a_1, a_2) \in \mathcal{I}$ if a_2 always occurs if a_1 occurs in the annotation.

Exclusivity (Ex) As opposed to what implication constraints describe, some actions are mutually exclusive: they will never co-occur in the same video. For example, if we know that a video contains a single activity that is "coffee" or "frying eggs", then pour_coffee cannot happen if put_egg_into_plate occurs at any time in the video. This constraint is written as

$$\psi_{\text{Ex}} = \wedge_{(a_i,a_j)\in\mathcal{X}}\text{Ex}(a_i,a_j) = \wedge_{(a_i,a_j)\in\mathcal{X}}(\text{F}a_i \rightarrow \neg\text{F}a_j).$$

Set $\mathcal{X}$ contains pairs of actions that never occur in the same video.

Finally, we connect all constraints using $\wedge$ as $\psi = \psi_{\text{BD}} \wedge \psi_{\text{FC}} \wedge \psi_{\text{Ip}} \wedge \psi_{\text{Ex}}$.

3.4 Training with Constraints

During training, we hope that the output $\hat{\boldsymbol{y}}$ is constrained by both ground truth $\boldsymbol{y}$ and logic constraints described by ψ. The constraints from ground truth are enforced by a task-specific task loss $\mathcal{L}_{\mathsf{T}}(\hat{\boldsymbol{y}}, \boldsymbol{y})$, for example, framewise cross-entropy loss for the temporal action segmentation task. For logic constraints, we treat $\hat{\boldsymbol{y}}$ as an assignment of ψ and from Theorem 1 we know $f_t(\psi, \hat{\boldsymbol{y}}) > 0$ if $\hat{\boldsymbol{y}}$ satisfies ψ from time t. Therefore, for any g we can minimize the following objective:

$$\mathcal{L} = \mathcal{L}_{\mathsf{T}} + \lambda\mathcal{L}_{\mathsf{TL}} = \mathcal{L}_{\mathsf{T}}(\hat{\boldsymbol{y}}, \boldsymbol{y}) + \lambda\sigma\big(f_0(\psi, \hat{\boldsymbol{y}})\big), \tag{11}$$

where $\mathcal{L}_{\mathsf{T}}$ is the loss term for the target task, λ is the weight of the loss of logic $\mathcal{L}_{\mathsf{TL}}$, and $\sigma(x) = \log(1 + e^{-x})$ penalizes negative evaluation results. We set $t = 0$ in $f_t(\psi, \hat{\boldsymbol{y}})$ in our experiments, since we require the prediction to satisfy the constraints from the first frame. Note that it is possible to set t to different values so that the constraints can be applied flexibly at different temporal locations.

4 Experiments

In this section, we first assess the proposed DTL using the temporal action segmentation task. Through this task, we show that DTL can provide dependency information to improve the performance of action analysis models, in both quantitative and qualitative manners. Then, we perform an ablation study to show the effects of different types of constraints. Finally, with a gradient-based analysis, we explain how DTL affects the task model. All experiments are run with PyTorch 1.10 on an NVIDIA A6000 GPU. More details are covered in the appendix.

Table 1: Results of action segmentation on 50Salads dataset.

Task Model		Edit	F1@10	F1@25	F1@50	Acc
GRU	Base	55.2 ± 2.3	63.7 ± 2.3	60.5 ± 2.7	53.4 ± 2.9	79.0 ± 2.4
	Base + DTL	62.1 ± 1.6	69.3 ± 1.4	66.5 ± 1.8	58.9 ± 1.9	80.3 ± 2.2
	Gain	$\mathbf{7.1 \pm 2.8}$	$\mathbf{5.6 \pm 2.7}$	$\mathbf{6.0 \pm 3.2}$	$\mathbf{5.5 \pm 3.6}$	$\mathbf{1.3 \pm 3.3}$
MS-TCN	Base [15]	67.9	76.3	74.0	64.5	80.7
	Base (Rerun)	69.5 ± 1.7	75.7 ± 1.7	73.0 ± 1.9	64.5 ± 2.2	80.0 ± 1.4
	Base + DTL	70.5 ± 1.0	78.3 ± 1.3	76.5 ± 1.1	67.6 ± 1.9	81.5 ± 1.5
	Gain	$\mathbf{1.0 \pm 2.0}$	$\mathbf{2.6 \pm 2.1}$	$\mathbf{3.5 \pm 2.2}$	$\mathbf{3.0 \pm 2.9}$	$\mathbf{1.5 \pm 2.0}$
ASFormer	Base [59]	79.6	85.1	83.4	76.0	85.6
	Base (Rerun)	76.9 ± 0.9	83.6 ± 0.9	81.5 ± 0.8	73.9 ± 1.3	84.2 ± 1.2
	Base + DTL	80.5 ± 1.5	87.1 ± 1.3	85.7 ± 1.2	78.5 ± 1.6	86.9 ± 1.5
	Gain	$\mathbf{3.6 \pm 1.7}$	$\mathbf{3.5 \pm 1.6}$	$\mathbf{4.2 \pm 1.4}$	$\mathbf{4.6 \pm 2.1}$	$\mathbf{2.7 \pm 1.9}$

Table 2: Results of action segmentation on Breakfast dataset.

Task Model		Edit	F1@10	F1@25	F1@50	Acc
GRU	Base	56.8 ± 2.0	53.3 ± 2.3	48.4 ± 2.5	38.4 ± 2.1	70.0 ± 1.7
	Base + DTL	58.4 ± 1.7	56.5 ± 1.1	51.4 ± 1.3	40.7 ± 2.1	70.3 ± 1.1
	Gain	$\mathbf{1.6 \pm 2.7}$	$\mathbf{3.2 \pm 2.6}$	$\mathbf{3.0 \pm 2.8}$	$\mathbf{2.3 \pm 2.9}$	$\mathbf{0.4 \pm 2.0}$
MS-TCN	Base [15]	61.7	52.6	48.1	37.9	66.3
	Base (Rerun)	71.2 ± 1.4	71.7 ± 1.3	65.7 ± 1.5	52.3 ± 1.8	71.3 ± 1.2
	Base + DTL	71.6 ± 1.1	73.0 ± 0.4	67.7 ± 1.2	54.4 ± 0.8	72.3 ± 0.5
	Gain	$\mathbf{0.5 \pm 1.8}$	$\mathbf{1.2 \pm 1.3}$	$\mathbf{2.0 \pm 2.0}$	$\mathbf{2.1 \pm 2.0}$	$\mathbf{1.1 \pm 1.3}$
ASFormer	Base [59]	75.0	76.0	70.6	57.4	73.5
	Base (Rerun)	76.2 ± 1.4	77.8 ± 1.2	72.9 ± 1.6	60.5 ± 1.7	75.0 ± 1.0
	Base + DTL	77.7 ± 1.6	78.8 ± 1.1	74.5 ± 1.5	62.9 ± 1.6	75.8 ± 0.9
	Gain	$\mathbf{1.5 \pm 2.1}$	$\mathbf{1.1 \pm 1.6}$	$\mathbf{1.6 \pm 2.2}$	$\mathbf{2.4 \pm 2.4}$	$\mathbf{0.8 \pm 1.4}$

4.1 Temporal Action Segmentation

Datasets We use 50Salads [51] and Breakfast [33] for this task. In these datasets, each video spans at least 200 seconds and contains at least five different actions. 50Salads contains 50 videos of salad preparation with frame-level action annotations of 19 actions. We generated a total of 313 constraints from its annotation. Breakfast consists of 1,712 videos with 18 video-level activities, and each frame has one of the 47 actions. We use only its action-level annotations for this task. There are a total of 2,145 constraints for this dataset. The details of the constraints are provided in Section E.

Task Models We use three task models in evaluation: a single-layer bidirectional Gated Recurrent Unit (GRU) [5], a temporal convolution model MSTCN [15], and a transformer model ASFormer [59]. We use all three task models to examine the performance gain brought by DTL. For the ablation study and further discussion, we use GRU and MSTCN for their ease of training. The architecture of GRU is detailed in the appendix. For experiments on MSTCN and ASFormer, we retrain the corresponding models using their released source codes. The task models are trained and assessed using the protocol in [15], where the inputs to the task models are the 2048-dimension features extracted using I3D [4] pre-trained on ImageNet [10]. Frame-wise cross-entropy is used as the task loss for all the task models. Levenshtein distance (Edit), F1 score with thresholds 0.1, 0.25, and 0.5 (F1@{10,25,50}), and frame-wise accuracy (Acc) are used to measure the quality of the outputs. The results are from a k-fold cross-validation, where $k = 5$ for 50Salads and $k = 4$ for Breakfast.

Performance Table 1 and 2 show the performance of task models trained without (Base) and with (Base + DTL) temporal logic objective $\mathcal{L}_{\mathsf{TL}}$, and the performance difference (Gain), on 50Salads and Breakfast respectively. For completeness, we also include the base performance published in the original papers where applicable. A prominent observation is that the performance gain is

Table 3: Performance gain of GRU and MSTCN when trained with individual/all types of constraints. There are no exclusivity constraints in 50Salads.

(a) 50Salads

Task Model		Edit	F1@10	F1@25	F1@50	Acc
GRU	+BD	0.7	-1.3	-0.3	-0.3	0.5
	+FC	7.0	5.5	6.6	6.3	2.1
	+Ex	—	—	—	—	—
	+Ip	-0.8	-0.9	-0.8	-1.2	0.6
	+All	7.1	5.6	6.0	5.5	1.3
MSTCN	+BD	0.1	0.3	0.6	0.0	0.1
	+FC	0.2	1.0	1.0	0.6	0.1
	+Ex	—	—	—	—	—
	+Ip	0.8	2.6	2.8	1.9	1.0
	+All	1.0	2.6	3.5	3.0	1.5

(b) Breakfast

Task Model		Edit	F1@10	F1@25	F1@50	Acc
GRU	+BD	0.4	0.8	0.3	0.2	-0.2
	+FC	1.0	2.2	1.8	1.1	-0.1
	+Ex	0.3	1.2	1.7	0.9	0.1
	+Ip	1.1	2.2	2.1	1.9	0.8
	+All	1.6	3.2	3.0	2.3	0.4
MSTCN	+BD	0.2	1.1	1.4	1.7	0.7
	+FC	-0.1	0.8	1.3	0.8	0.2
	+Ex	0.1	0.7	0.8	1.1	-0.7
	+Ip	0.0	0.4	1.0	1.0	-0.2
	+All	0.5	1.2	2.0	2.1	1.1

(a) Example output of MSTCN on 50Salads.

(b) Example output of GRU on Breakfast.

Figure 4: Qualitative comparison between model trained with and without DTL, which shows that logical errors are fixed by DTL. In each group, from the top to the bottom show the ground truth (GT), the prediction from baseline task model, and the prediction from task model trained with DTL.

consistently positive. This indicates the applicability and efficacy of DTL on task models with different architectures and on datasets of different scales. Another observation is that the gains on different metrics are not even: there are noticeable improvements on Levenshtein distance and F1 scores while the improvement on frame-wise accuracy is not as significant. We conjecture the reason to be that frame-wise accuracy treats frames independently and does not distinguish well between models with and without the over-segmentation problem. Since DTL constrains the relations between *segments* of actions instead of frames, metrics based on segment differences like Levenshtein distance and F1-scores are therefore comparably better reflections of performance improvement.

4.2 Ablation Study

To understand the effects of each constraint type, we perform an ablation study using GRU and MSTCN on both datasets. In each experiment, we apply only one type of constraint and retrain the task model. The results are shown in Table 3. The first observation is that while applying constraints helps the task model, the extent of improvement differs for different models on different datasets. On 50Salads, nearly all the action classes are present in each video sample, making Ip constraints less helpful for GRU. However, Ip is the most helpful constraint type for MSTCN, which indicates that MSTCN is comparably weaker in modelling action co-occurrence and benefits more from this constraint. On the other hand, the more complicated nature of Breakfast makes both models benefit from all four types of constraints. Another important observation is that more constraints do not guarantee better performance. For example, GRU enjoys the highest improvement with only FC constraints — higher than when BD and Ip are added. This reveals a trade-off between precision and completeness of constraints: Because the $\min^\gamma$ and $\max^\gamma$ functions used in Eqn. (5)-(10) are approximations, the evaluation becomes less accurate when the number of constraints increases. For MSTCN on 50Salads and both models on Breakfast, the benefit of more complete constraints outweighs the increased evaluation errors, resulting in more improvements when using all constraints. Understanding the fine-grained interactions between the constraints and different backbone models remains an open question.

4.3 Qualitative Results

We qualitatively compare predictions from task models trained with and without DTL in Fig. 4. In Fig. 4a, we show the output of MSTCN on 50Salads. Note that the baseline MSTCN

Figure 5: A snippet prediction from 50Salads and the most influential constraints for each segmentation. The black arrows represent temporal order. The colored arrows represent constraint types.

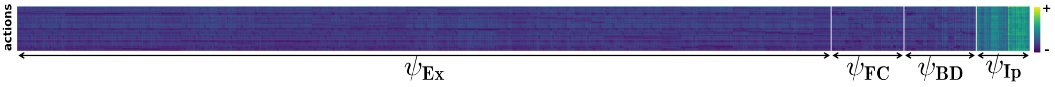

Figure 6: A heatmap showing $\mathbf{\Delta}_a^{\psi_i}$ of Breakfast, the vertical axis represents the 47 actions a and the horizontal axis is the 2,145 constraints ψ_i. Labels at the bottom indicate constraint types. A brighter color indicates that a constraint has a more positive effect on an action.

predicts a mix_dressing action after add_dressing, which is in conflict with a FC constraint FC(add_dressing, mix_dressing) because dressing cannot be mixed after it has been added to the salad. A similar example of GRU on Breakfast is shown in Fig. 4b, where the action pour_coffee is erroneously predicted in a video for "tea preparation". The baseline output violates an exclusivity constraint F add_teabag $\rightarrow$ $\neg$F pour_coffee because, except in special cases, coffee should not be added to tea. Both errors are fixed in the models trained with DTL. Moreover, we notice that correcting an individual wrong segment improves the quality of the whole output sequence. This is because the task models we consider are: (a) recurrent models that condition current output on past inputs, or (b) convolutional/transformer models that predict based on receptive field/attention information. In either case, correcting an error fixes the cascade effect it has on the model states, which improves the quality of all nearby outputs.

4.4 Analyzing the Effects of DTL

One of the greatest benefits of DTL is that it provides additional supervisory signals for temporal constraints, which are implicit in training data. We would like to understand how those signals affect the task model. Formally, given a model g parameterized by $\mathbf{\Theta}$ and an input $\boldsymbol{x}$, we would like to know if a constraint ψ_i promotes or suppresses the model's output about action a at time t. We assume that the output of the model is continuous with respect to its parameters, or $\lim_{\|\delta\mathbf{\Theta}\|\rightarrow 0} g_{\mathbf{\Theta}+\delta\mathbf{\Theta}}(\boldsymbol{x}) = g_{\mathbf{\Theta}}(\boldsymbol{x})$. With this assumption, the effect of ψ_i on $\mathbf{\Theta}$ can be approximated by the changes in the output as we update $\mathbf{\Theta}$ based on $f_0(\psi_i, \hat{\boldsymbol{y}})$. Specifically, we first compute $\delta^{\psi_i}\mathbf{\Theta} = \partial\sigma(f_0(\psi_i, \hat{\boldsymbol{y}}))/\partial\mathbf{\Theta}$, where $\sigma(x) = \log(1 + e^{-x})$ as in Eqn. (11). Then, we update Θ as $\Theta' = \Theta - \gamma\delta^{\psi_i}\Theta$, where $\gamma = 10^{-4}$ is a small update step. Finally, the difference in output caused by ψ_i is obtained as $\mathbf{\Delta}^{\psi_i,\boldsymbol{x}} = g_{\Theta'}(\boldsymbol{x}) - g_{\Theta}(\boldsymbol{x})$, where $\mathbf{\Delta}^{\psi_i,\boldsymbol{x}} \in \mathbb{R}^{N\times L}$. A positive (negative) $\Delta_{a,t}^{\psi_i,\boldsymbol{x}}$ indicates a promotive (suppressive) effect of ψ_i on $\hat{\boldsymbol{y}}_{a,t}$ as it increases (decreases) the score for action a at time t.

We use $\mathbf{\Delta}^{\psi_i,\boldsymbol{x}}$ to pinpoint the constraint that makes a model predict an action a at time t. This can be done by calculating $\Delta_{a,t}^{\psi_i,\boldsymbol{x}}$ for all ψ_i in ψ and find the most positively influential constraint as $\arg\max_{\psi_i} \Delta_{a,t}^{\psi_i,\boldsymbol{x}}$. For example, Fig. 5 shows a snippet of prediction from MSTCN on 50Salads and the constraints we found contributed the most to each predicted segment. One important observation from this is that actions in earlier time steps are generally promoted by later actions, as the former could serve as the prerequisites of the latter.

We can also summarize the total effect of ψ_i on all time steps and samples as $\mathbf{\Delta}_a^{\psi_i} = \frac{1}{|\mathcal{X}|}\sum_{\boldsymbol{x}\in\mathcal{X}}\frac{1}{L}\sum_{t=0}^{L-1}\Delta_{a,t}^{\psi_i,\boldsymbol{x}}$. $\mathbf{\Delta}_a^{\psi_i}$ provides two pieces of information. The first is the effect of ψ_i on all actions. As illustrated as a heatmap in Fig. 6, $\mathbf{\Delta}_a^{\psi_i}$ suggest that the Ip constraints play mainly promotional roles, where the FC, BD, and Ex constraints are mostly suppressive. This is consistent with our design intuitions: Ip encourages co-occurrence of actions, FC and Ex constraints are intrinsically prohibitive, and BD constraints suppress an action if its prerequisites are missing. The second information is how an action a is affected by other actions. This can be determined by selecting the most promotive and suppressive constraints for action a, and checking the actions involved in these constraints. Table 4 shows the relations between example actions in the Breakfast

Table 4: Actions in Breakfast dataset, and the most influential constraints. "$\sim$" refers to the action in the leftmost column of the row it is in.

Action	Top Positive Constraint	Top Negative Constraint
stir_fruit	Ip(cut_fruit, $\sim$)	BD($\sim$, put_fruit2bowl)
stir_tea	Ip(add_teabag, $\sim$)	Ex(put_milk, $\sim$)
put_toppingOnTop	Ip(put_bunTogether, $\sim$)	FC(put_bunTogether, $\sim$)

dataset. An interesting observation is that for put_ToppingOnTop the most promotive and suppressive action is the same put_PutBunTogether, meaning that relation between two actions could largely change depending on the context. This also indicates that DTL can help the model learn different dependencies between the same pair of actions.

5 Conclusion

We propose DTL, a framework that uses temporal logic to constrain the training of action analysis models. Experimental results on the action segmentation task show that DTL effectively improves the performance of task models with different architectures. An ablation study reveals the divergent effect of different types of rules on different task models. Our work suggests that temporal constraints can be explicitly provided to a deep network and reduce logical errors in its output. The source code of our work is accessible at `https://diff-tl.github.io/`.

Limitations There remain some limitations in this work. First, in this work, we only explored a subset of temporal knowledge that is expressible as frequency matrices. When such temporal correlation is sparse, DTL is less effective. This calls for more flexible knowledge curation methods (e.g. with human involvement). Besides, non-temporal knowledge about actions, such as object affordance, can be exploited to handle more complicated actions like the verb-object compositions in Epic-Kitchens [7]. Second, in terms of knowledge type, DTL can benefit from more expressive logic languages, like Allen's Interval Algebra [2], which supports first-order temporal constraints beyond necessities and possibilities. Moreover, DTL formulae are evaluated in their raw forms, which means that the evaluation efficiency could be further improved by better formula compilation [9] and optimized message passing on directed acyclic graphs. In addition, the constraints we collected from dataset annotations only form a partial and clean view of real-world scenarios. The real-world constraints are more comprehensive and complex, which poses a higher requirement on generalization and robustness against uncertainties.

Acknowledgments and Disclosure of Funding

This research is partially supported by the National Research Foundation, Singapore under its Strategic Capability Research Centres Funding Initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

References

[1] Hyemin Ahn and Dongheui Lee. Refining action segmentation with hierarchical video representations. In *CVPR*, pages 16302–16310, 2021.

[2] James F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843, 1983.

[3] Subhabrata Bhattacharya, Mahdi M. Kalayeh, Rahul Sukthankar, and Mubarak Shah. Recognition of complex events: Exploiting temporal dynamics between underlying concepts. In *CVPR*, pages 2243–2250, 2014.

[4] João Carreira and Andrew Zisserman. Quo vadis, action recognition? A new model and the kinetics dataset. In *CVPR*, pages 4724–4733, 2017.

[5] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS Workshop on Deep Learning*, 2014.

[6] Marco Cuturi and Mathieu Blondel. Soft-DTW: a differentiable loss function for time-series. In *ICML*, pages 894–903, 2017.

[7] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Antonino Furnari, Evangelos Kazakos, Jian Ma, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Rescaling egocentric vision: Collection, pipeline and challenges for EPIC-KITCHENS-100. *Int. J. Comput. Vis.*, 130(1):33–55, 2022.

[8] Adnan Darwiche. On the tractable counting of theory models and its application to truth maintenance and belief revision. *J. Appl. Non Class. Logics*, 11(1-2):11–34, 2001.

[9] Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *J. Artif. Intell. Res.*, 17:229–264, 2002.

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009.

[11] Ivan Donadello, Luciano Serafini, and Artur S. d'Avila Garcez. Logic tensor networks for semantic image interpretation. In *IJCAI*, pages 1596–1602, 2017.

[12] Paolo Dragone, Stefano Teso, and Andrea Passerini. Neuro-symbolic constraint programming for structured prediction. In *NeSy*, pages 6–14, 2021.

[13] Georgios Fainekos, Hadas Kress-Gazit, and George J. Pappas. Temporal logic motion planning for mobile robots. In *ICRA*, pages 2020–2025, 2005.

[14] Georgios E. Fainekos and George J. Pappas. Robustness of temporal logic specifications for continuous-time signals. *Theor. Comput. Sci.*, 410(42):4262–4291, 2009.

[15] Yazan Abu Farha and Jürgen Gall. MS-TCN: multi-stage temporal convolutional network for action segmentation. In *CVPR*, pages 3575–3584, 2019.

[16] Alireza Fathi, Xiaofeng Ren, and James M. Rehg. Learning to recognize objects in egocentric activities. In *CVPR*, pages 3281–3288, 2011.

[17] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *ICCV*, pages 6201–6210, 2019.

[18] Marc Fischer, Mislav Balunovic, Dana Drachsler-Cohen, Timon Gehr, Ce Zhang, and Martin T. Vechev. DL2: training and querying neural networks with logic. In *ICML*, pages 1931–1941, 2019.

[19] Eleonora Giunchiglia and Thomas Lukasiewicz. Multi-label classification neural networks with hard logical constraints. *J. Artif. Intell. Res.*, 72:759–818, 2021.

[20] Eleonora Giunchiglia, Mihaela Catalina Stoian, and Thomas Lukasiewicz. Deep learning with logical constraints. In *IJCAI-ECAI*, 2022.

[21] Asaad Hakeem, Yaser Sheikh, and Mubarak Shah. CASEE: A hierarchical event representation for the analysis of videos. In *AAAI*, pages 263–268, 2004.

[22] Joseph Y. Halpern and Yoav Shoham. A propositional modal logic of time intervals. *J. ACM*, 38(4):935–962, 1991.

[23] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *CVPR*, pages 961–970. IEEE Computer Society, 2015.

[24] Nicholas Hoernle, Rafael-Michael Karampatsis, Vaishak Belle, and Kobi Gal. MultiplexNet: Towards fully satisfied logical constraints in neural networks. In *AAAI*, 2022.

[25] Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard H. Hovy, and Eric P. Xing. Harnessing deep neural networks with logic rules. In *ACL*, 2016.

[26] Yifei Huang, Yusuke Sugano, and Yoichi Sato. Improving action segmentation via graph-based temporal reasoning. In *CVPR*, pages 14021–14031, 2020.

[27] Craig Innes and Subramanian Ramamoorthy. Elaborating on learned demonstrations with temporal logic specifications. In *RSS*, 2020.

[28] Yuchi Ishikawa, Seito Kasai, Yoshimitsu Aoki, and Hirokatsu Kataoka. Alleviating over-segmentation errors by detecting action boundaries. In *WACV*, pages 2321–2330, 2021.

[29] Alec Kerrigan, Kevin Duarte, Yogesh S. Rawat, and Mubarak Shah. Reformulating zero-shot action recognition for multi-label actions. In *NeurIPS*, pages 25566–25577, 2021.

[30] Yoon Kim, Yacine Jernite, David A. Sontag, and Alexander M. Rush. Character-aware neural language models. In *AAAI*, pages 2741–2749, 2016.

[31] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

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

[33] Hilde Kuehne, Ali Bilgin Arslan, and Thomas Serre. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *CVPR*, pages 780–787, 2014.

[34] Hilde Kuehne, Juergen Gall, and Thomas Serre. An end-to-end generative framework for video segmentation and recognition. In *WACV*, pages 1–8, 2016.

[35] Hilde Kuehne, Alexander Richard, and Juergen Gall. A hybrid RNN-HMM approach for weakly supervised temporal action segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(4):765–779, 2020.

[36] Phong Le and Willem H. Zuidema. Compositional distributional semantics with long short term memory. In **SEM@NAACL-HLT*, pages 10–19, 2015.

[37] Colin Lea, Michael D. Flynn, René Vidal, Austin Reiter, and Gregory D. Hager. Temporal convolutional networks for action segmentation and detection. In *CVPR*, pages 1003–1012, 2017.

[38] Peng Lei and Sinisa Todorovic. Temporal deformable residual networks for action segmentation in videos. In *CVPR*, pages 6742–6751, 2018.

[39] Shi-Jie Li, Yazan AbuFarha, Yun Liu, Ming-Ming Cheng, and Juergen Gall. Ms-tcn++: Multi-stage temporal convolutional network for action segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2020.

[40] Yitao Liang and Guy Van den Broeck. Learning logistic circuits. In *AAAI*, pages 4277–4286, 2019.

[41] Angelo Montanari. *Metric and Layered Temporal Logic for Time Granularity*. PhD thesis, University of Amsterdam, 1996.

[42] Toshiyuki Ogihara. Tense and aspect in truth-conditional semantics. *Lingua*, 117:392–418, 2007.

[43] Dhruvesh Patel, Pavitra Dangati, Jay-Yoon Lee, Michael Boratko, and Andrew McCallum. Modeling label space interactions in multi-label classification using box embeddings. In *ICLR*, 2022.

[44] A. J. Piergiovanni and Michael S. Ryoo. Temporal gaussian mixture layer for videos. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 5152–5161. PMLR, 2019.

[45] Amir Pnueli. The temporal logic of programs. In *FOCS*, pages 46–57, 1977.

[46] Aniruddh Gopinath Puranic, Jyotirmoy V. Deshmukh, and Stefanos Nikolaidis. Learning from demonstrations using signal temporal logic in stochastic and continuous domains. *IEEE Robotics Autom. Lett.*, 6(4):6250–6257, 2021.

[47] Alexander Richard, Hilde Kuehne, and Juergen Gall. Weakly supervised action learning with RNN based fine-to-coarse modeling. In *CVPR*, pages 1273–1282, 2017.

[48] Sungyong Seo, Sercan Ö. Arik, Jinsung Yoon, Xiang Zhang, Kihyuk Sohn, and Tomas Pfister. Controlling neural networks with rule representations. In *NeurIPS*, pages 11196–11207, 2021.

[49] Gunnar A. Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *ECCV (1)*, volume 9905 of *Lecture Notes in Computer Science*, pages 510–526. Springer, 2016.

[50] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *CoRR*, abs/1212.0402, 2012.

[51] Sebastian Stein and Stephen J. McKenna. Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *UbiComp*, pages 729–738, 2013.

[52] Praveen Tirupattur, Kevin Duarte, Yogesh Singh Rawat, and Mubarak Shah. Modeling multi-label action dependencies for temporal action localization. In *CVPR*, pages 1460–1470, 2021.

[53] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *CVPR*, pages 6450–6459, 2018.

[54] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pages 6000–6010, 2017.

[55] Yaqi Xie, Ziwei Xu, Kuldeep S. Meel, Mohan S. Kankanhalli, and Harold Soh. Embedding symbolic knowledge into deep networks. In *NeurIPS*, pages 4235–4245, 2019.

[56] Yaqi Xie, Fan Zhou, and Harold Soh. Embedding symbolic temporal knowledge into deep sequential models. In *ICRA*, pages 4267–4273, 2021.

[57] Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck. A semantic loss function for deep learning with symbolic knowledge. In *ICML*, 2018.

[58] Ziwei Xu, Xudong Shen, Yongkang Wong, and Mohan S. Kankanhalli. Unsupervised motion representation learning with capsule autoencoders. In *NeurIPS*, pages 3205–3217, 2021.

[59] Fangqiu Yi, Hongyu Wen, and Tingting Jiang. ASFormer: Transformer for action segmentation. In *BMVC*, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 5.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] While Theorem 1 is true by construction, we provide a sketchy proof in Section D.2.
3. If you ran experiments...

(a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Training details are mostly provided in Section 4 while some extra information is in the appendix.

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] With an exception that error bars are omitted for clarity in Table 3.

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes] We used publically available 50Salads and Breakfast dataset, the original papers have been cited.

(b) Did you mention the license of the assets? [N/A]

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] The code and data are publicly available and the consent has been described in the cited papers.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] No such information was found in the dataset we use.

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Table of Notations

The table below shows the notations grouped by the modules.

Table A1: Table of Notations

Model	
$g_{\boldsymbol{\Theta}}$	A task model parameterized by $\boldsymbol{\Theta}$
$\boldsymbol{y}$	Ground truth
$\hat{\boldsymbol{y}}$	Output of the task model
a_i	The ith action class, or its corresponding atomic proposition
ψ	A DTL formula
Ψ	The set of all DTL formulas
$f_t(\psi, \hat{\boldsymbol{y}})$	Evaluation of $\hat{\boldsymbol{y}}$ against formula ψ at time t
Sizes	
L	Length of input sequence
N	Number of action classes
Logic Operators	
$\neg, \wedge, \vee$	Logical NEGATION, logical AND, and logical OR
X	Temporal operator NEXT
F	Temporal operator EVENTUAL
W	Temporal operator WEAK_UNTIL
S	Temporal operator SINCE
Constraints	
BD(a_i, a_j)	Action a_i is backward dependent on action a_j
FC(a_i, a_j)	Action a_i forward cancels action a_j
Ip(a_i, a_j)	Action a_i implies action a_j
Ex(a_i, a_j)	Action a_i excludes action a_j

B Implementation Details

Task Models. The implementation for MSTCN [15] and ASFormer [59] are from existing open-source code provided by corresponding authors. The GRU is implemented as follows: a fully-connected layer first transforms the 2048-dimension input into a 512-dimension vector. The vector is then sent to a bi-directional Gated Recurrent Unit layer with a hidden size of 512, which yields a 1024-dimension vector. The vector is finally transformed by a second fully-connected layer into an N-dimension vector, where each dimension represents the un-normalized score of an action class.

Hyperparameters. There are two hyperparameters in DTL: γ and λ. γ is the parameter for $\min^{\gamma}()$ and $\max^{\gamma}()$, and is set to 1. λ is the weight for temporal logic loss. We set $\lambda = 0.1$ for both GRU and MSTCN, and $\lambda = 0.005$ for ASFormer. All hyperparameters are determined empirically.

Training. All the experiments are performed on an NVIDIA A6000 GPU with PyTorch 1.10. The training for MSTCN and ASFormer follows the way defined in the corresponding source code. We use Adam [31] to optimize the GRU with a learning rate of 5×10^{-4} for 50 epochs. The model with the best validation performance is reported.

C Class-wise Performances

We provide class-wise performance (measured by F1@25) comparison between task models trained with and without DTL. The result is shown in Fig. A1 and Fig. A2. A general observation is that actions with lower performance on task models without DTL benefit more from DTL. We conjecture that actions with higher performance reflect the part of knowledge which is better learned by the

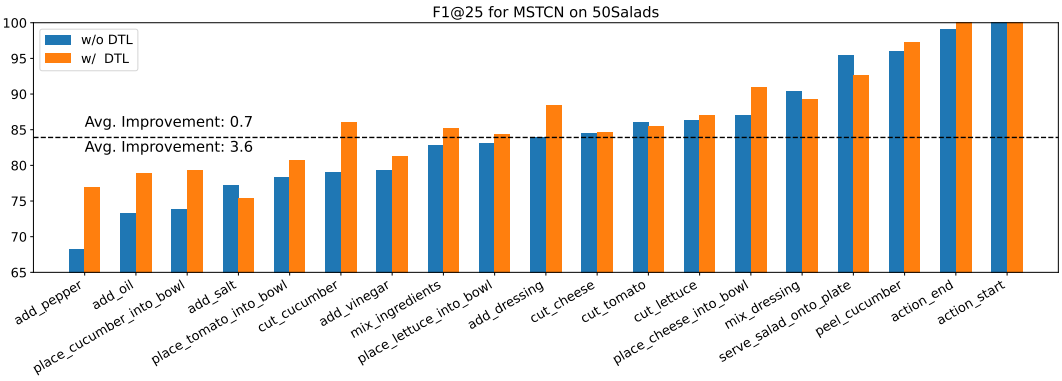

Figure A1: Class-wise F1@25 score for MSTCN on 50Salads. The classes on the horizontal axis are sorted based on the performance of the task model without DTL. Dashed line shows the median performance of all classes. The annotation above (below) the line indicates the averaged improvement for classes ranked at top (bottom) 50% in the baseline performance.

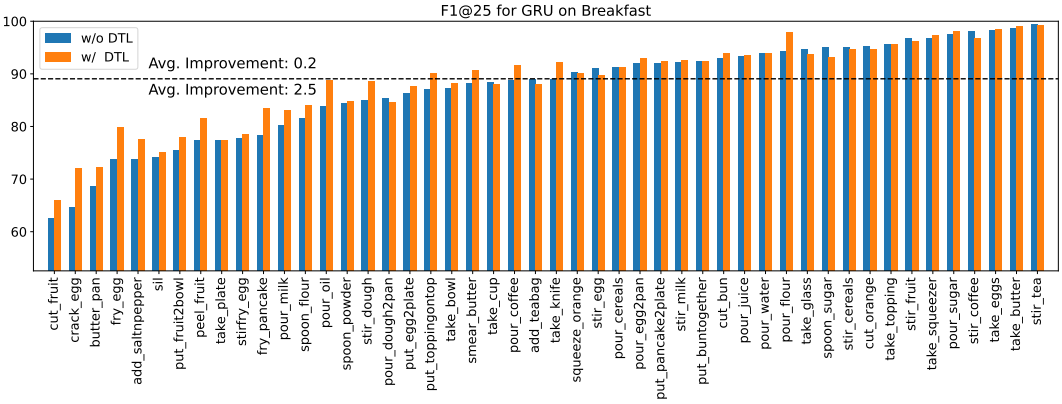

Figure A2: Class-wise F1@25 score for GRU on Breakfast. The classes on the horizontal axis are sorted based on the performance of the task model without DTL. Dashed line shows the median performance of all classes. The annotation above (below) the line indicates the average improvement for classes ranked at top (bottom) 50% in the baseline performance.

task model from the annotation. Therefore, DTL is less effective for this subset of actions because currently its constraints are also procured from annotation. On the other hand, this shows that DTL is able to help the model better learn constraints from annotation. We anticipate more performance improvement with more general constraints that go beyond knowledge in the annotations in future works.

D Extended Discussion on DTL

Section 3.1 introduced logic operators and briefly introduced their semantics. In Section 3.2, we introduced the evaluation of those operators in detail. In this section, we extend the discussion on DTL and answer the following two questions:

1. What is the formal definition to the semantics of DTL operators (cf. Section D.1)?
2. What is the relation between the evaluation of DTL operators related and their semantics (cf. Section D.2)?

D.1 Semantics of Logic Operators

This section formally defines the semantics of these logic operators, which is largely an extension of linear temporal logic. It is recommended to compare the semantics defined in Eqn. (A2)-(A9) below

with the evaluation defined in Eqn. (3)-(10). Also recall that $\psi \in \Psi$ takes the following forms:

$$\psi \in \Psi := \mathsf{True} \mid \mathsf{False} \mid a \mid \neg\psi_1 \mid (\psi_1 \wedge \psi_2) \mid (\psi_1 \vee \psi_2) \mid \mathrm{X}\psi_1 \mid \mathrm{F}\psi_1 \mid (\psi_1 \mathrm{W} \psi_2) \mid (\psi_1 \mathrm{S} \psi_2). \quad \text{(A1)}$$

Before defining the semantics, we first introduce a symbol ω, which is a truth assignment of ψ in L time steps. ω is an L-long word $\omega_{0:L-1} = \omega_0\omega_1 \ldots \omega_{L-1}$, where ω_t is the set of atomic propositions that are True at time t[b]. We use $\omega_{t_1:t_2}$ to denote a substring of ω from time t_1 to time t_2 and $\omega_{t_1:}$ as a shorthand for $\omega_{t_1:L-1}$. $\omega_{t:} \models \psi$ means $\omega_{t:}$ satisfies ψ under the semantics of Ψ.

Now we start to define the semantics of Ψ. The semantics of constants are straight-forward:

$$\omega_{t:} \models \mathsf{True}, \quad \omega_{t:} \not\models \mathsf{False}, \quad \omega_{t:} \models a \text{ iff. } a \in \omega_t \quad \text{(A2)}$$

which states that any truth assignment satisfies ψ if $\psi = \mathsf{True}$, and no truth assignment will satisfy ψ if $\psi = \mathsf{False}$. If $\psi = a$, then a satisfying assignment must assign True to a (make a occur) at time 0.

For common propositional logic operators $\neg$ (NEGATION), $\wedge$ (AND) and $\vee$ (OR), we define their semantics as

$$\omega_{t:} \models \neg\psi_1 \quad \text{iff. } \omega_{t:} \not\models \psi_1, \quad \text{(A3)}$$

$$\omega_{t:} \models \psi_1 \wedge \psi_2 \quad \text{iff. } \omega_{t:} \models \psi_1 \text{ and } \omega_{t:} \models \psi_2, \quad \text{(A4)}$$

$$\omega_{t:} \models \psi_1 \vee \psi_2 \quad \text{iff. } \omega_{t:} \models \psi_1 \text{ or } \omega_{t:} \models \psi_2, \quad \text{(A5)}$$

where $\psi_1, \psi_2 \in \Psi$.

Finally, we define the semantics for modal operators X, F, W, and S:

$$\omega_{t:} \models \mathrm{X}\psi_1 \quad \text{iff. } \omega_{t+1:} \models \psi_1, \quad \text{(A6)}$$

$$\omega_{t:} \models \mathrm{F}\psi_1 \quad \text{iff. } \exists t \leq t_1 < L \text{ s.t. } \omega_{t_1:} \models \psi_1, \quad \text{(A7)}$$

$$\omega_{t:} \models (\psi_1 \mathrm{W} \psi_2) \quad \text{iff. } \exists t \leq t_2 < L \text{ s.t. } \omega_{t_2:} \models \psi_2 \text{ and } \forall t \leq t_1 < t_2,\ \omega_{t_1:} \models \psi_1, \text{ or } \forall t \leq t_2 < L,\ \omega_{t_2:} \not\models \psi_2, \quad \text{(A8)}$$

$$\omega_{t:} \models (\psi_1 \mathrm{S} \psi_2) \quad \text{iff. } \exists t \leq t_2 < L \text{ s.t. } \omega_{t_2:} \models \psi_2 \text{ and } \forall t_2 \leq t_1 < L,\ \omega_{t_1:} \models \psi_1, \text{ or } \forall t \leq t_2 < L,\ \omega_{t_2:} \not\models \psi_2. \quad \text{(A9)}$$

Intuitively, X shifts the time to the next time step. F states the necessity of ψ_1. Both W and S state the possibility of ψ_1 during some time intervals specified by the occurrence of ψ_2.

D.2 A Proof of Soundness for DTL

Theorem 1 in the main paper states the soundness of DTL: if $f_t(\psi, \hat{\boldsymbol{y}}) > 0$, then $\hat{\boldsymbol{y}}$ satisfies the constraints in formula ψ at time t. In other words, it states that *evaluated satisfaction entails semantic satisfaction*. In this section, we provide a sketchy proof for Theorem 1.

This proof starts with a basic assumption, where we formally establish the connection between $\hat{\boldsymbol{y}}$ and a truth assignment ω:

Assumption A1. $\hat{\boldsymbol{y}}_{a_i,t} > 0 \Leftrightarrow a_i \in \omega_t$.

We also have the following assumption that rules out value "0" throughout the evaluation. This is necessary as $-0 = 0$ can break the evaluation of logical negation.

Assumption A2. $\hat{\boldsymbol{y}}_{a_i,t} \neq 0$, $\forall a \in \mathcal{A}$ *and* $0 \leq t < L$,

Then we can prove the following Theorem A1, which directly leads to Theorem 1.

Theorem A1. *With* $\psi \in \Psi, \gamma \to \infty$*:* $f_t(\psi, \hat{\boldsymbol{y}}) > 0 \Leftrightarrow \omega_{t:} \models \psi$.

Proof. Let $\psi, a, \psi_1, \psi_2 \in \Psi$, where a is an atomic proposition. We need to prove Theorem 1 for all forms in Eqn. (A1). We do this by induction.

<u>**Base Case**</u> When $\psi = a$.

$$f_t(\psi, \hat{\boldsymbol{y}}) > 0 \overset{\psi=a}{\Longleftrightarrow} f_t(a, \hat{\boldsymbol{y}}) > 0 \overset{\text{Eqn. (3)}}{\Longleftrightarrow} \hat{\boldsymbol{y}}_{a,t} > 0 \overset{\text{Asm. A1}}{\Longleftrightarrow} a \in \omega_t$$

$$\overset{\text{Eqn. (A2)}}{\Longleftrightarrow} \omega_{t:} \models a \overset{\psi=a}{\Longleftrightarrow} \omega_{t:} \models \psi.$$

[b] In the context of temporal action segmentation, ω_t is equivalent to the set of action that occurs at time t.

Therefore the base case is true.

Inductive Step Inductive Hypothesis (I.H.): $f_t(\psi, \hat{\boldsymbol{y}}) > 0 \Leftrightarrow \omega_{t:} \models \psi$ for $\psi = \psi_1$ and $\psi = \psi_2$.

Case 1: $\psi = \neg\psi_1$.

- If $f_t(\psi, \hat{\boldsymbol{y}}) > 0$:

$$f_t(\psi, \hat{\boldsymbol{y}}) > 0 \xRightarrow{\psi=\neg\psi_1} f_t(\neg\psi_1, \hat{\boldsymbol{y}}) > 0 \xRightarrow{\text{Eqn. (4)}} -f_t(\psi_1, \hat{\boldsymbol{y}}) > 0$$
$$\Rightarrow f_t(\psi_1, \hat{\boldsymbol{y}}) < 0 \xRightarrow{\text{I.H.}} \omega_{t:} \not\models \psi_1 \xRightarrow{\text{Eqn. (A3)}} \omega_{t:} \models \neg\psi_1$$
$$\xRightarrow{\psi=\neg\psi_1} \omega_{t:} \models \psi.$$

- If $\omega_{t:} \models \psi$:

$$\omega_{t:} \models \psi \xRightarrow{\psi=\neg\psi_1} \omega_{t:} \models \neg\psi_1 \xRightarrow{\text{Eqn. (A3)}} \omega_{t:} \not\models \psi_1 \xRightarrow{\text{I.H. and Asm. A2}} f_t(\psi_1, \hat{\boldsymbol{y}}) < 0$$
$$\Rightarrow -f_t(\psi_1, \hat{\boldsymbol{y}}) > 0 \xRightarrow{\text{Eqn. (4)}} f_t(\neg\psi_1, \hat{\boldsymbol{y}}) > 0$$
$$\xRightarrow{\psi=\neg\psi_1} f_t(\psi, \hat{\boldsymbol{y}}) > 0.$$

Case 2: $\psi = (\psi_1 \wedge \psi_2)$.

$$f_t(\psi, \hat{\boldsymbol{y}}) > 0 \xLeftrightarrow{\psi=(\psi_1\wedge\psi_2)} f_t(\psi_1 \wedge \psi_2, \hat{\boldsymbol{y}}) > 0 \xLeftrightarrow{\text{Eqn. (5)}} \min\{f_t(\psi_1, \hat{\boldsymbol{y}}), f_t(\psi_2, \hat{\boldsymbol{y}})\} > 0$$
$$\Leftrightarrow f_t(\psi_1, \hat{\boldsymbol{y}}) > 0 \text{ and } f_t(\psi_2, \hat{\boldsymbol{y}}) > 0 \xLeftrightarrow{\text{I.H.}} \omega_{t:} \models \psi_1 \text{ and } \omega_{t:} \models \psi_2$$
$$\xLeftrightarrow{\text{Eqn. (A4)}} \omega_{t:} \models (\psi_1 \wedge \psi_2) \xLeftrightarrow{\psi=(\psi_1\wedge\psi_2)} \omega_{t:} \models \psi.$$

Case 3: $\psi = (\psi_1 \vee \psi_2)$. The proof is similar to Case 2.

Case 4: $\psi = \mathbf{X}\psi_1$.

$$f_t(\psi, \hat{\boldsymbol{y}}) > 0 \xLeftrightarrow{\psi=\mathrm{X}\psi_1} f_t(\mathrm{X}\psi_1, \hat{\boldsymbol{y}}) > 0 \xLeftrightarrow{\text{Eqn. (7)}} f_{t+1}(\psi_1, \hat{\boldsymbol{y}}) > 0$$
$$\xLeftrightarrow{\text{I.H.}} \omega_{t+1:} \models \psi_1 \xLeftrightarrow{\text{Eqn. (A6)}} \omega_{t:} \models \mathrm{X}\psi_1 \xLeftrightarrow{\psi=\mathrm{X}\psi_1} \omega_{t:} \models \psi.$$

Case 5: $\psi = \mathbf{F}\psi_1$.

$$f_t(\psi, \hat{\boldsymbol{y}}) > 0 \xLeftrightarrow{\psi=\mathrm{F}\psi_1} f_t(\mathrm{F}\psi_1, \hat{\boldsymbol{y}}) > 0 \xLeftrightarrow{\text{Eqn. (8)}} \max\{f_{t:L-1}(\psi_1, \hat{\boldsymbol{y}})\} > 0$$
$$\Leftrightarrow \exists t \leq t_1 < L \text{ s.t. } f_{t_1}(\psi_1, \hat{\boldsymbol{y}}) > 0 \xLeftrightarrow{\text{I.H.}} \omega_{t_1:} \models \psi_1$$
$$\xLeftrightarrow{\text{Eqn. (A7)}} \omega_{t:} \models \mathrm{F}\psi_1 \xLeftrightarrow{\psi=\mathrm{F}\psi_1} \omega_{t:} \models \psi.$$

Case 6: $\psi = (\psi_1 \mathbf{W} \psi_2)$.

- If $\exists k \in [t, L)$ s.t. $f_k(\psi_2, \hat{\boldsymbol{y}}) > 0$:

$$f_t(\psi, \hat{\boldsymbol{y}}) > 0 \xLeftrightarrow{\psi=(\psi_1\mathrm{W}\psi_2)} f_t(\psi_1 \mathrm{W} \psi_2, \hat{\boldsymbol{y}}) > 0$$
$$\xLeftrightarrow{\text{Eqn. (9)}} \min\{f_{k':t}(\psi_1, \hat{\boldsymbol{y}}) > 0\}, \text{ and } k' \in [t, L) \text{ is the min. integer s.t. } f_{k'}(\psi_2, \hat{\boldsymbol{y}}) > 0$$
$$\Leftrightarrow \exists t_1 \in [t, k] \text{ s.t. } f_{t_1}(\psi_1, \hat{\boldsymbol{y}}) > 0 \text{ and } \exists k \in [t, L) \text{ s.t. } f_k(\psi_2, \hat{\boldsymbol{y}}) > 0$$
$$\xLeftrightarrow{\text{I.H.}} \exists t_1 \in [t, k] \text{ s.t. } \omega_{t_1:} \models \psi_1 \text{ and } \exists k \in [t, L) \text{ s.t. } \omega_{k:} \models \psi_2$$
$$\xLeftrightarrow{\text{Eqn. (A8)}} \omega_{t:} \models (\psi_1 \mathrm{W} \psi_2) \xLeftrightarrow{\psi=(\psi_1\mathrm{W}\psi_2)} \omega_{t:} \models \psi.$$

- If $\forall k \in [t, L), f_k(\psi_2, \hat{\boldsymbol{y}}) < 0$: This is a special case for W (and S as well). When this happens, we directly set $f_t(\psi, \hat{\boldsymbol{y}}) = 1 > 0$ without any evaluation. Following Eqn. (9) we know $\omega_{t:} \models (\psi_1 \mathrm{W} \psi_2)$. Then immediately $f_t(\psi, \hat{\boldsymbol{y}}) = 1 > 0 \Leftrightarrow \omega_{t:} \models (\psi_1 \mathrm{W} \psi_2)$.

Case 7: $\psi = (\psi_1 \mathbf{S} \psi_2)$. The proof is similar to Case 6. □

E Constraints

This section provides a quick view of the constraints used in our experiment. We first explain how constraints are collected and then provide samples for different types of constraints collected for the two datasets used in our experiment.

E.1 Collecting Constraints

The constraints are automatically generated from the existing annotations of datasets. Algorithm A1 shows how statistics about co-occurrences between actions can be collected. Then Algorithm A2 uses those statistics to generate the four types of constraints discussed in Section 3.3.

Algorithm A1: Algorithm to collect the co-occurrence statistics from dataset annotation.

Input: Set of samples $\mathcal{M} = \{\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_M\}$, where $\boldsymbol{m}_i = [y_0, y_2, ..., y_{L-1}]$ and $y_i \in \mathcal{A}$ is the index of one of the N actions.

▷ $\boldsymbol{B}[a_i, a_j]$ `is the frequency of` a_i `occurring before` a_j
▷ $\boldsymbol{P}[a_i, a_j]$ `is the frequency of` a_i `occurring after` a_j
▷ $\boldsymbol{J}[a_i, a_j]$ `is the number of videos where` a_i `occurs with` a_j
▷ $\boldsymbol{C}[a_i, a_j]$ `is the number of videos where` a_i `occurs`

$\boldsymbol{B}, \boldsymbol{P}, \boldsymbol{J} \leftarrow$ zero matrices of size $N \times N$;
$\boldsymbol{C} \leftarrow$ zero vector of size N;
foreach $\boldsymbol{m} \in \mathcal{M}$ **do**
  occur_flags $\leftarrow$ zero vector of size N;
  co_occur_flags $\leftarrow$ zero matrix of size $N \times N$;
  $y_0, y_1, ..., y_{L-1} \leftarrow$ annotation of $\boldsymbol{m}$;
  foreach $t \in 0, 1, \ldots, L-1$ **do**
    if *occur_flags[y_t] == 0* **then**
      $\boldsymbol{C}[y_t] \leftarrow \boldsymbol{C}[y_t] + 1$;
      occur_flags[y_t] $\leftarrow$ 1;
    foreach $u \in \{0, 1, \ldots, t\}$ **do**
      $\boldsymbol{B}[y_u, y_t] \leftarrow \boldsymbol{B}[y_u, y_t] + 1$;
      if *co_occur_flags[y_t, y_u] == 0* **then**
        $\boldsymbol{J}[y_u, y_t] \leftarrow \boldsymbol{J}[y_u, y_t] + 1$;
        co_occur_flags[y_u, y_t] = 1;
    foreach $u \in \{t+1, t+2, \ldots, L-1\}$ **do**
      $\boldsymbol{P}[y_u, y_t] \leftarrow \boldsymbol{P}[y_u, y_t] + 1$;
      if *co_occur_flags[y_t, y_u] == 0* **then**
        $\boldsymbol{J}[y_u, y_t] \leftarrow \boldsymbol{J}[y_u, y_t] + 1$;
        co_occur_flags[y_u, y_t] = 1;
return $\{\boldsymbol{B}, \boldsymbol{P}, \boldsymbol{J}, \boldsymbol{C}\}$;

E.2 Samples of Constraints

For clarity, we show 10 entries for each type of constraint.

E.2.1 Breakfast

Breakfast contains 48 actions for ten different activities about breakfast preparation. Each video contains a single activity, which could be making coffee, cereal, tea, fried egg, pancake, sandwich,

Algorithm A2: Algorithm to generate the constraints from the collected statistics.

Input: Statistics $\{\boldsymbol{B}, \boldsymbol{P}, \boldsymbol{J}, \boldsymbol{C}\}$ collected by Algorithm A1.

```
▷ B[a_i, a_j] is the frequency of a_i occurring before a_j
▷ P[a_i, a_j] is the frequency of a_i occurring after a_j
▷ J[a_i, a_j] is the number of videos where a_i occurs with a_j
▷ C[a_i, a_j] is the number of videos where a_i occurs
Constraints ← {} ;
▷ Empty set of rules
foreach i ∈ {0, 1, ..., N − 1} do
    foreach j ∈ {0, 1, ..., N − 1} do
        if i ≠ j and J[i, j] == 0 then
            ▷ action i is ‘‘backward dependent’’ on j
            Constraints ← append_BD(i, j) ;
        if J[i, j] > 0 and P[i, j] == 0 then
            ▷ action j ‘‘forward cancels’’ j
            Constraints ← append_FC(j, i) ;
        if i ≠ j and J[i, j]/C[j] == 1 then
            ▷ action j implies i
            Constraints ← append_Ip(j, i) ;
        if i ≠ j and J[i, j] == 0 then
            ▷ action i and j is exclusive
            Constraints ← append_Ex(i, j) ;
return Constraints;
```

juice, etc. Breakfast is therefore different from 50Salads because its actions could be mutually exclusive. Below is a list of actions:

- take_cup
- pour_coffee
- pour_milk
- pour_sugar
- stir_coffee
- spoon_sugar
- add_teabag
- pour_water
- stir_tea
- cut_bun
- smear_butter
- put_toppingOnTop
- put_bunTogether
- take_plate
- take_knife
- take_butter
- take_topping
- cut_orange
- squeeze_orange
- take_glass
- pour_juice
- take_squeezer
- take_bowl
- pour_cereals
- stir_cereals
- spoon_powder
- stir_milk
- pour_oil
- take_eggs
- crack_egg
- add_saltnpepper
- fry_egg
- put_egg2plate
- butter_pan
- cut_fruit
- put_fruit2bowl
- peel_fruit
- stir_fruit
- stirfry_egg
- stir_egg
- pour_egg2pan
- spoon_flour
- stir_dough
- pour_dough2pan
- fry_pancake
- put_pancake2plate
- pour_flour
- SIL

Backward Dependency The following shows a subset of the back dependency constraints, where $\mathrm{BD}(a_i, a_j) = (\mathrm{F}a_i \wedge \mathrm{F}a_j) \rightarrow (\neg a_i \mathrm{W} a_j)$ reads "action a_i is backward dependent on action a_j."

- BD(pour_dough2pan, spoon_flour),
- BD(squeeze_orange, cut_orange),
- BD(add_saltnpepper, take_bowl),
- BD(put_egg2plate, pour_egg2pan),
- BD(stir_coffee, pour_coffee),
- BD(butter_pan, take_eggs),
- BD(pour_dough2pan, butter_pan),
- BD(stir_tea, take_cup),
- BD(pour_milk, pour_coffee),
- BD(pour_juice, take_plate).

Forward Cancellation The following shows a subset of the forward cancellation constraints, where $\text{FC}(a_i, a_j) = (\text{F}a_i \wedge \text{F}a_j) \rightarrow (\neg a_j \text{S} a_i)$ reads "action a_i cancels the future occurrence of a_j."

- FC(pour_dough2pan, pour_flour),
- FC(take_squeezer, take_knife),
- FC(fry_pancake, pour_dough2pan),
- FC(put_buntogether, take_topping),
- FC(pour_dough2pan, take_bowl),
- FC(put_toppingontop, cut_bun),
- FC(stir_fruit, peel_fruit),
- FC(pour_milk, pour_coffee),
- FC(put_buntogether, take_knife),
- FC(stir_coffee, take_cup).

Implication The following shows a subset of the implication constraints, where $\text{Ip}(a_i, a_j) = \text{F}a_i \rightarrow \text{F}a_j$ reads "action a_i implies the occurrence of a_j."

- Ip(fry_pancake, pour_dough2pan),
- Ip(pour_flour, stir_dough),
- Ip(butter_pan, crack_egg),
- Ip(stir_dough, pour_milk),
- Ip(pour_cereals, pour_milk),
- Ip(add_teabag, pour_water),
- Ip(spoon_powder, pour_milk),
- Ip(take_topping, cut_bun),
- Ip(take_butter, smear_butter),
- Ip(stirfry_egg, crack_egg).

Exclusivity The following shows a subset of the exclusivity constraints, where $Ex(a_i, a_j) = \text{F}a_i \rightarrow \neg\text{F}a_j$ reads "if action a_i occurs, action a_j will not occur in the same video".

- Ex(butter_pan, cut_bun),
- Ex(smear_butter, spoon_sugar),
- Ex(take_butter, pour_sugar),
- Ex(stir_cereals, cut_bun),
- Ex(add_teabag, take_butter),
- Ex(take_butter, stir_cereals),
- Ex(put_fruit2bowl, put_pancake2plate),
- Ex(put_fruit2bowl, pour_sugar),
- Ex(cut_bun, spoon_sugar),
- Ex(spoon_flour, add_teabag).

E.2.2 50Salads

50Salads contains 19 actions for a single activity "making salad":

- cut_tomato
- place_tomato_into_bowl
- cut_cheese
- place_cheese_into_bowl
- cut_lettuce
- place_lettuce_into_bowl
- add_salt
- add_vinegar
- add_oil
- add_pepper
- mix_dressing
- peel_cucumber
- cut_cucumber
- place_cucumber_into_bowl
- add_dressing
- mix_ingredients
- serve_salad_onto_plate
- action_start
- action_end

Backward Dependency

- BD(serve_salad_onto_plate, peel_cucumber),
- BD(serve_salad_onto_plate, place_tomato_into_bowl),
- BD(add_dressing, cut_tomato),
- BD(serve_salad_onto_plate, cut_cucumber),
- BD(serve_salad_onto_plate, cut_lettuce),
- BD(add_dressing, add_pepper),
- BD(serve_salad_onto_plate, mix_ingredients),
- BD(serve_salad_onto_plate, add_salt),
- BD(serve_salad_onto_plate, place_cucumber_into_bowl),
- BD(add_dressing, cut_cheese).

Forward Cancellation

- FC(add_dressing, peel_cucumber),
- FC(serve_salad_onto_plate, add_vinegar),
- FC(place_cheese_into_bowl, cut_cheese),
- FC(mix_ingredients, cut_cheese),
- FC(add_dressing, add_oil),
- FC(place_cucumber_into_bowl, peel_cucumber),
- FC(serve_salad_onto_plate, mix_ingredients),
- FC(serve_salad_onto_plate, cut_cheese),
- FC(serve_salad_onto_plate, peel_cucumber),
- FC(add_dressing, place_cucumber_into_bowl).

Implication

- Ip(place_cheese_into_bowl, action_end),
- Ip(cut_cheese, cut_lettuce),
- Ip(add_oil, place_tomato_into_bowl),
- Ip(place_tomato_into_bowl, cut_tomato),
- Ip(mix_dressing, cut_cheese),
- Ip(add_salt, add_pepper),
- Ip(place_lettuce_into_bowl, add_oil),
- Ip(add_salt, place_tomato_into_bowl),
- Ip(place_tomato_into_bowl, action_end),
- Ip(add_oil, cut_tomato).

Exclusivity There are no exclusivity constraints because all actions are from the same activity.

F Action Detection on Charades

One might be curious about whether DTL works without the "one-action-per-frame" assumption used in the main paper. In fact, this assumption emerges from the definition of the action segmentation task rather than DTL itself.

To examine the applicability of DTL without this assumption, we assess DTL using the action detection task on the Charades [49] dataset. Different from the datasets used in the action segmentation task, every frame in Charades can be labeled with zero or multiple actions. The dataset contains 9,848 videos with 157 frame-level action categories. Each video spans about 30 seconds and contains six actions on average. Following the same procedure introduced in Section 3.3, we collected a total of 9,668 constraints. We sampled 2,000 constraints for this experiment.

Table A2: Performances on the Charades dataset.

Task Model		mAP (%)
GRU	Base	20.7
	Base + DTL	21.6
	Gain	**0.9**
TConv	Base	17.2
	Base + DTL	18.3
	Gain	**1.1**

Following the procedures in [44], we use I3D [4] pretrained on Kinetics-400 to extract the frame-level features and mean average precision (mAP) as the performance metric. DTL is assessed using two task models, namely, a GRU with 512 hidden units (abbreviated as GRU), and a three-layer temporal convolutional network (abbreviated as TConv). As shown in Table A2, improvements are observed for both task models when they are trained with DTL. Note that since we aim to show that DTL works across different tasks, we do not use constraints unique to the action detection task. For example, we do not collect constraints about co-occurrences of actions in the same frame. This means further improvement is possible when those absent constraints are collected and enforced. The definition of a more comprehensive set of constraints and their applications in various action analysis tasks are left as future works.