# Learning Causality under Uncertainty for Egocentric Action Anticipation

Antonio Garrido[0000−0002−8219−6777]

Valencian Research Institute for Artificial Intelligence (VRAIN)
Universitat Politecnica de Valencia
Camino de Vera s/n, 46022, Valencia (Spain)
agarridot@dsic.upv.es

**Abstract.** Egocentric action anticipation, as the task to infer future actions based on egocentric observations, is very appealing to discover patterns, conducts and, eventually, action causality. In the real world, the observations are usually uncertain, which means they are incomplete and noisy. Consequently, addressing uncertainty implies some kind of optimization. Traditional approaches develop classification tasks to calculate the actions with the highest probability to appear next (maximization problem), but they do not exploit the cause-and-effect relationships. We present here a simple two-step approach. The first step formulates and solves a constraint optimization problem that learns the action causality from a training dataset. Given an observation segment of actions over a new dataset, the second step uses such causality in a matching process to anticipate the highest scored actions. Additionally, this causality can be used to derive the planning action model. The experiments in the EPIC-KITCHENS dataset, a large-scale egocentric benchmark, show that our approach is competitive in terms of accuracy and recall.

**Keywords:** Causality learning · Action anticipation · Egocentric observations · Uncertainty · Action planning.

## 1 Introduction

Modern wearable devices allow individuals to interact in natural ways. Egocentric approaches, focused on individuals, exploit the information captured by these devices, and have gained popularity as a promising solution for data collection+interpretation in a context of ambient intelligence. This intelligence refers to a digital environment that proactively supports people in their daily routines, with the ultimate goal of discovering and learning how people behave.

Egocentric vision is a field of computer vision that is characterized by the acquisition, annotation and analysis of captured video from the first person perspective. The idea is to concentrate on people's interaction, recognize the executed activities and, eventually, to understand their needs and augment their abilities [5, 7]. This poses several challenges [3, 5]. One video-based challenge, which requires visual reasoning, is to convert the video into a sequence of annotated labels, *aka* actions. Each action is annotated with a label formed by one

verb and zero or more nouns. The verb represents the activity and the nouns are the objects, e.g.: `taking medicine`, `toileting`, `open refrigerator`, `turn-up heat oven`, etc. Another action-based challenge, which requires logic reasoning, is to predict the next action(s) after observing a sequence of actions. Given past observations, anticipating what is going to happen in the near future is a fundamental ability both for humans and intelligent prediction systems [3, 4, 7].

The correctness and completeness of the observations are crucial for successful action anticipation. However, uncertainty is frequent in the observations [7], as they are usually noisy and incomplete. First, since most actions are annotated by humans, noise and contradictions are common in the labels. Different verbs are labeled with the same meaning, e.g., `open` *vs.* `open-to`, or equivalent nouns are used indistinctly, e.g., `fridge` *vs.* `refrigerator`. Second, the observations are incomplete, e.g., many actions `open door` are observed but no `close door` is observed, or vice versa, which leads to missing information. After all, sensors, observations and labeling are inherently uncertain in ambient intelligent settings: some information is observed when it should not (noise), whereas other information is not observed when it should (missing data). Therefore, anticipation approaches must handle uncertainty.

Most approaches in AI address action anticipation as a classification task, where each input sample (i.e., collection of uncertain observations) is believed to belong to one class (i.e., the action to be anticipated). This task is mainly solved via machine learning and pattern recognition techniques. These techniques use complex architectures of convolutional and regression neural networks, and reinforcement learning to predict future actions [2, 9–11, 13, 14]. These techniques have shown successful in action anticipation, but they do not provide the underlying model that discovers and explains why an action is actually anticipated.

Discovering the action causality allows us to learn the action model, which can be used for action anticipation, plan+goal recognition, and AI planning in general. After all, causality is the basis of planning, to decide how to act and plan to reach goals. Although learning the action causality shows a promising approach for action anticipation, to the best of our knowledge, there is no current approach that combines causality and action anticipation.

This paper contributes in several ways to bridge the gap between causality and action anticipation. First, we contribute with a constraint-based formulation, inspired in [8], to model the potential causal relationships of the actions. This formulation is automatically compiled from a series of plans, formed by annotated actions with uncertainty. Second, we contribute with a simple way to learn the action causality by solving a constraint optimization problem that maximizes the number of constraints satisfied over the input plans. Given new observations on actions, we match them with the learned cause-and-effect relationships to anticipate the most promising actions. This way of addressing action anticipation is the third contribution of the paper. Finally, as a collateral contribution, we propose an automated way to derive the planning action model from the action causality, which is very interesting to give further recommendations in proactive systems.

## 2 Formalization and Related Work

### 2.1 AI Planning

Planning is a deliberative task that elaborates a plan of actions to achieve a set of goals. Planning relies on an action model over a set of actions $\mathcal{A}$, where each action is defined as follows:

**Definition 1 (Action).** *An action has a name, a duration, a set of preconditions and a set of effects (denoted as $a \in \mathcal{A}$, $\mathsf{dur}(a)$, $\mathsf{pre}(a)$ and $\mathsf{eff}(a)$, respectively). All the preconditions need to be satisfied for the action $a$ to be executed. After the execution of $a$, all its effects are achieved.*

**Definition 2 (Causal Relationship).** *Given two actions $a_1, a_2 \in \mathcal{A}$, a causal relationship $\langle a_1, p, a_2 \rangle$ represents the fact that $a_1$ supports, as one of its effects, the precondition $p$ required by $a_2$; $p \in \mathsf{eff}(a_1)$ and $p \in \mathsf{pre}(a_2)$.*

In a general planning scenario, an action usually has several (positive+negative) effects. For example, $\mathsf{eff}($`put-in supplement mixer`$)$ =$\{$`in supplement mixer`, ¬`in supplement container`, ¬`mixer empty`$\}$, with the meaning of having the supplement in the mixer, not having the supplement in its container, and not having the mixer empty. However, in a context of egocentric action anticipation, the granularity level required to represent the full knowledge is unknown or, simply, ignored. In this context, we want to know that the supplement is finally in the mixer, but not if it was previously stored in a container (box or jar) or in a refrigerator. This means that the effects of an action $a$ are usually simplified to one, and denoted as $\mathsf{eff}(a) = a$`-done`, i.e., $a$ is already done. Consequently, the causal relationship $\langle a_1, a_1$`-done`$, a_2 \rangle$ can be simplified to $\langle a_1, a_2 \rangle$, meaning that $a_1$ is required by $a_2$ or, alternatively, that $a_1$ supports $a_2$. For simplicity, this is the notation that we use for causal relationships in the rest of the paper.

**Definition 3 (Plan).** *A plan $\pi = \{\langle t_{start_1}, t_{end_1}, a_1 \rangle, \langle t_{start_2}, t_{end_2}, a_2 \rangle \ldots \langle t_{start_n}, t_{end_n}, a_n \rangle\}$ is a sequence of actions, where each $\langle t_{start_i}, t_{end_i}, a_i \rangle$ stands for an instance of the action $a_i \in \mathcal{A}$ that is executed in interval $[t_{start_i}, t_{end_i}]$. Different instances, or repetitions, of the same action can happen at different times. The plan starts from an initial state and must reach a set of goals.*

Fig. 1 depicts the fragment of a plan, which contains the actions that are necessary to satisfy all the causal relationships. For example, the relationship $\langle$`open supplement`, `put-in supplement mixer`$\rangle$ means that the first action is necessary, as a supporter, to satisfy the second one.

When the action model is unknown, reasoning on a training dataset that consists of a series of plans allows us to learn the underlying causality to eventually derive the action model.

**Definition 4 (Learning Causality).** *Let us consider a collection of $n$-plans, namely $\Pi_n$, that execute actions in $\mathcal{A}$. Learning causality is the task denoted by $\mathcal{L}(\Pi_n)$ to find a solution, as a set of causal links: $sol(\mathcal{L}(\Pi_n)) = \{\langle a_i, a_j \rangle\}$ with $a_i, a_j \in \mathcal{A}$.*

```
[32059,34320]: open supplement
[33008,34580]: close package
[33554,35543]: put-in supplement mixer
[34096,35734]: open refrigerator
[37080,40047]: put-in berry refrigerator
[39072,41729]: close supplement
[39554,40899]: close drawer
[40003,41379]: close refrigerator
```

**Fig. 1.** Fragment of a plan of actions. The numbers in brackets represent the start and end times (in milliseconds).

In an ideal scenario with no uncertainty, $sol(\mathcal{L}(\Pi_n))$ would be complete and all its causal links would be satisfied in $\Pi_n$. In a real scenario, however, we cannot guarantee the completeness nor correctness of $sol(\mathcal{L}(\Pi_n))$: some causal links may be missing because actions in $\Pi_n$ are incomplete, and others may be incorrect because actions in $\Pi_n$ are noisy. Therefore, the causal links in $sol(\mathcal{L}(\Pi_n))$ are those that are most satisfied in $\Pi_n$.

Deriving the action model from $sol(\mathcal{L}(\Pi_n))$ is straightforward. For every action $a \in \mathcal{A}$, $\mathsf{pre}(a) = \{a_i\} \mid \{\langle a_i, a\rangle\} \in sol(\mathcal{L}(\Pi_n))$, and $\mathsf{eff}(a) = a$-`done`.

### 2.2 Action Anticipation

**Definition 5 (Action anticipation).** *Let us consider a collection of unknown actions $\mathcal{A}?$ in $\pi$, namely as the ground truth actions, executed in $[T_{start}, T_{end}]$. Given an anticipation time $T_{ant}$, i.e., how far in advance to anticipate the actions, and an observation time $T_{obs}$, i.e., the length of the observed segment, the aim of action anticipation is to predict the upcoming actions in $\mathcal{A}?$ by just observing the segment $[T_{start} - (T_{ant} + T_{obs}), T_{start} - T_{ant}]$ of $\pi$.*

As an example, let us consider the plan in Fig. 1, $T_{start} = 37000$ and $T_{end} = 42000$, so $\mathcal{A}?$ =`{put-in berry refrigerator, close supplement, close drawer, close refrigerator}`. If we consider $T_{ant} = 1000$ and $T_{obs} = 2000$, the observed actions are `{open supplement, close package, put-in supplement mixer, open refrigerator}`. Predicting only the first action of the possible actions in $\mathcal{A}?$ is too limited and underestimates the potential of the anticipation task. In consequence, solving this task usually involves returning the TOP-k actions by choosing the $k$-highest scored predictions [7]. For example, if $k$=4, a possible TOP-4 is `{turn-on tap, close refrigerator, open drawer, close drawer}`.

Evaluating the success of the TOP-k anticipated actions is done by measuring the number of matches (hits) in $\mathcal{A}?$. However, restricting a hit to a syntactic match of the entire action, as a *verb+nouns pair*, is very difficult due to multiple reasons [7], such as the lack of accurate labels, the ambiguity of actions, etc. To mitigate this difficulty, the works in the literature propose to measure not only the entire action, but also the verb and nouns separately [5]. We show this in

**Table 1.** TOP-4 anticipated actions against the ground truth actions in $\mathcal{A}?$. Three possible hits (True/False) are shown: for verb, nouns and action (verb+nouns pair).

| | $\mathcal{A}?$ | | | |
|---|---|---|---|---|
| TOP-4 | `put-in berry refrigerator` | `close supplement` | `close drawer` | `close refrigerator` |
| 1. `turn-on tap` | FFF | FFF | FFF | FFF |
| 2. `close refrigerator` | FTF | TFF | TFF | TTT |
| 3. `open drawer` | FFF | FFF | FTF | FFF |
| 4. `close drawer` | FFF | TFF | TTT | TFF |

Table 1, where every TOP-4 anticipated action is matched against every action in $\mathcal{A}?$. For example, `turn-on tap` does not match against any action in $\mathcal{A}?$ for the verb, nouns and action, so it is always marked as FFF (False, False, False). `close refrigerator` is marked as FTF against `put-in berry refrigerator` (the noun is a hit), TFF against `close supplement` and `close drawer` (the verb is a hit), and TTT against `close refrigerator` (the verb, nouns and action are a hit).

### 2.3 Constraint Optimization

**Definition 6 (Constraint Optimization Problem (COP)).** *A COP is a tuple $\langle \mathcal{X}, \mathcal{D}, \mathcal{C}, metric(\mathcal{X}) \rangle$, where $\mathcal{X}$ is a set of decision variables, $\mathcal{D}$ represents the domain for each of these variables and $\mathcal{C}$ is a set of constraints among the variables in $\mathcal{X}$ that bound their values in $\mathcal{D}$. $metric(\mathcal{X})$ is a function defined over the variables in $\mathcal{X}$. Solving a COP means to find an evaluation of values in $\mathcal{D}$ to all variables in $\mathcal{X}$ that do not violate any of the constraints in $\mathcal{C}$ and minimizes/maximizes the value of $metric(\mathcal{X})$.*

A COP is very flexible to define soft constraints that do not need to be always satisfied. For example, we can maximize the number of constraints that are finally satisfied.

### 2.4 Related Work

Egocentric action anticipation can be considered as a special case of multi-label classification, into one or more future actions, with missing labels [7]. It is multi-label because multiple future actions can naturally follow a given observation. It contains missing labels because both the observed and the future actions are likely to be uncertain.

In [12], a Hidden Markov Model is used to create a flexible ordered graph that learns the interdependencies between actions to predict an ongoing action. [2] uses Deep Reinforcement Learning to recover a direct mapping from environmental features to build a forecasting model over all the possible activities. [11] models and forecasts long-term goals to incrementally learn spatial and semantic intentions via Inverse Reinforcement and Markov Decision Processes. More recently, the use of training networks has been widespread for action anticipation [6]. [9] trains a model by using standard cross entropy and then tunes it with class

reweighting. [10] uses a causal transformer decoder by using soft labels, learned by the teacher model, as knowledge to guide the student network to learn the information for anticipation. [14] proposes a Vision Swin Transformer which is trained, as the action anticipation base model, with the objective of finding the maximum anticipation score for the future actions. [13] uses recurrent networks to model the sequential structure, as a graph with a set of vertices and edges, to learn the edge connectivity.

Although some works for action anticipation learn the interdependencies and connectivity between actions [12, 13], they do not infer the action model that postulates the causality. The action model is not only useful for action anticipation, but also for plan and goal recognition, which are key in proactive systems. Therefore, learning the causality allows us to better understand the actions. The way in which we learn the causality is a little inductive (it makes inferences based on the observations) and a little abductive (it forms a probable conclusion from those observations). Unlike other works that rely on training networks, we do not need a validation phase. This simplifies our approach, requires less memory consumption and makes it more scalable.

## 3 Proposed Approach

Our approach consists of two steps. First, we create an intuitive constraint-based formulation to learn the action causality. Second, we match the actions of a plan *w.r.t.* the action causality to anticipate the most promising future actions.

### 3.1 Step 1. Learning the Action Causality

We learn the causality of actions in $\mathcal{A}$ by reasoning on the instances in $\Pi_n$ and by formulating one COP per action $a \in \mathcal{A}$. Thus, we iterate for each $a \in \mathcal{A}$.

*Variables and Domains.* First, for every instance $\langle t_{start_i}, t_{end_i}, a_i \rangle \in \Pi_n$, we create two variables: $\mathsf{t_{start}}(a_i), \mathsf{t_{end}}(a_i) \in \mathbb{Z}^+$, with its start and end time, respectively. This models the dataset $\Pi_n$, so it is equal no matter the action $a \in \mathcal{A}$.

Second, for every pair of instances $\langle t_{start_i}, t_{end_i}, a_i \rangle, \langle t_{start}, t_{end}, a \rangle \in \Pi_n$, being $a_i \neq a$, we create a Boolean variable to model the potential causal link for $a$: $\mathsf{clink}(a_i, a)$. This variable represents whether the causal link $\langle a_i, a \rangle$ is satisfied or not, being True=1 and False=0.

*Constraints.* The constraint for learning each causal link is straightforward. For every variable $\mathsf{clink}(a_i, a)$, we need to post:

if $(\mathsf{t_{start}}(a) - \mathsf{t_{end}}(a_i) \in [min\_value, max\_value])$ then $\mathsf{clink}(a_i, a) = True$ else $\mathsf{clink}(a_i, a) = False$

This models the constraint to be satisfied for $\langle a_i, a \rangle$: if $a_i$ supports $a$, $a_i$ must be executed in a temporal window before $a$, which can be easily adjusted. In our approach, we use a temporal window of 5000ms. Since the actions are noisy and the times are not always precisely annotated, we define $min\_value$=-1000 and $max\_value$=4000. These two values can be adjusted to modify the number of potential causal links to be satisfied by the constraint.

*Metric.* The metric of this COP is defined to find the causal links that are most satisfied for $a$, namely $sol(\mathcal{L}(\Pi_n), a) = \{\langle a_i, a\rangle\}$. This involves solving the general problem that maximizes the expression:

$$sol(\mathcal{L}(\Pi_n), a) = \{\langle a_i, a\rangle\} \mid \text{argmax} \sum_{\forall a_i \in \mathcal{A}} \mathsf{clink}(a_i, a)$$

Note that learning the causal links for $a$ is equivalent to learning its preconditions: $\mathsf{pre}(a) = \{a_i\} \mid \{\langle a_i, a\rangle\} \in sol(\mathcal{L}(\Pi_n), a)$, thus forming the action model. After creating and solving the COP for every action $a \in \mathcal{A}$, we obtain $sol(\mathcal{L}(\Pi_n)) = \bigcup_{\forall a \in \mathcal{A}} sol(\mathcal{L}(\Pi_n), a)$.

*A Simple Example.* Let us consider four actions $\{a, b, c, d\} \in \mathcal{A}$ with the following number of instances of each action in $\Pi_n$: $a = \{a_1, a_2, a_3, a_4\}$, $b = \{b_1, b_2, b_3, b_4, b_5\}$, $c = \{c_1, c_2, c_3\}$ and $d = \{d_1, d_2, d_3, d_4, d_5\}$. There are 4+5+3+5 =17 instances, so we need 17 $\mathsf{t_{start}}$ and 17 $\mathsf{t_{end}}$ variables.

The COP for action $a$ has the 17+17=34 previous variables, plus 20+12+20=52 $\mathsf{clink}$ variables, that is, 5*4=20 $\mathsf{clink}(b_i, a_j)$, 3*4=12 $\mathsf{clink}(c_i, a_j)$ and 5*4=20 $\mathsf{clink}(d_i, a_j)$ variables. Consequently, we need to post 52 constraints.

After observing the times in $\Pi_n$, let us assume that there are 2 $\mathsf{clink}(b_i, a_j)$, 3 $\mathsf{clink}(c_i, a_j)$ and 3 $\mathsf{clink}(d_i, a_j)$ variables that are True, so the max number of causal links that are satisfied in the metric is 3. Consequently, $sol(\mathcal{L}(\Pi_n), a) = \{\langle c, a\rangle, \langle d, a\rangle\}$, which means that $\mathsf{pre}(a) = \{c, d\}$. Note that $\langle b, a\rangle$ is not learned because it is not satisfied the max number of times.

Analogous COPs will be created for action $b$ (with 17+60 variables), $c$ (with 17+42 variables) and $d$ (with 17+60 variables) to obtain $sol(\mathcal{L}(\Pi_n), b)$, $sol(\mathcal{L}(\Pi_n), c)$ and $sol(\mathcal{L}(\Pi_n), d)$, respectively. With the union of these, we obtain $sol(\mathcal{L}(\Pi_n)$.

Although learning the action causality implies solving one COP per action (action $a \in \mathcal{A}$, not instance $a_i \in \Pi_n$), its size and complexity remains small. Actually, it is more scalable to solve multiple small COPs than one large COP.

### 3.2 Step 2. Anticipating Actions

We use the action causality to anticipate actions with an anticipation time $T_{ant}$. Given a collection of actions executed in an observation segment of length $T_{obs}$, denoted as $\mathcal{A}_{obs} = \{a_i\}$, we match every $a_i$ with the first component of every causal link in $sol(\mathcal{L}(\Pi_n)) = \{\langle a_i, a_j\rangle\}$ to anticipate the actions in a multiset $\{a_j\}$. It is a multiset because the same $a_j$ can appear more than once: multiple instances of $a_i$ can be observed in $\mathcal{A}_{obs}$, or different $a_i$ in $\mathcal{A}_{obs}$ can support the same $a_j$. The number of occurrences of $a_j$ in the multiset is defined as its score.

Although many actions $\{a_j\}$ might be potentially anticipated, we return only those with the $k$-highest score. In other words, the step 2 returns a set with the TOP-k anticipated actions.

*A Simple Example.* Let us consider $sol(\mathcal{L}(\Pi_n)) = \{\langle a, b\rangle, \langle a, c\rangle, \langle b, c\rangle, \langle b, d\rangle\}$. If $\mathcal{A}_{obs} = \{a\}$, the multiset is $\{b, c\}$ and either $b$ or $c$ can be the TOP-1 anticipated action. If $\mathcal{A}_{obs} = \{a, b\}$, the multiset is $\{b, c, c, d\}$ and $c$ will be the unique TOP-1 anticipated action.

**Table 2.** Size of the datasets used for training and testing.

| Dataset | Participants | Verbs | Nouns | Actions/Unique | Instances |
|---|---|---|---|---|---|
| Training. Learning | 32 | 856 | 2191 | 13656/8330 | 62991 |
| Testing. Overall | 32 | 333 | 719 | 3359/2145 | 9142 |
| Testing. Unseen Participants | 2 | 154 | 75 | 539/364 | 1012 |
| Testing. Tail Classes | 32 | 294 | 522 | 1755/1231 | 3047 |

## 4 Experimental Evaluation

We evaluate our approach on the EPIC-KITCHENS dataset [3–5], a large-scale egocentric video benchmark recorded in kitchen environments. The participants annotate their own videos, thus reflecting their true intentions. Due to the manual process for the annotation, noise and contradictions are common.

### 4.1 Setup

We use one dataset for training (learning causality in step 1) and one for testing (action anticipation in step 2). Unlike other methods that rely on neural networks, our approach does not need an additional validation dataset.

The results in literature show the verb/noun/action separately. The testing dataset is managed as three different subsets, reported also separately. *Overall* contains all data available for testing. *Unseen Participants* contains only participants not present in the training dataset, to extrapolate the causality learned to new participants. *Tail Classes* is defined as the set of the smallest classes (for verbs and nouns) whose instances account for 20% of the instances in training. A tail action class contains either a tail verb class or a tail noun class.

Table 2 shows the size of all datasets, in terms of participants, verbs, nouns, different actions, unique actions that only appear once, and total instances. For example, the training dataset has plans of 32 participants. There exist 856 verbs, 2191 nouns, 13656 different actions, 8330 of which are unique, and 62991 instances in total. In the training dataset, the proportion between unique actions and actions gives us an idea of how good or bad the learning can be. In an ideal scenario, we expect the actions to appear multiple times to build an accurate action model. However, having 61% of unique actions (8330/13656) is not particularly good for learning because the learned causal links are not very informative. This is an important limitation, inherent to the training dataset, because the maximization problem to get the causal links may return incorrect results. If one action rarely appears (just once or twice) in the training dataset, and considering that the dataset is uncertain, the learned causal links may be erroneous. For example, let us imagine that the action `open refrigerator` of Fig. 1 is unique. The max number of causal links that are satisfied in the metric is 1, thus learning $\{\langle$`open supplement`, `open refrigerator`$\rangle, \langle$`close package`, `open refrigerator`$\rangle\}$, which might not be very helpful for the anticipation of `open refrigerator`.

In our experiments, step 1 always learns the action causality from the entire training dataset. For the action anticipation in step 2, we divide each participant's testing plan in 25 segments of equal length, thus performing 25 action

anticipation tasks per participant. The anticipation and observation times are $T_{ant}$=1000ms and $T_{obs}$=3000ms, respectively. We return average results.

We use an Intel i5-6400 @ 2.70GHz with 8GB of RAM. We use Choco[1] to solve our COPs. The solving time for step 1 is around 40s. The step 2 does not require the use of Choco and its solving time remains below 2s per dataset, which shows the low time consumption of our approach.

### 4.2 Evaluation of Accuracy and Recall

Let TP, FP, TN, FN be True Positive, False Positive, True Negative and False Negative, respectively. In TP the element is anticipated when it should (it matches with the ground truth class), and in TN the element is not anticipated because it should not. In FP the element is anticipated when it should not, and in FN the element is not anticipated when it should. TOP-k accuracy is defined as TP/(TP+FP). The divisor is TP+FP$\leq$k, since it is possible to anticipate fewer elements than k. TOP-1 accuracy may underestimate the score [7], so higher values of k are common in the literature, such as k=5.

Table 3 shows the TOP-1 and TOP-5 accuracy of our approach *vs.* the six most successful approaches investigated in [7]. They include verb-noun marginal cross entropy loss functions to explicitly take into account the future actions, which improve their anticipation results. Such functions, while maintaining the focus on actions, leverage the uncertainty by the verb+nouns representation. In short, the approaches that are proposed are: i) VN-CE, which uses the posterior probability distributions of verbs and nouns independently; ii) VNMCE, which uses Temporal Segment Networks (TSNs) to improve the anticipation; iii) SVM-TOP3 and SVM-TOP5, which combine Support Vector Machines (SVMs) and TSNs trained with the Smooth TOP-3 and TOP-5 methods, respectively; and iv) VNMCE+T3 and VNMCE+T5, which use TSNs combining the TOP-3 and TOP-5, respectively, entropy loss methods.

We do not use entropy loss functions and, in principle, our results should be less accurate than other approaches. Despite this, as shown in Table 3, our approach is competitive for TOP-1. However, exploiting loss functions targeted to the optimization of higher TOP-k scores improves the results of the accuracy in the six approaches, and makes ours less competitive for TOP-5. How to improve our accuracy in this situation still requires future research.

Accuracy is an indicator of the anticipation success, but when the dataset is unbalanced (e.g., the actions are not uniformly distributed, are missing or have noise), recall is a better indicator. After all, it is impossible to assess whether an anticipated action that is not in the ground truth class is correct or not, as the ground truth may be incomplete [7]. Recall evaluates the fraction of cases in which the ground truth class is anticipated by the k predicted actions. TOP-k recall is defined as TP/(TP+FN), where TP$\leq$k.

[1] Choco (`www.choco-solver.org`) is an open-source Java library for constraint programming.

**Table 3.** TOP-1/TOP-5 accuracy comparison. The results are taken from [7], which only provide the values for the Overall dataset. The best scores are in bold face.

| | Verb | Noun | Action |
|---|---|---|---|
| VN-CE [5] | **0.32**/**0.78** | 0.16/**0.40** | 0.06/0.17 |
| VNMCE | 0.27/0.74 | 0.15/0.39 | 0.10/0.25 |
| SVM-TOP3 [1] | 0.26/0.73 | 0.16/0.38 | 0.11/0.25 |
| SVM-TOP5 [1] | 0.25/0.69 | 0.15/0.37 | 0.10/0.24 |
| VNMCE+T3 | 0.28/0.74 | 0.16/0.39 | 0.10/**0.26** |
| VNMCE+T5 | 0.27/0.74 | 0.16/0.39 | 0.11/**0.26** |
| Ours | **0.32**/0.30 | **0.27**/0.25 | **0.20**/0.14 |

**Table 4.** TOP-5 recall for the three testing datasets. The results are taken from [6], which only provide the values for TOP-5. The best scores are in bold face.

| | Overall | | | Unseen | | | Tail | | |
|---|---|---|---|---|---|---|---|---|---|
| | Verb | Noun | Action | Verb | Noun | Action | Verb | Noun | Action |
| SCUT [10] | 0.38 | **0.42** | 0.20 | 0.28 | **0.37** | 0.18 | 0.32 | **0.36** | 0.17 |
| NVIDIA-UNIBZ [13] | 0.30 | 0.38 | 0.20 | 0.23 | 0.35 | 0.16 | 0.23 | 0.31 | 0.17 |
| ICL-SJTU [9] | 0.42 | 0.36 | 0.20 | 0.33 | 0.27 | 0.16 | **0.41** | 0.33 | 0.17 |
| PCO-PSNRD [14] | 0.31 | 0.41 | 0.19 | 0.26 | 0.35 | 0.16 | 0.25 | 0.35 | 0.16 |
| Ours | **0.44** | 0.36 | **0.25** | **0.55** | 0.36 | **0.55** | 0.20 | **0.36** | **0.20** |

Table 4 shows the TOP-5 recall of our approach *vs.* the four most successful approaches reported in [6], which have been introduced in Section 2.4.[2] Our approach is very competitive, particularly in this order: Unseen, Overall and Tail datasets. The reason for this order relies in the proportion of new actions (present in testing but not in the training dataset) divided by the number of actions in the testing dataset. If an action does not appear in the training dataset, no causality can be learned for it. In an ideal scenario, we expect low values for this proportion. In our experiments, the number of new actions (and proportion) is 250 (250/539=46%), 1600 (1600/3359=48%) and 986 (986/1755=56%) in the Unseen, Overall and Tail datasets, respectively. The size of the dataset might also affect the learning, reducing the variability of the observed actions, but lower proportion values have a better impact in learning than small datasets.

### 4.3 Using the Action Model for Planning

Learning the causality allows us to derive the action model. In essence, this means that we can automatically build the PDDL (Planning Domain Definition Language) action model directly from the observation of the plans of the training dataset, without the necessity of a human planning expert that manually defines the semantics of the actions. Although the action model is not necessarily complete, it is a good starting point (obviously, better than starting from scratch) that helps reduce the expert effort in a context of proactive systems.

[2] These are the results of the 2022 challenge for action anticipation. To our knowledge, the detailed results of the 2023 challenge are not published yet in `https://epic-kitchens.github.io/2023#results`.

```
(:durative-action have-food                          |    [0,280]: wash bowl
 :parameters ()                                      |    [0,412]: open box
 :duration (= ?duration 537)                         |   [0,1019]: tear wrapper
 :condition (and (at start (add-water_DONE))         |[1020,1323]: add water
           (at start (remove-wrapper_DONE)))         |[1324,1784]: remove wrapper
 :effect (at end (have-food_DONE)))                  |[1785,2322]: have food
```

**Fig. 2.** PDDL action that has been automatically compiled for `have-food` (left), and a recommended plan that uses it (right).

As an example from the EPIC-KITCHENS training dataset, we have learned the semantics for the PDDL action `have-food`, with two preconditions and one effect, depicted in Fig. 2. The duration of the action is calculated as a mean value, because the annotation on the start/end times is typically noisy.

In the future, if one participant is interested in having some food, and given his/her initial information, one potential plan is shown in Fig. 2. The plan may be incomplete, because the participant may prefer a different way of having food or his/her kitchen configuration may be different, but it is a valuable and proactive recommendation to start with, according to other participants' behavior.

## 5 Conclusions and Lessons Learned

Predictive and proactive systems are very interesting in ambient intelligence contexts, such as for notification purposes, to foresee the occurrence of possibly dangerous actions, to act preemptively, etc. Egocentric action anticipation is key for the success of these systems. However, action anticipation must face the problem of dealing with incomplete and noisy observations.

This work has proposed a two-step approach. The first step solves a COP to learn the action causality, which allows us to derive the action model. The second step uses the action causality to anticipate future actions. Our approach is simple but effective, and has allowed us to learn several lessons. First, the constraint-based formulation is flexible to support different types of uncertainty; e.g., actions that may be missing, $\mathsf{t}_{\mathsf{start}}$ or $\mathsf{t}_{\mathsf{end}}$ values that are not fully known, etc. Second, it also allows us to learn *longer transitive* causal relationships; e.g., rather than learning $\mathsf{clink}(a, b)$ and $\mathsf{clink}(b, c)$, learning $\mathsf{clink}(a, c)$ instead for longer action anticipation. This, together with the fact that we can derive the action model from the action causality, facilitates the prediction of *the next of the next* actions (trajectory forecasting), useful for goal detection over a longer time horizon [2, 11]. Third, it solves multiple small maximization problems (one per action), rather than one large problem, which helps improve scalability. Fourth, our approach supports multiple configurations. We have run additional experiments for action anticipation with the 16 following configurations, resulting from: testing plans divided in {25, 30, 35, 40} segments, $T_{ant}$={1000ms, 2000ms}, and $T_{obs}$={2000ms, 3000ms}. In the Overall dataset, which is the most general dataset, there are no differences among the configurations. However, in the Unseen and Tail datasets, which are more specific, we have found some differences.

Fifth, action anticipation relies on the previous learning of the action causality, and this is impossible when actions do not appear in the training dataset. Although this seems obvious, it represents our main limitation.

**Acknowledgments.** This work has been partially supported by grant PID2021-127647NB-C22 funded by MCIN/AEI/10.13039/501100011033 and by "ERDF A way of making Europe".

**Disclosure of Interests.** No competing interests are to be declared.

## References

1. Berrada, L., Zisserman, A., Kumar, M.P.: Smooth loss functions for deep top-k classification. In: Int. Conf. on Learning Representations. pp. 1–25 (2018)
2. Bokhari, S., Kitani, K.: Long-term activity forecasting using first-person vision. In: Asian Conf. on Computer Vision. pp. 346–360 (2016)
3. Damen, D., Doughty, H., Farinella, G., Fidler, S., Furnari, A., Kazakos, E., Moltisanti, D., Munro, J., Perrett, T., Price, W., Wray, M.: The EPIC-KITCHENS dataset: collection, challenges and baselines. IEEE Trans. on Pattern Analysis and Machine Intelligence **43**(11), 4125–4141 (2021)
4. Damen, D., Doughty, H., Farinella, G., Furnari, A., Kazakos, E., Ma, J., Moltisanti, D., Munro, J., Perrett, T., Price, W., Wray, M.: Rescaling egocentric vision: collection, pipeline and challenges for EPIC-KITCHENS-100. Int. Journal of Computer Vision **130**, 33–55 (2022)
5. Damen, D., Doughty, H., Farinella, G.M., Fidler, S., Furnari, A., Kazakos, E., Moltisanti, D., Munro, J., Perrett, T., Price, W., Wray, M.: Scaling egocentric vision: the EPIC-KITCHENS dataset. In: Computer Vision. pp. 753–771 (2018)
6. Damen, D., Fragomeni, A., Perrett, T., Whettam, D., Wray, M., Zhu, B., Furnari, A., Farinella, G.M., Moltisanti, D.: EPIC-KITCHENS-100-2022 challenges report. (2022)
7. Furnari, A., Battiato, S., Farinella, G.: Leveraging uncertainty to rethink loss functions and evaluation measures for egocentric action anticipation. In: Computer Vision. pp. 389–405 (2019)
8. Garrido, A.: A unified constraint-based approach for plan and goal recognition from unreliable observations. Knowledge-Based Systems **278**, 110895 (2023)
9. Gu, X., Guo, Y., Li, Z., Qiu, J., Lo, B., Yang, G.Z.: LTDS: ICL-SJTU submission to EPIC-kitchens action anticipation 2022. pp. 32–35 (2022)
10. Jiang, Z., Ding, C.: 1st place solution to the EPIC-kitchens action anticipation challenge 2022. pp. 23–25 (2022)
11. Rhinehart, N., Kitani, K.M.: First-person activity forecasting with online inverse reinforcement learning (2017)
12. Soran, B., Farhadi, A., Shapiro, L.: Generating notifications for missing actions: don't forget to turn the lights off! In: IEEE Int. Conf. on Computer Vision. pp. 4669–4677 (2015)
13. Tai, T.M., Lanz, O., Fiameni, G., Wong, Y.K., Poon, S.S., Lee, C.K., Cheung, K.C., See, S.: NVIDIA-UNIBZ submission for EPIC-KITCHENS-100 action anticipation challenge 2022. pp. 26–31 (2022)
14. Yamamuro, Y., Takenaka, S., Sato, Y., Fujimatsu, T.: Technical report for EPIC-100 action anticipation challenge 2022. pp. 36–40 (2022)