Explainable Activity Recognition in Videos using Dynamic Cutset Networks

Chiradeep Roy*, Mahsan Nourani‡, Mahesh Shanbagh*, Samia Kabir†, Tahrima Rahman*, Eric D. Ragan‡, Nicholas Ruozzi*, and Vibhav Gogate*

*: The University of Texas at Dallas, ‡: University of Florida, Gainsville, †: Texas A&M University

[Draft: To Appear in IUI 2019 Workshop on Explainable Smart Systems (ExSS)]
*****Do not Distribute*****

Abstract

We consider the following activity recognition task: given a video, infer the set of activities being performed in the video and assign each frame to an activity. Although this task can be solved accurately using existing deep learning techniques, their use is problematic in *interactive settings*. In particular, deep learning models are black boxes: it is difficult to understand how and why the system assigned a particular activity to a frame. This reduces the users' trust in the system, especially in the case of end-users who need to use the system on a regular basis. We address this problem by feeding the output of our proposed deep learning model to a tractable, interpretable probabilistic graphical model called dynamic cutset networks and then performing joint inference over the two. The key benefit of our proposed approach is that deep learning helps achieve high accuracy while cutset networks because of their poly-time probabilistic reasoning capabilities make the system explainable. We demonstrate the efficacy of our approach using conventional evaluation measures such as the Jaccard Index and Hamming Loss as well as a preliminary human subjects study.

1 Introduction

Activity recognition is an important task in video content analysis as it forms a basis for comprehending and semantically understanding the video itself. For instance, when a person is shown a random video clip without any prior information about its type or content, he/she typically uses activity recognition to help aid his/her understanding of the video. If the person is shown a cooking video, for example, he/she might notice certain activities such as eggs being taken out of a refrigerator, being beaten, and then being scrambled on a frying pan. These activities help the viewer conclude that the video is about "making scrambled eggs." If, on the other hand, a person in the video takes an egg out of a refrigerator, throws that egg at another person, and then laughs, then the observer would conclude that the video is about a "prank between friends."

In this paper, we focus on the following activity recognition task: given a video and a predefined set of activities (including 'None'), assign an activity to each frame of the video. Recent advances in machine learning, specifically, deep learning, can be leveraged to accurately solve this task, assuming that ample labeled (video) data is available for training. Despite high accuracy, a hallmark of many deep learning techniques, the learned algorithms/models are not easily "interpretable" by developers and end-users. In particular, most deep learning methods return a black box solution from which it is difficult to ascertain the reasoning behind the provided answers. This lack of interpretability is often undesirable, especially for solving interactive human-machine tasks (cf. Kulesza *et al.* [2015]) where the user seeks to solve a task or make decisions with the aid of a machine learning (ML) system. In such cases, the users need to trust the predictions of the system and be able to easily determine when the ML system is (typically) correct and when it is not. To facilitate this, the system should be such that both its functioning and the reasoning behind its actions are clear to the users, i.e., the system should be explainable.

The purpose of this paper is to describe a model for activity recognition that is interpretable, accurate, and explainable and to show that it helps improve the users' trust and understanding of the system. Our model has two layers. The top layer, with which a user interacts, is a tractable, interpretable probabilistic graphical model (cf. [Koller and Friedman, 2009]), specifically a cutset network [Rahman *et al.*, 2014]. Unlike conventional graphical models such as Bayesian and Markov networks in which probabilistic inference is NP-hard in general and inaccurate in practice, cutset networks are desirable in that they admit accurate linear-time inference and often have the same generalization performance as Bayesian and Markov networks [Rahman *et al.*, 2014; Rooshenas and Lowd, 2014; Liang *et al.*, 2017; Di Mauro *et al.*, 2015a]. The bottom layer of our model is a deep neural network that yields accurate predictions, which are fed into the cutset network layer. A possible interpretation of this model is that the deep learning layer provides noisy sensory inputs to the cutset network layer, which in turn removes the noise and provides explainability. The latter is possible in a cutset network but not in a neural network because the cutset network is a tractable model in that it can yield poly-time explanations by performing (abductive) probabilistic inference over the learned network. To model temporal aspects in video, we further refine this model, propose a novel temporal probabilistic modeling framework called dynamic cutset networks, and show that it improves the estimation accuracy.

We experimentally demonstrate the efficacy of our approach

by building an interactive visual interface and an ML system based on dynamic cutset networks for activity recognition for the Textually Annotated Cooking Scenes (TaCOS) dataset [Regneri *et al.*, 2013]. The purpose of building the visual interface and ML system is two-fold. First, we want to show that we can create a working prototype for an explainable AI system that not only performs accurate activity recognition in videos but can also generate human understandable explanations. Second, we use the resulting system as the basis for human subjects studies of how different types of explanations affect users' trust and understanding of ML models.

2 Related Work

This effort is closely related to the work of Rohrbach *et al.* [2014]; Donahue *et al.* [2015] on generating a semantic representation from videos at an activity level using deep learning architectures. Instead of generating sentences in natural language however, we assign a number of pre-defined labels divided into categories. Related efforts have considered the task of dense captioning [Krishna *et al.*, 2017], i.e., generating summaries of texts from particular segments. Song *et al.* [2016] attempted to create captioning methods that require minimum supervision on the TaCOS dataset. Duan *et al.* [2018] attempted to combine caption generation and sentence localization to feed off of each other to create a weakly supervised training model. These works focus on creating text summaries for video segments, and as is typical of deep learning approaches, they are essentially black boxes. Our approach, on the other hand, aims to create a semantic representation for activities in each frame that can be used to both answer queries easily as well as generate explanations (via probabilistic inference) that justify these answers.

There have also been a number of studies on how trust influences interactions between humans and automated systems, e.g., [Muir, 1994], [Muir and Moray, 1996], [Lee and See, 2004], and [Hoff and Bashir, 2015]. These studies examine factors that might affect the trust of the user in the system, such as showing the past performance of the system and making the working of the system more understandable [Lee and See, 2004]. Hoffman [2017] provides a more detailed taxonomy of such factors and explains how trust is context-specific and dynamic. In other words, trust might vary with respect to specific contexts of automation and must also be maintained over time. Our aim is to be able to control and measure user trust with respect to these systems in order to better understand what kind of explanations influence the trust variable.

Our work on feeding the output of deep neural networks to graphical models (cutset networks in our case) is related to a recent line of works which combine graphical models and neural networks (cf. Johnson *et al.* [2016]). Unlike these works which perform joint learning, the main idea in our work is to treat the output of the neural network as a noisy but highly accurate sensor and then use the graphical model to reduce the noise and provide (common-sense) reasoning capabilities.

3 Activity Recognition with Explanations

The objective of our proposed system is two-fold: (a) perform accurate activity recognition in videos and (b) compile knowledge acquired while learning to recognize activities into an explanatory model. The latter can then be used to explain why a particular activity was assigned to a frame by the system.

3.1 Activity Recognition Task

We define an activity as a triple (*action*, *object*, *location*). The *action* component forms the core part of the activity. These are usually verbs such as wash, cut, slice, open, etc. The *object* component denotes the entities over which the activity is performed. These are generally nouns such as apples, refrigerator, cutting board, knife, etc. Finally, the *location* component tells us *where* the activity is taking place. These are generally location nouns such as kitchen, bathroom, counter top, sink, etc. but can also overlap with the nouns we use as objects. For example, when we "kick open a door," the activity is "kick" and the object is "door," but the same entity might play a different semantic role in a different activity such as if a baby "draws a picture on the door." Here "draw" is the activity, "picture" is the object, and "door" is the location.

For the purposes of our initial system, we make the following simplifying assumptions. First, we train our system on a closed-domain. In this study, we use cooking videos. Second, we assume that only one major activity is taking place per frame (minor activities are ignored). Finally, the action must always be present, while the object and the location are optional. For reflexive actions, such as "walking," the object is "None." In future, we plan on making activities more complex so that we can pose more interesting queries on them.

Users interact with our system by posing so-called *selection questions*: "Did a particular activity defined by the triple (*action*, *object*, *location*) happen in the video?" where object and location can be "None," but action is not allowed to be "None." Examples of selection questions include: (1) "Did the person slice an orange on the counter?" where slice, orange, and counter denote the action, object, and location respectively; (2) "Did the person take out grapes from the refrigerator?" where take out, grapes, and refrigerator denote the action, object, and location respectively; (3) "Did the person open the refrigerator?" where open and refrigerator denote action and object respectively and location is None.

3.2 Explanations

We want the system to generates three types of explanations:

1. **Video Explanations:** When the system answers "yes," we want the system to highlight segments (possibly more than one) of the video where the activity happened. For "no" answers, we want the system to highlight segments where a related activity happened, e.g., carrots were cut in the video but not oranges. If no related activity is found in case of a "no answer," we want the system to output the most likely activity in the video.
2. **Ranked (action, object, location) Triples:** We want the system to display top-k predicted activity triples in the video that are relevant to the query.
3. **Most Probable Entities:** We want the system to display the most probable actions, objects and locations (along with their likelihood) that are relevant to the query.

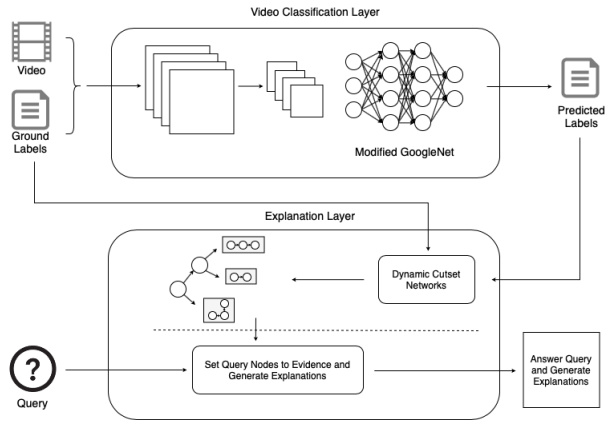

Figure 1: High-level Architecture and Data Processing Pipeline. Our system has two layers: a video classification layer based on deep learning whose output is fed to an explanation layer which is based on cutset networks [Rahman *et al.*, 2014], a tractable interpretable probabilistic model. During the learning phase, the classification layer uses the video and the ground truth (labels) as input and learns a mapping from frames to object, action and location. During the learning phase, the explanation layer uses the labels predicted by the classification layer and ground truth as input and learns a mapping from predicted labels to the ground truth. During the query phase, the system answers questions by performing marginal and MAP inference over the cutset network (in the explanation layer).

4 System Description

Fig. 1 shows a high-level overview of the components of the system and the processing pipeline. We evaluated and tailored the system to the Textually Annotated Cooking Scenes (TaCOS) dataset. Each frame in each video in the dataset is labeled with an *action, object, location* triple. The dataset has 28 labels (our vocabulary) which includes 12 actions, 7 objects, 8 locations and a special label called 'Nothing'. Roughly speaking, the system can be categorized into the following two layers: (a) *video classification layer* which takes as input video frames and a vocabulary file and assigns a set of labels from the vocabulary to each frame; and (b) *explanation layer* which takes the predicted labels from the video classification layer as input, corrects them using a probabilistic model, and outputs (potentially more accurate) labels and explanations.

4.1 Video Classification Layer

For this layer, we used GoogLeNet [Szegedy, 2014], a 22-layer neural network that is pre-trained on the ImageNet dataset [Russakovsky *et al.*, 2015]. It uses a key component called the Inception Module (for details, please refer to [Szegedy *et al.*, 2015]) that creatively uses convolutions of size 1×1 to increase the representation power of the network without increasing the number of parameters. Moreover these 1×1 convolutions use the rectified linear units to circumvent the omnipresent vanishing gradient problems of neural networks as well as avoid the introduction of sparse activations in the hidden layers of the network. Another advantage of GoogLeNet is that its design helps us to abstract features at various scales using different patch sizes for the filters (1×1, 3×3 and 5×5) and then aggregate these features so that the next layer can abstract still higher representation from different scales simultaneously.

To tailor GoogLeNet to our video dataset which has 28 labels, we replaced the topmost softmax layer in GoogLeNet with a fully-connected layer with 28 nodes that use sigmoid cross-entropy loss. The latter is used because it is a standard loss function for solving multi-label classification problems; note that we are solving a multi-label classification task since each frame can have multiple objects and locations. We used the backpropagation algorithm with stochastic gradient descent (SGD) and Adaptive Moment Estimation (Adam) optimizers to further train the pre-trained model on the TaCOS dataset and found that Adam yields the best performance.

4.2 Explanation Layer

In this section, we present dynamic conditional cutset networks (DCCNs), a new tractable temporal probabilistic representation. We will use DCCNs in the explanation layer to: (a) correct errors in the labels predicted by the GoogLeNet at each frame; (b) model the dynamics as well as persistence (activities don't change rapidly between frames) in the video; and (c) provide explanations via abductive poly-time probabilistic inference.

Conditional Cutset Networks

Tractable probabilistic models (TPMs) [Bach and Jordan, 2002; Lowd and Domingos, 2008; Darwiche, 2000] are probabilistic models which admit poly-time posterior marginal inference (MAR)—the task of computing marginal probability distribution over each variable given evidence which is defined as an assignment of values to a subset of variables—and maximum-a-posteriori (MAP) inference—the task of computing the most likely assignment to all non-evidence variables given evidence. Examples of popular TPMs include cutset networks [Rahman *et al.*, 2014], arithmetic circuits [Darwiche, 2000], sum-product networks [Poon and Domingos, 2011] and probabilistic sentential decision diagrams [Bekker *et al.*, 2015]. Although, TPMs are less expressive than intractable (latent) probabilistic models and as a result have slightly poor generalization performance as compared to the latter, their accuracy at test time is often much higher than intractable models. This is because tractable models use exact inference at prediction time while one has to use inaccurate approximate inference algorithms in intractable models.

Cutset networks [Rahman *et al.*, 2014] are TPMs which represent multidimensional joint probability distributions using a rooted (directed) OR tree with tree Bayesian networks at each leaf node of the OR tree. Each OR node in the OR tree is labeled with a variable and just like in decision trees represents conditioning over the variable. Unlike decision trees however, the arcs in the OR tree are labeled with conditional probability of the variable taking the corresponding value given an assignment of values from the root node to the OR node. Tree Bayesian networks at each leaf l represent the conditional distribution $P(\boldsymbol{Y}|path(l))$ where $path(l)$ denotes the assignment from the root to l and $\boldsymbol{Y}$ is the subset of variables not assigned in $path(l)$. MAR and MAP inference over cutset networks can be performed in linear time in the size of the network using cutset conditioning [Pearl, 1988; Dechter and Mateescu, 2007]. However, unlike conventional cutset conditioning algorithms, cutset networks take advantage of context-specific independence, dynamic variable orders and determinism. As a result, cutset networks can compactly represent and perform tractable

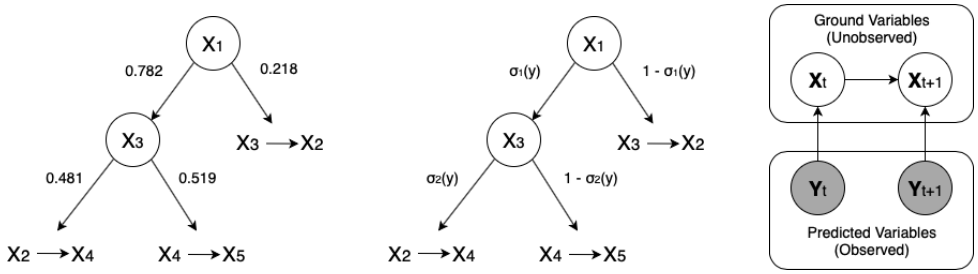

Figure 2: (a) A cutset network over 5 variables $\{X_1, \ldots, X_5\}$. OR nodes are denoted by circles. X_1 is the root node of the OR tree. Left and right arcs emanating from an OR node labeled by X_i indicate conditioning over true and false values of X_i respectively. Arcs emanating from OR nodes are labeled with conditional probabilities. For example, the arc labeled with 0.481 denotes the conditional probability $P(X_3 = 1|X_1 = 1)$. The leaf nodes of the OR tree are tree Bayesian networks. (b) A conditional cutset network (CCN) representing $P(X_1, \ldots, X_5|\boldsymbol{Y})$. Arcs emanating from OR nodes are labeled with (calibrated) classifier functions. For example, the arc from the OR node X_1 to the OR node X_3 is labeled with a logistic regression classifier $\sigma_1(\boldsymbol{y})$. Given $\boldsymbol{Y} = \boldsymbol{y}$, the CCN yields a cutset network having the same structure as the one given in (a) except that the parameters will be computed using $\sigma_1(\boldsymbol{y})$ and $\sigma_2(\boldsymbol{y})$. (c) 2-slice dynamic conditional cutset network. The CCN at time slice t represents $P(\boldsymbol{X}_t|\boldsymbol{Y}_t)$ while the CCN at time slice $t+1$ represents $P(\boldsymbol{X}_{t+1}|\boldsymbol{Y}_{t+1}, \boldsymbol{X}_t)$.

inference in probability distributions that admit high treewidth probabilistic graphical models [Di Mauro *et al.*, 2015a,b; Rahman and Gogate, 2016].

Recently, [Anonymous, 2019] proposed a new framework called conditional cutset networks (CCNs) that extends the cutset networks framework to compactly represent and perform efficient reasoning over conditional probability distributions, namely distributions of the form $P(\boldsymbol{X}|\boldsymbol{Y})$ where $\boldsymbol{X}$ and $\boldsymbol{Y}$ are sets of variables. The main idea in CCNs is to replace all conditional probability distributions $P(X|path(n))$ at each OR node as well as those attached to each variable X in each tree Bayesian network in the cutset network with a calibrated probabilistic classifier [Niculescu-Mizil and Caruana, 2005]. The latter takes an assignment $\boldsymbol{y}$ to $\boldsymbol{Y}$ and $path(n)$ as input and outputs a conditional probability distribution over X, namely $P(X|\boldsymbol{Y} = \boldsymbol{y}, path(n))$ by using only polynomial (in $|\boldsymbol{Y}|$) number of parameters. For example, when we use logistic regression, we have $P(X = 1|\boldsymbol{Y} = \boldsymbol{y}, path(n)) = \sigma(w_0 + \sum_{y_i \in \boldsymbol{y}} w_i y_i)$ where w_i's are the weights (parameters) and σ denotes the sigmoid function. We learn the parameters of the calibrated classifier (e.g., logistic regression) using a subset of the data that is consistent with $path(n)$. CCNs are conditionally tractable in that given an assignment $\boldsymbol{Y} = \boldsymbol{y}$, each (probabilistic) classifier yields a probability distribution over the (class) variable X and thus given $\boldsymbol{Y} = \boldsymbol{y}$, a CCN yields a (tractable) cutset network. (see Fig. 2 for an example).

We learn the structure of CCNs using a conventional cutset networks top-down induction algorithm. This algorithm has two recursive steps. In the base case, if very few examples, remain the algorithm uses the Chow-Liu algorithm [Chow and Liu, 1968] to yield a tree Bayesian network. Otherwise, it heuristically selects a variable to condition on, computes the conditional probability distribution over the variable using the given dataset, and then recurses on the true and false value assignment to the variable. After the structure is learned from data, we learn a classifier at each OR node and each variable at each Bayesian network in the CN using popular classification algorithms such as logistic regression, random forests, and neural networks and pick the best using cross validation.

To use CCNs in our framework, we feed the output of GoogLeNet to the CCN. More formally, let $\boldsymbol{Y}$ denote the set of output nodes of GoogLeNet and $\boldsymbol{X}$ denote the set of true labels at a frame. We use the CCN to model $P(\boldsymbol{X}|\boldsymbol{Y})$ and learn its structure and parameters using a dataset constructed as follows. Each frame in each video is a training example and is composed of true labels ($\boldsymbol{X}$) and labels predicted by GoogleNet ($\boldsymbol{Y}$) with the pixels in the frame as input. At test time, at each frame, we instantiate all the classifiers in the CCN using the predicted labels to yield a cutset network and then perform inference over the cutset network to yield the final set of labels. In other words, the CCN treats the output of the neural network as a noisy sensor (see Fig. 2(a)) and computes a conditional joint probability distribution over the true labels given the predicted (noisy) labels.

Dynamic Conditional Cutset Networks

An issue with CCNs is that they are static and do not explicitly model temporal aspects of video. For instance, we can use persistence, namely objects do not change their position rapidly between subsequent frames to correct prediction errors at a frame by using data from neighboring frames. To address this issue, we propose a novel framework called dynamic conditional cutset networks (DCCNs). Formally, let a video consist of n frames, let $\boldsymbol{X}_i$ and $\boldsymbol{Y}_i$ be the set of true labels and predicted labels (evidence) at frame i. Then, the DCCN represents the following probability distribution:

$$P(\boldsymbol{x}_{1:n}|\boldsymbol{y}_{1:n}) = P(\boldsymbol{x}_1|\boldsymbol{y}_1) \prod_{i=2}^{n} P(\boldsymbol{x}_i|\boldsymbol{y}_{1:i}, \boldsymbol{x}_{1:i-1}), \quad (1)$$

where the notation $\boldsymbol{x}_{1:n}$ (similarly $\boldsymbol{y}_{1:n}$) denotes an assignment of values to all true (predicted) labels in frames 1 to n. We will use the notation $\boldsymbol{X}_{1:n}$ to denote the set $\bigcup_{i=1}^{n} \boldsymbol{X}_i$.

The representation given in Eq. (1) is not compact as n increases. To circumvent this issue, we use two standard assumptions widely used in temporal or dynamic probabilistic models—the 1-Markov and stationarity assumptions [Rabiner,

1989]. Specifically, we assume that each frame is conditionally independent of all frames before it given the previous frame (1-Markov) and all conditional distributions are identical (stationarity). With these assumptions, we can represent $P(\boldsymbol{x}_{1:n}|\boldsymbol{y}_{1:n})$ using

$$P(\boldsymbol{x}_{1:n}|\boldsymbol{y}_{1:n}) = P(\boldsymbol{x}_1|\boldsymbol{y}_1)\prod_{i=2}^{n} P(\boldsymbol{x}_i|\boldsymbol{y}_i, \boldsymbol{x}_{i-1}), \qquad (2)$$

where $P(\boldsymbol{X}_1|\boldsymbol{Y}_1)$ and $P(\boldsymbol{X}_i|\boldsymbol{Y}_i, \boldsymbol{X}_{i-1})$ are conditional cutset networks and $P(\boldsymbol{X}_i|\boldsymbol{Y}_i, \boldsymbol{X}_{i-1})$ is the same for all i.

We learn DCCNs using the following approach. The prior model $P(\boldsymbol{X}_1|\boldsymbol{Y}_1)$ is the same as the CCN described in the previous section. To learn the structure and parameters of $P(\boldsymbol{X}_i|\boldsymbol{Y}_i, \boldsymbol{X}_{i-1})$, we construct the dataset as follows. Each frame in each video is a training example and is composed of true labels at frame i ($\boldsymbol{X}_i$), true labels at frame $i-1$ ($\boldsymbol{X}_{i-1}$) and labels predicted by GoogleNet at frame i ($\boldsymbol{Y}_i$) using the pixels in the frame as input.

Inference over DCCNs can be performed using sequential sampling approaches such as particle filtering and smoothing [Doucet *et al.*, 2000]. Here, we generate k assignments $(\boldsymbol{x}_1^{(1)}, \ldots, \boldsymbol{x}_1^{(k)})$ uniformly at random from $P(\boldsymbol{X}_1|\boldsymbol{y}_1)$, then for each assignment $\boldsymbol{x}_1^{(i)}$ we sample one assignment from $P(\boldsymbol{X}_2|\boldsymbol{y}_2, \boldsymbol{x}_1^{(i)})$, and so on. At the end of the sampling process, we will have k particles from $P(\boldsymbol{X}_{1:n}|\boldsymbol{y}_{1:n})$. The main virtue of DCCNs is that unlike widely used temporal models such as dynamic Bayesian networks [Murphy, 2002], the particles in DCCNs are generated from the posterior distribution $P(\boldsymbol{x}_i|\boldsymbol{y}_i, \boldsymbol{x}_{i-1})$ at each frame. As a result, issues such as particle degeneracy—particles vanish because their weights become too low as i increases—that typical sequential sampling algorithms suffer from will be less severe in DCCNs.

The three explanation types (see section 3.2) can be computed by performing MAP and MAR inference in CCNs and DCCNs. To compute video explanations, we use an ontology that models relationships between activities and objects (e.g., 'chef's knife' is related to 'kitchen knife', 'slice' is related to 'cut', etc.) and display video segments in which the marginal probability of the queried activity or activities related to it (recall that activity is a (*action*, *object*, *location*) triple) is larger than a threshold. Ranked triples over each segment in video explanations are computed by performing k-best MAP inference. Again, the key advantage of CCNs and DCCNs is that k-best MAP inference is linear in k and the number of parameters at each frame. Most probable entities in each highlighted segment are derived as follows. We compute the marginal probability of each entity in each highlighted segment given evidence (via MAR inference) and display the top k most likely ones according to the marginal probability. Note that these inferences are not possible on GoogLeNet or recurrent deep architectures (cf. Donahue *et al.* [2015]) unless we treat the labels as independent entities.

4.3 User Interface

The prototype system uses an interactive visual interface that allows users to load videos, ask queries, and review the model output along with explanations. The goal for the interface design was to limit the amount of model information presented to the user in order to avoid overwhelming users with information. For this reason, the system uses simple visual representations in the form of graphical annotations, textual component lists, and simple bar charts. Figure 3 shows the interface.

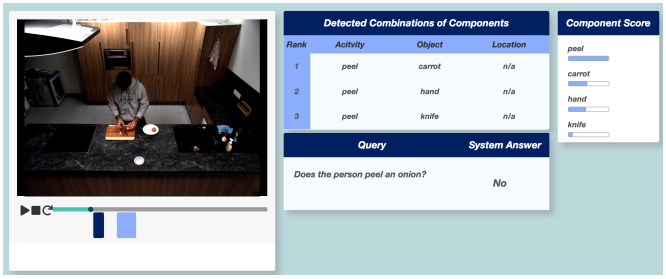

Figure 3: The interactive visual interface allows users to load videos and ask queries. The interface shows the ML system's answer along with explanatory elements for the output. The most relevant portions of the video play time are shown by colored bars beneath the video, and the right side shows detected video components and combinations of components relevant to the video and query.

The interface includes a video player that allows users to watch the selected video to help review and analyze the system's answers to the queries. When a query is submitted, the video player highlights the most relevant segments of the video through visual annotations added under the video play bar (shown as orange and purple bars under the video in Fig. 3). The video player will also automatically jump to the appropriate segment to help users see the video frames most important for determining the output. In addition, the right side of the interface summarizes the detected video components (*action*, *objects*, and *locations*) for the query as well as detected combinations of components. To help users to quickly judge component scores, graphical bars are shown underneath detected components to visually represent the values of the component scores. Users can select different video segments to view the corresponding component scores and combinations from different portions of the video.

5 Experiments

In order to evaluate our system, we designed two experiments using the TACoS video dataset and annotations. We performed a model evaluation to measure how successful the system is at recognizing activities, and we performed a user study to assess system trust and utility.

5.1 Model (Machine Learning) Evaluation

We selected 60313 frames for training and 9355 frames for testing distributed over 17 videos. For each set, we selected a set of ground labels and used the video classification layer to generate the predicted labels. We performed exact inference over CCNs and used particle filtering with 100 particles for inference in DCCNs. We performed the following ablation study: (1) Our system in which the explanation layer is removed (GoogLeNet); (2) Our system which uses (static) conditional cutset networks in the explanation layer (CCNs); and (3) the full system (dynamic CCNs).

Table 1 outlines the accuracy scores for correct activity recognition according to various evaluation metrics. Since pre-

Metric	GoogLeNet	CCNs	Dynamic CCNs
K-1	0.9335	0.9677	**0.9687**
K-2	0.8557	0.8998	**0.9197**
K-3	0.7918	0.7962	**0.8168**
Jaccard Index	0.8608	0.8559	**0.8674**
Hamming Loss	0.1392	0.1286	**0.1160**

Table 1: Accuracy for Activity Recognition on Test Videos. Bold results indicate the best performing model.

dicting each activity correctly is a multilabel classification task, we use K-Group measures to calculate the overall percentage of instances where K labels out of the total number of labels were predicted correctly. We report K-1, K-2, and K-3 since each activity comprises of *action*, *object* and *location*. In addition, we also use standard measures such as the Hamming Loss and the Jaccard Index. We observe that dynamic CCNs are more accurate than CCNs and GoogLeNet.

5.2 Human Subjects Evaluation

We conducted a human-subjects study to evaluate the overall effectiveness of the system for machine-assisted activity recognition. The study was implemented as an online experiment in order to be able to crowdsource the study using Amazon Mechanical Turk (AMT). The study consisted of 80 novice participants with self-reported "low" machine learning knowledge. We arranged a between-subjects study to compare two conditions: (1) No explanations (namely after removing the three explanation types from our visual interface given in Fig. 3); and (2) With explanations (i.e., the full interface). For both groups, the system provided ML answer for given queries.

Participants were tasked with the review of cooking videos from the TaCOS data set, and they had to answer a set of queries about where certain activities or objects were present in each video clip. The study measured participants' abilities to accurately answer each query in a repeated number of trials using the explainable ML system. All participants used the system with the same set of instructions, though specifics of the system and instructions were customized based on the assigned experimental condition (i.e., explanation or no explanation).

For experimental control, the user interface was modified to present a sequence of given queries and videos rather than allow user choice of queries. The participant task was to determine the correct answer to the query as quickly and correctly as possible. For this exercise, we wanted to assess whether participants would be able to rely on the explanations to improve their understanding of the system, which in turn will enable them to answer more accurately. However, the highest base accuracy of the model would not work for this purpose because response accuracy would be too high if participants always agreed with the model output, whereas a lower accuracy would be more useful for evaluating human understanding of the explanation interface. For this reason, we selected a controlled subset of queries to allow all participants to experience the system with a simulated 80% accuracy.

Each participant conducted the task with a total of 20 queries—5 queries for each of 4 videos. The video clips ranged in length from 1.5 to 6 minutes. For the condition with explanations, the interface showed the full interface with the video player, the segments of the video used to determine the answer, component scores, and detected combination of components. Users were allowed to click on the segments to jump to that portion of the video. For each segment, users would see different component scores and detected combination of components in the corresponding section.

Metric	No Exp.	Correct Exp.
Accuracy	$86.88\% \pm 7.4\%$	**90.26% ± 6.03%**
Speed (sec)	986.25 ± 561.35	**517.42± 198.1**
Agreement	$74.38\% \pm 7.78\%$	**78.95%± 4.95%**

Table 2: Performance of novice users with and without explanations on the activity recognition task. We measured the accuracy of the participants on the task, the speed at which they completed the task, and the fraction of instances on which their answer agreed with the system's answer. Bold results indicate significantly higher score.

For the condition with no explanations, the interface still included the video player but without highlighted segments for queries. However, none of the explanatory interface elements were included for the no-explanation condition. In both conditions, the system provided the ML answer for each query.

After the participants submitted their answers, the system provided feedback by telling the participant the correct answer to the query. This was done to further help users develop a better mental model of the system over time. To do this, we froze their responses after answering in order to keep them from changing it when shown correct answer.

Overall, the results demonstrate that the system with explanations significantly improved task performance, based on both speed and accuracy, for the activity recognition task (see Table 2). Users who were given the system with explanations were significantly more accurate and completed the task significantly faster than those who were given no explanations. In addition, we measured participants' response agreement—the percentage of participant responses that matched the ML output over all queries. Participants who used the system with explanations demonstrated greater agreement with the system's prediction. These results, combined with the fact that participants in the explanation condition were able to complete the task more quickly, suggest that participants developed a higher level of trust in the system than the group with no explanations.

6 Conclusion

We proposed a method of generating explanations for activity recognition in video that combines the discriminative power of deep neural networks with tractable models. Our experimental results suggest that this approach can yield a highly accurate prediction system, and our initial user study implies that quality explanations are key to building trust in ML systems.

As a part of our future work, we plan on improving upon our current system by (1) adding support for more vocabulary in the video classification layer, (2) adding support for complex models and automatic query conversion from natural language in the explanation layer, and (3) adding support for a larger variety of complex queries, e.g., temporal queries. We expect that adding these features will increase the trust of the users in the system as the range of activities, the precision of the explanations, and the variety of queries will all increase.

References

Anonymous. Cutset Bayesian Networks. In *IJCAI 2019: Under Review*, 2019.

F. R. Bach and M. I. Jordan. Thin junction trees. In *NeurIPS*, 2002.

J. Bekker, J. Davis, A. Choi, A. Darwiche, and G. Van den Broeck. Tractable learning for complex probability queries. In *NeurIPS*, 2015.

C. K. Chow and C. N Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14:462–467, 1968.

A. Darwiche. A differential approach to inference in Bayesian networks. In *UAI*, 2000.

R. Dechter and R. Mateescu. AND/OR search spaces for graphical models. *Artificial Intelligence*, 171:73–106, 2007.

N. Di Mauro, A. Vergari, and T. M. A. Basile. Learning Bayesian random cutset forests. In F. Esposito, O. Pivert, M. Hacid, Z. W. Rás, and S. Ferilli, editors, *Foundations of Intelligent Systems*, pages 122–132. Springer International Publishing, 2015.

N. Di Mauro, A. Vergari, and F. Esposito. Learning accurate cutset networks by exploiting decomposability. In *Congress of the Italian Association for Artificial Intelligence*, 2015.

J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015.

A. Doucet, N. de Freitas, K. P. Murphy, and S. J. Russell. Rao-Blackwellised particle filtering for dynamic Bayesian networks. In *UAI*, 2000.

X. Duan, W. Huang, C. Gan, J. Wang, W. Zhu, and J. Huang. Weakly supervised dense event captioning in videos. In *NeurIPS*, 2018.

K. A. Hoff and M. Bashir. Trust in automation: Integrating empirical evidence on factors that influence trust. *Human Factors*, 57(3), 2015.

R. R. Hoffman. A taxonomy of emergent trusting in the human–machine relationship. *Cognitive Systems Engineering: The Future for a Changing World*, 2017.

M. J. Johnson, D. Duvenaud, A. B. Wiltschko, S. R. Datta, and R. P. Adams. Composing graphical models with neural networks for structured representations and fast inference. In *NeurIPS*, 2016.

D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

R. Krishna, K. Hata, F. Ren, L. Fei-Fei, and J. C. Niebles. Dense-captioning events in videos. In *ICCV*, 2017.

T. Kulesza, M. Burnett, W. Wong, and S. Stumpf. Principles of explanatory debugging to personalize interactive machine learning. In *Proceedings of the 20th International Conference on Intelligent User Interfaces*, 2015.

J. D. Lee and K. A. See. Trust in automation: Designing for appropriate reliance. *Human factors*, 46(1):50–80, 2004.

Y. Liang, J. Bekker, and G. Van den Broeck. Learning the structure of probabilistic sentential decision diagrams. In *UAI*, 2017.

D. Lowd and P. M. Domingos. Learning arithmetic circuits. In *UAI*, 2008.

B. M. Muir and N. Moray. Trust in automation. part ii. experimental studies of trust and human intervention in a process control simulation. *Ergonomics*, 39(3):429–460, 1996.

B. M. Muir. Trust in automation: Part i. theoretical issues in the study of trust and human intervention in automated systems. *Ergonomics*, 37(11):1905–1922, 1994.

K. P. Murphy. *Dynamic Bayesian networks: representation, inference and learning*. PhD thesis, University of California, Berkeley, 2002.

A. Niculescu-Mizil and R. Caruana. Predicting good probabilities with supervised learning. In *ICML*, 2005.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

H. Poon and P. Domingos. Sum-product networks: A new deep architecture. In *ICCV Workshops*, 2011.

L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

T. Rahman and V. Gogate. Learning ensembles of cutset networks. In *AAAI*, 2016.

T. Rahman, P. Kothalkar, and V. Gogate. Cutset networks: A simple, tractable, and scalable approach for improving the accuracy of chow-liu trees. In *ECML PKDD*, 2014.

M. Regneri, M. Rohrbach, D. Wetzel, S. Thater, B. Schiele, and M. Pinkal. Grounding action descriptions in videos. *TACL*, 1:25–36, 2013.

A. Rohrbach, M. Rohrbach, W. Qiu, A. Friedrich, M. Pinkal, and B. Schiele. Coherent multi-sentence video description with variable level of detail. In *German conference on pattern recognition*, 2014.

A. Rooshenas and D. Lowd. Learning sum-product networks with direct and indirect variable interactions. In *ICML*, 2014.

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3), 2015.

Y. C. Song, I. Naim, A. Al Mamun, K. Kulkarni, P. Singla, J. Luo, D. Gildea, and H. A. Kautz. Unsupervised alignment of actions in video with text descriptions. In *IJCAI*, 2016.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

C. Szegedy. Googlenet pre-trained model. http://dl.caffe.berkeleyvision.org/bvlc_googlenet.caffemodel, 2014.