Transactions on
Interactive Intelligent Systems

Deciphering a Deep Learning Black-Box via a Cutset Network: Explainable Activity Recognition in Videos

Journal:	*Transactions on Interactive Intelligent Systems*
Manuscript ID	Draft
Manuscript Type:	SI: Interactive Visual Analytics for Making Explainable and Accountable Decisions
Date Submitted by the Author:	n/a
Complete List of Authors:	Roy, Chiradeep; University of Texas at Dallas Nourani, Mahsan; University of Florida Shanbhag, Mahesh; University of Texas at Dallas Rahman, Tahrima; University of Texas at Dallas Ragan, Eric ; University of Florida, Ruozzi, Nicholas; University of Texas at Dallas Gogate, Vibhav; University of Texas at Dallas

Deciphering a Deep Learning Black-Box via a Cutset Network: Explainable Activity Recognition in Videos

CHIRADEEP ROY, The University of Texas at Dallas
MAHSAN NOURANI, University of Florida
MAHESH SHANBHAG, The University of Texas at Dallas
TAHRIMA RAHMAN, The University of Texas at Dallas
ERIC D. RAGAN, University of Florida
NICHOLAS RUOZZI, The University of Texas at Dallas
VIBHAV GOGATE, The University of Texas at Dallas

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 into a tractable, interpretable probabilistic graphical model called a dynamic conditional cutset network 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 human-subjects study.

CCS Concepts: • **Computing methodologies** → **Activity recognition and understanding**; **Maximum a posteriori modeling**; • **Human-centered computing** → *User studies.*

Additional Key Words and Phrases: activity recognition, temporal models, cutset networks

ACM Reference Format:
Chiradeep Roy, Mahsan Nourani, Mahesh Shanbhag, Tahrima Rahman, Eric D. Ragan, Nicholas Ruozzi, and Vibhav Gogate. 2019. Deciphering a Deep Learning Black-Box via a Cutset Network: Explainable Activity Recognition in Videos. *J. ACM* 37, 4, Article 111 (August 2019), 24 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Video activity recognition—inferring high level activities from a sequence of frames—has received increasing attention in recent years. This task is notoriously difficult especially when: (1) the number of activities is large; (2) each frame is associated with multiple activities; and (3) activities in different frames depend on each other. Despite the high degree of difficulty, recent advances in deep learning

Authors' addresses: Chiradeep Roy, The University of Texas at Dallas, Chiradeep.Roy@utdallas.edu; Mahsan Nourani, University of Florida, mahsannourani@ufl.edu; Mahesh Shanbhag, The University of Texas at Dallas, rayashanbhag@gmail.com; Tahrima Rahman, The University of Texas at Dallas, Tahrima.Rahman@utdallas.edu; Eric D. Ragan, University of Florida, eragan@ufl.edu; Nicholas Ruozzi, The University of Texas at Dallas, Nicholas.Ruozzi@utdallas.edu; Vibhav Gogate, The University of Texas at Dallas, Vibhav.Gogate@utdallas.edu.

0004-5411/2019/8-ART111 $15.00
https://doi.org/10.1145/1122445.1122456

architectures and algorithms [22, 39, 58, 59] have made it possible to accurately solve this task. In particular, we can identify activities in a given video using the following approach: (1) predict activities happening in each frame using deep learning based image classification techniques; and (2) aggregate the predictions by leveraging prior knowledge to resolve discrepancies between the predictions (e.g., using a constraint that activities do not change rapidly). Unfortunately, a problem with the approach just described is that deep learning models are black-boxes; they do not give their users a good understanding of how they work and why they arrived at a particular decision.

To address this issue, we focus on *Explainable Activity Recognition (XAR)*, which we loosely define as the task of inferring high-level activities from videos (or in general from low-level sensors) along with an explanation of why the activities were chosen in lieu of other activities [17]. An XAR system can benefit and enable a wide variety of real-world applications. For instance, a video surveillance system would greatly benefit from (human understandable) explanations that describe why the system flagged (predicted) activities happening in specific video segments as suspicious or benign. These explanations can either be short or detailed. A short explanation, for example, would tag the most important sub-segments in each video segment where the specific (e.g., when a package is stolen from a porch) or a relevant activity (e.g., when a package is touched but not stolen after seeing the surveillance equipment) happened. A detailed explanation, for example, would describe alternate or competing hypotheses along with a confidence on each hypothesis and its various components (e.g., the system believes that with a probability greater than 70%, the package on the porch was touched at a later time by the delivery person because he/she forgot to scan the package when it was delivered). Finally, an indirect advantage of explanations is that they help the user better understand how the system works, which in turn helps her/him build a mental model of the system's functioning and gain greater trust in its decisions.

The main purpose of this paper is to describe a general approach for XAR and then apply and evaluate it—using both machine learning metrics and human subjects studies—for activity recognition in cooking videos. Specifically, we build an XAR system that can perform the following three tasks: (1) parse a video into a set of pre-defined activity labels, namely divide each video into segments and associate activity labels with each segment; (2) use this information to answer Yes/No queries posed by the user; and (3) provide three different kinds of explanations to add context to the system's answers. The third task, in particular, can only be performed by an explainable machine learning system and is of particular interest to us.

Our system consists of two parts: (1) an explainable machine learning model, which forms the nuts and bolts of our system; and (2) a visual interface which provides answers to user's queries as well as explanations. Our model, in turn, has two layers, a video classification layer and an explanation layer (see Fig. 1). The video classification (top) layer is a deep neural network that takes video frames as input and predicts an activity label for each frame. The predicted labels are then fed into the explanation (bottom) layer. The latter aggregates the predictions made by the neural network and improves the accuracy using a probabilistic model that represents and reasons about relationships between different activities as well as temporal constraints. The explanation layer provides answers to the queries posed by the user as well as explanations; both tasks are solved by performing inference over the probabilistic model.

The explanation layer consists of a tractable, interpretable probabilistic graphical model [24], specifically a cutset network [47]. 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 [10, 31, 35, 35, 45–47, 52]. In other words, inference over cutset networks is always fast and accurate, and as a result they often yield significantly better quality predictions and explanations than Bayesian and Markov networks.

A possible interpretation of our explainable machine learning model is that the deep learning layer provides noisy sensory inputs to the cutset network layer, which in turn removes the noise and provides explanations. The neural network, by itself, is unable to provide explanations because it does not model, and therefore is unable to reason about, the relationships between the predicted labels. On the other hand, a cutset network explicitly models relationships between various activities and can provide fast, high quality explanations by performing (abductive) probabilistic inference over the 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.

The second part of our system consists of a browser-based user interface that can be used to pose Yes/No questions to the system (see 5). The user first chooses a video and can then choose from a list of questions of the form "Did activity X take place?" The system then uses the explanation layer to search for frames where there is an activity match and displays explanations in the form of video segments, ranked triples, and component-wise contributions to the explanations. If no perfect matches are found, then the system answers *No* to the query and uses partial component-wise matches to explain its decision.

We evaluated the machine learning part of our system using standard information retrieval metrics such as K-group measures, the Jaccard Index and the Hamming Loss. Specifically, we compared our two-layer architecture, which contains a video classification layer and an explanation layer, with a one-layer architecture that only contains the video classification layer. We observed that the two-layer architecture is more accurate than the one-layer architecture. We evaluated the interface using human-subjects studies where each user was shown a set of videos and presented with questions that she/he had to answer using the explanations provided by the system. Our results clearly demonstrate that the users who used the explanations found it easier to complete the task and were able to develop a higher level of trust in the system.

The rest of the paper is organized as follows. In the next section, we describe related work. In section 3, we describe the desiderata of an XAR system for cooking videos and show how to build the system using machine learning representations and algorithms in section 4. We empirically evaluate the machine learning models in section 5 and describe the results of a human-subjects study for measuring explanation effectiveness in section 6. Finally, we summarize our contributions and present avenues for future work in section 7.

2 RELATED WORK

Traditionally, researchers have used spatiotemporal models for activity recognition that treat the video as a 3D spatiotemporal object having co-ordinates (x, y, t) where x and y represent the spatial co-ordinates of each image at time slice t of the video. For example, Laptev [28] generalized Harris' interest point detector to the 3D domain in order to locate spatiotemporal chunks that exhibit high variations of local pixel intensities. Other spatiotemporal approaches have tried comparing sub-volumes of the 3D video cuboid to predefined action templates, e.g., [3, 54], while others have tried to track the trajectories of points in motion, e.g., [6, 49], and compare these trajectories to those of known actions. While these approaches have been fairly successful in the past, most of them have been used to detect simple action primitives such as walking, jumping, etc. and have difficulty generalizing to more complex activities. They also typically only provide information about the action taking place; our model aims to provide additional information such as the object that is being affected by the action and the location where the action is taking place.

More recently, Convolutional Neural Networks (CNNs) [29], which have already found extraordinary success in large scale image classification and object detection, e.g., [26, 58], have been applied to this task. For example, Yang et al. [63] successfully applied CNNs as low-level object detectors with a probabilistic parser built on top that acts as a semantic wrapper around these features. Other works have demonstrated that CNNs can be used to automatically extract relevant features that are often better than hand-crafted features used in the spatiotemporal approaches, e.g., [14, 61]. We wanted to combine the strength of CNNs, detecting intricate spatial patterns and extracting rich visual features, together with the strengths of temporal probabilistic models such as Hidden Markov Models [4, 56, 62] and Dynamic Bayesian Networks [5, 16, 20], which have been used extensively in the activity recognition literature, in order to build a robust XAR system that is both accurate and explainable.

This effort is also closely related to the work of Rohrbach et al. [51] and Donahue et al. [11] 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 [25], i.e., generating summaries of texts from particular segments. Song et al. [55] attempted to create captioning methods that require minimum supervision on the TaCOS dataset. Duan et al. [13] 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., [18, 30, 36, 37]. These studies examine factors that might affect the trust of the user in the system, such as the past performance of the system and how understandable the system is to the user [30]. Hoffman [19] 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 work that combines graphical models and neural networks, e.g., [21]. 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 each frame of the video 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.

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 all activities are simple and not part of other, more complex activities. 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 the 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 (simple) 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

In addition to answering queries posed by the user, we also want the system to provide explanations to justify its answers. The current framework generates three different 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. For example, for the question "does the person in the video wash his hands?", there might be two segments in the video from say, 01:00 to 01:10 and from 04:15 to 04:25 where the person washes his hands. We want our system to detect these segments and use them as explanations to justify its answer to the question ("yes" in this case). If the person does not wash his hands in the video, we would expect the system to answer "no" and explain its answer by highlighting a section of the video (say from 00:20 to 00:35) where the person performs a similar activity such as washing a knife or washing a peeler. The system is expected to therefore justify "no" answers by saying that similar activities were detected but not the specific activity (or activities) that the user was querying for.

(2) **Ranked (action, object, location) Triples:** We want the system to display the top-s predicted activity triples in the video that are relevant to the query. For example, for the question "does the person cut a carrot?", the system might answer "yes" and display a list of three possible explanations: *(cut, carrot, cutting-board)*, *(cut, carrot, plate)* and *(cut, orange, plate)*. We want these explanations to be *ordered* in descending order of likelihood (or confidence). In this case, we know that the system believes that the activity taking place was *(cut, carrot, cutting-board)* with the highest degree of confidence, followed by *(cut, carrot, plate)* and *(cut, orange, plate)*. These explanations not only provide more context to the answers but also help the user decode patterns in the behavior of the system. For instance, the user might notice that the system frequently generates "orange" as an alternative explanation for "carrot" presumably because they have the same color. The model that we use for our system is able to generate these kinds of explanations.

(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. Using the same example as above, we want the system to give us a component-wise score for the components *cut* (action), *carrot* (object), *orange* (object), *cutting-board* (location) and *plate* (location). The system might have a 100% score for *cut* because it is very confident that the cutting action is

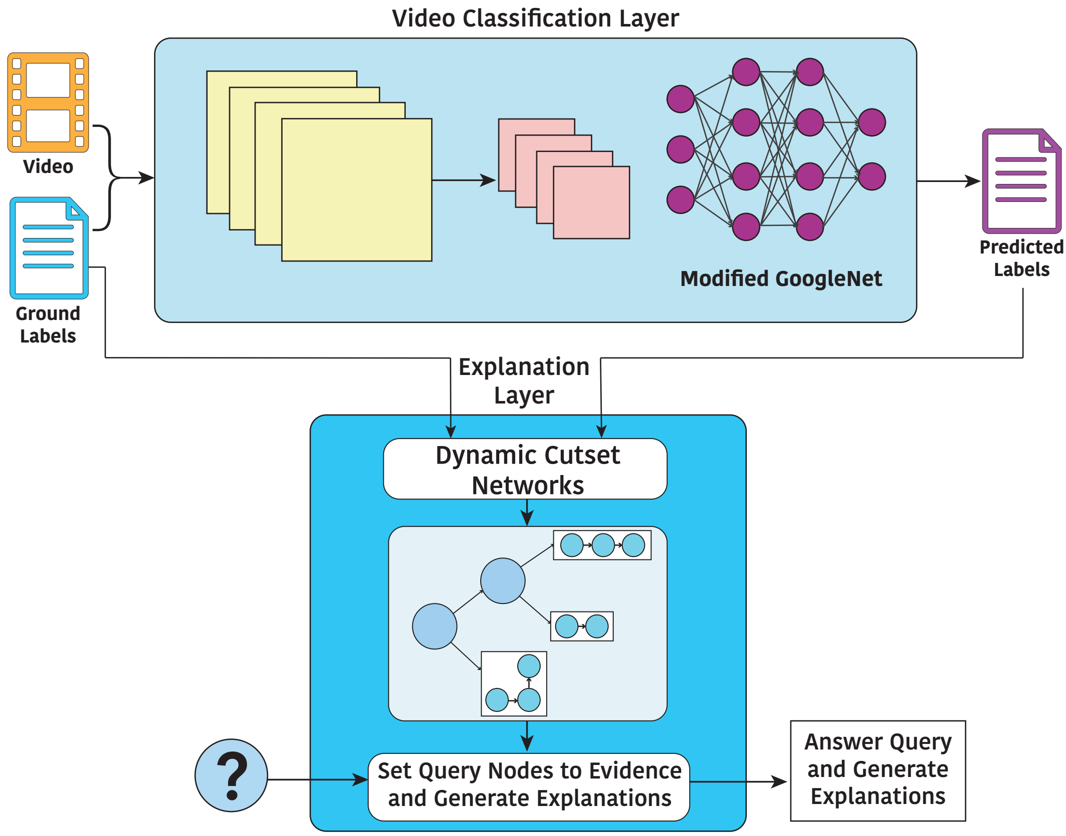


Fig. 1. High-level Architecture and Data Processing Pipeline. Our system has two layers: a video classification layer based on deep learning whose output is fed to an explanation layer which is based on cutset networks [47], a tractable interpretable probabilistic model. During the learning phase, the 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).

taking place but only a 60% score for *carrot* because it is not sure if the object being cut is a carrot or an orange. Once again, the cutset network that we use in the explanation layer makes it very easy to generate these types of explanations.

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'. The system can be roughly organized 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

The input to this layer is a single video frame that is mapped to a set of output activity labels via a deep neural network. This process is repeated for each frame of each video. This layer was implemented using GoogLeNet [57], a 22-layer Convolutional Neural Network[1] (CNN) that is pre-trained on the ImageNet dataset [53]. It uses a key component called an Inception Module that creatively uses convolutions of size 1×1 to increase the representation power of the network without increasing the number of parameters. Fig. 2 shows how the convolution and pooling layers are arranged in this architecture.

4.1.1 1×1 Convolutions. While filters (or convolutions) in CNNs can be used to greatly reduce the number of parameters for fully-connected networks (FCNs), in many practical scenarios this reduction is still not enough to avoid overfitting. GoogleNet attempts to circumvent this problem by using 1×1 convolutions. Consider a scenario where the input tensor has dimensions 28×28×192 which we want to reduce to a tensor of size 28×28×32 by using a filter of size 5×5×32. This would imply that every unit of the output tensor 28×28×32 would be a dot product of dimensions 5×5×32 from the input volume. Unfortunately, this results in a little over 120 million parameters. If instead we were to stack a 1×1×16 filter followed by a 5×5×16 filter then we would have a total of only 12 million parameters which is a reduction of over 90%. Therefore, these 1×1 convolution layers serve to increase not only the depth but also the width of the network without noticeable loss in performance. This idea was first proposed by the authors of Network-in-Network [32] which has been applied in GoogLeNet [58] to the CNN setting.

4.1.2 Inception Module. In most state-of-the-art CNN architectures, a choice needs to be made for each layer wherein the modeler needs to choose between a stack of 3×3 filters, 5×5 filters or max pooling. The Inception Module (see Fig. 3 for a high-level overview) circumvents this problem by combining all these components together followed by stacking their results together (Filter Concatenation) and feeding it to the next layer without an exponential increase in the number of parameters. The dimensionality of the images is reduced by performing 1×1 convolutions before applying 3×3 and 5×5 filters and after applying max pooling. The 1×1 convolutions are helpful for capturing features from layers closer to the input layer where pixel correlations will form local clusters. The 3×3 and 5×5 convolutions are used to capture the semantic relationships between clusters that are spread out.

4.1.3 Training. We modified the GoogleNet architecture slightly to accommodate our problem. Specifically, we replaced the output layer of GoogLeNet by a softmax layer over a set of 28 activity labels (12 actions, 7 objects and 8 locations plus the special label called "Nothing"). As mentioned earlier, the base architecture was already pre-trained on the ImageNet dataset which was a part of the ILSVRC challenge [53]. This dataset has over 200 object classes and over 450K training instances. Our modified architecture was then trained on our dataset for a fixed number of iterations.

[1]Convolutional Neural Networks are a type of deep neural networks that are often used for solving image classification tasks. At the heart of this network is the convolution operation which takes as input an image represented using a tensor having dimensions $width \times height \times 3$ (3 channels for RGB) and a filter represented using a tensor having dimensions $f_1 \times f_2 \times 3$ and outputs a tensor having dimensions $width - f_1 + 1 \times height - f_2 + 1 \times 1$. The output tensor is obtained by sliding the filter over the image (shifting it by one horizontally and vertically) and performing element wise dot product at each step. At a high level, each layer in a convolutional network reduces the images into a new feature representation which is easier to process and is likely to yield high accuracy.

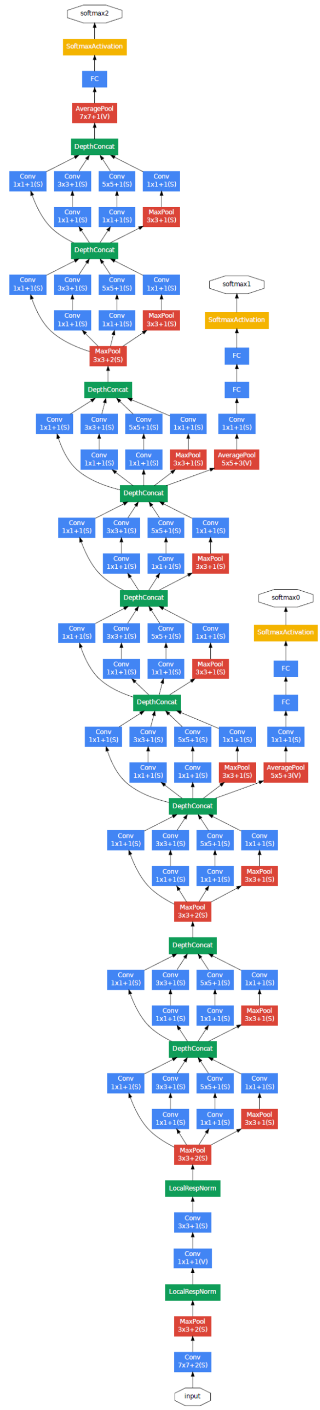


Fig. 2. Schematic layout of the GoogleNet architecture. The blue boxes represent convolutional layers (filters), the red boxes pooling layers, the yellow boxes softmax layers and the green boxes depth concatenation layers. Depth concatenation simply concatenates the results of its inputs along the depth dimension.

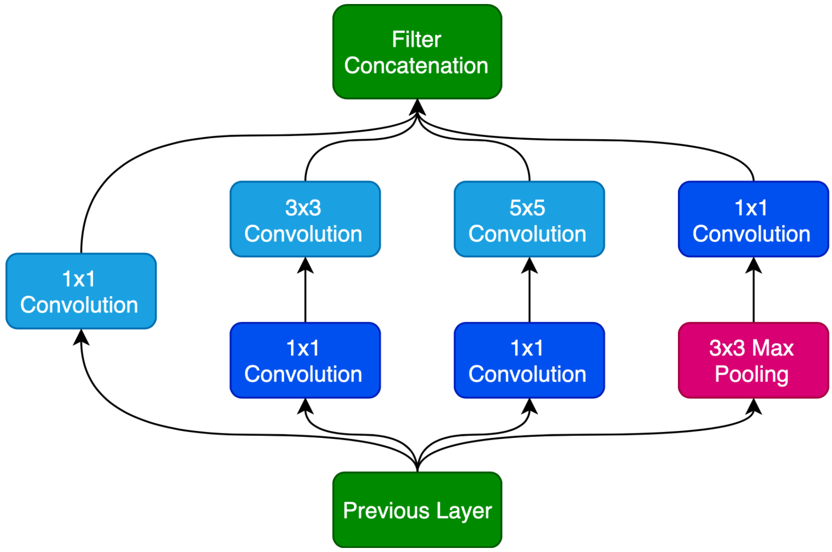


Fig. 3. Architectural layout of the inception module. Instead of having to choose the best filters, the outputs of a 1×1, 3×3, 5×5 convolution layer and a max-pooling layer are concatenated together at the concatenation layer. Three additional 1×1 units are used to further reduce dimensionality size—two for the 3×3 and 5×5 convolution layers and one for the max pooling layer.

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 do not change rapidly between frames) in the video; and (c) provide explanations via abductive poly-time probabilistic inference.

4.2.1 Cutset Networks. Probabilistic Graphical Models (PGMs) such as Bayesian and Markov networks [24] are widely used in practice to represent and reason about uncertainty. At a high level, they are a compact representation of the joint probability distribution over a large number of random variables. Once learned from data, they can be used to answer any query posed over the joint distribution via probabilistic inference. The two main types of inference (queries) tasks are posterior marginal (MAR) and maximum-a-posteriori (MAP) inference. In MAR inference, we are interested in computing the marginal probability distribution over each query variable given evidence where evidence (or observation) is an assignment of values to a subset of random variables. In MAP inference, we are interested in computing the most likely assignment to all query variables given evidence. Both tasks are notoriously difficult to solve in many practical networks, and theoretically they are NP-hard in general. As a result, in practice, one has to often use approximate inference algorithms to solve these problems (approximately). Unfortunately, these algorithms are unreliable and often yield inaccurate query answers.

Tractable probabilistic models (TPMs) [1, 8, 33] are special types of probabilistic models which admit poly-time MAR and MAP inference and thus circumvent the problem of unreliability of approximate inference in Bayesian and Markov networks. Although TPMs are less expressive than intractable (latent) probabilistic models such as Bayesian and Markov networks, their prediction accuracy (at test time) is often much higher than intractable models. This is because tractable

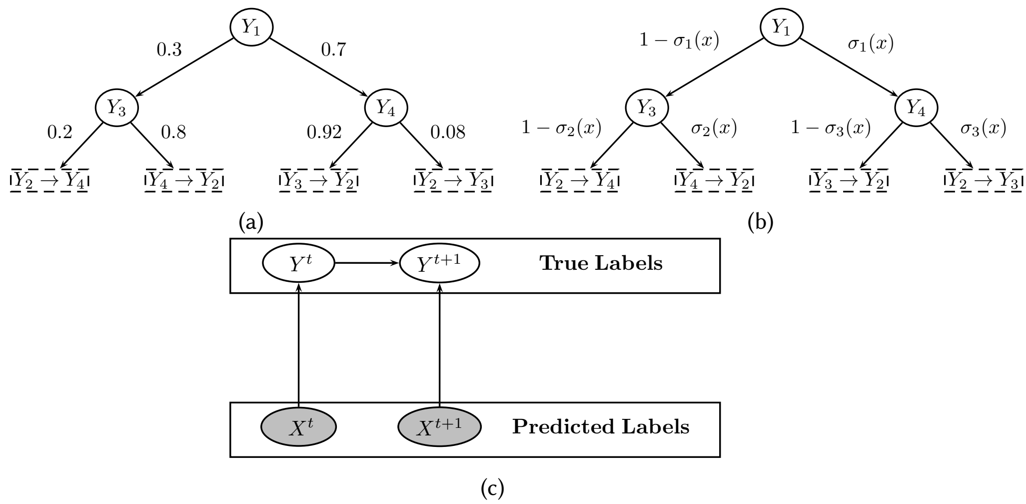


Fig. 4. (a) A cutset network over 4 variables $\{Y_1, \ldots, Y_4\}$. OR nodes are denoted by circles. Y_1 is the root node of the OR tree. Left and right arcs emanating from an OR node labeled by Y_i indicate conditioning over false (assignment of 0) and true (assignment of 1) values of Y_i respectively. Arcs emanating from OR nodes are labeled with conditional probabilities. For example, the arc labeled with 0.08 denotes the conditional probability $P(Y_4 = 1|Y_1 = 1)$. The leaf nodes of the OR tree are tree Bayesian networks. (b) A conditional cutset network (CCN) representing $P(Y_1, \ldots, Y_4|X)$. Arcs emanating from OR nodes are labeled with (calibrated) classifier functions. For example, the arc from the OR node Y_1 to the OR node Y_3 is labeled with a logistic regression classifier $1 - \sigma_1(x)$. Given an assignment $X = x$ to all variables in X, the CCN yields a cutset network having the same structure as the one given in (a) except that the parameters will be computed using σ_1, σ_2 and σ_3. (c) 2-slice dynamic conditional cutset network. The CCN at time slice t represents $P(Y^t|X^t)$ while the CCN at time slice $t+1$ represents $P(Y^{t+1}|X^{t+1}, Y^t)$.

models use exact inference at prediction time while one has to use inaccurate approximate inference algorithms in Bayesian and Markov networks. Examples of popular TPMs include cutset networks [44, 45, 47], arithmetic circuits [8, 33], sum-product networks [42] and probabilistic sentential decision diagrams [2].

Cutset networks [47] are a class of TPMs that use recursive cutset conditioning [34, 41] to build a rooted OR tree where each non-leaf node corresponds to a conditioned variable and each leaf node corresponds to a tree-structured Bayesian Network defined over all variables not appearing on the path from the root to the leaf. Formally, given a set of variables $X = \{X_1, \ldots, X_n\}$, a cutset network C is a pair (O, T) where O represents an OR tree and T represents a set of tree-structured Bayesian Networks, one for each leaf node in O (see Fig. 4(a) for an example). Assuming that all the variables in X are binary, each non-leaf node in O will have two branches. We will assume that the left and right branches of an OR node labeled by X_i in O correspond to the values $\overline{x}_i$ and x_i respectively where $\overline{x}_i$ (similarly x_i) denotes an assignment of value 0 to X_i (similarly 1). Each directed edge between an OR node labeled by X_i and its child node in O is labeled with the conditional probability of the variable X_i taking the corresponding value given the assignment on the path from the root to X_i. For example, in Fig. 4a, 0.92 equals the conditional probability $P(\overline{y}_4|y_1)$. Every non-leaf node partitions the probability space into data points that agree with $\overline{x}_i$ in the left sub-tree and those that agree with x_i in the right sub-tree. The probability of a full assignment x w.r.t. the cutset network C is given by

$$P_C(x) = T_{l(x)}\left(x_{V\left(T_{l(x)}\right)}\right) \cdot \prod_{p \in O(x)} p \tag{1}$$

Algorithm 1 LearnCNet

Input: Dataset $D = \{x^{(1)}, \dots, x^{(m)}\}$ having m training examples, Variables $X = \{X_1, \dots, X_n\}$
Output: Cutset network C

1: **if** termination condition **then**
2: **return** ChowLiuTree(D)
3: Use a splitting heuristic to select a variable $X_i \in X$
4: Create a new OR node o and label it by X_i
5: $D_{\overline{x}_i} \leftarrow \{x \in D | X_i = 0\}$
6: $D_{x_i} \leftarrow \{x \in D | X_i = 1\}$
7: Let l and r denote the left and right child nodes of o
8: $l \leftarrow$ LearnCNet($D_{\overline{x}_i}, X \setminus \{X_i\}$)
9: $r \leftarrow$ LearnCNet($D_{x_i}, X \setminus \{X_i\}$)
10: Let $p_{o,l}$ and $p_{o,r}$ denote the conditional probabilities on the edge between o and l and between o and r respectively.
11: $p_{o,l} \leftarrow \frac{|D_{\overline{x}_i}|}{|D|}$
12: $p_{o,r} \leftarrow \frac{|D_{x_i}|}{|D|}$
13: **return** o

where $l(x)$ denotes the leaf node in C corresponding to the assignment x, $T_{l(x)}$ denotes the tree Bayesian network at $l(x)$, $V(T_{l(x)})$ denotes the set of variables over which $T_{l(x)}$ is defined, $x_{V(T_{l(x)})}$ denotes the projection of the assignment x on the variables $V(T_{l(x)})$ (where $V(T_{l(x)}) \subseteq X$) and $O(x)$ is the set of conditional probabilities on the path from root to the leaf node $l(x)$ in the OR tree O.

The time complexity of posterior marginal estimation (MAR) and full maximum a-posteriori estimation (MAP) is linear in the size of the cutset network as it requires just two passes over the cutset network [47]. The fact that most prediction tasks can be reduced to these two types of inference queries makes these models an attractive choice for applications that rely heavily on exact inference at test time.

The structure and parameters of cutset networks can be learned from data using the top-down, recursive induction approach described in Algorithm 1. The algorithm has two main steps: base case and conditioning step. In the base case, the algorithm returns a tree Bayesian network if a pre-defined termination condition (a popular condition is described below) is satisfied. The tree Bayesian network is learned from data using the Chow-Liu algorithm [7]. This algorithm first constructs an undirected weighted complete graph in which each node corresponds to a variable X_i in X and each edge (X_i, X_j) is weighed using the mutual information score (MIScore) between X_i and X_j:

$$\text{MIScore}(X_i, X_j) = \sum_{i=0}^{1} \sum_{j=0}^{1} P_D(X_i = i, X_j = j) \log \frac{P_D(X_i = i, X_j = j)}{P_D(X_i = i) P_D(X_j = j)}$$

where $P_D(X_i = i, X_j = j)$ is estimated from the dataset D; the estimate equals the number of times the partial assignment $(X_i = i, X_j = j)$ appears in the data divided by the number of examples in D, and $P_D(X_i = i) = \sum_{j=0}^{1} P_D(X_i = i, X_j = j)$ (similarly, $P_D(X_j = j) = \sum_{i=0}^{1} P_D(X_i = i, X_j = j)$). Then, the Chow-Liu algorithm finds a maximum spanning tree from the weighted complete graph and converts the tree to a directed tree K using depth-first search. The latter yields a tree Bayesian network which represents the following distribution:

$$T(x) = \prod_{i=1}^{n} P_D(x_{\{X_i\}} | x_{pa_K(X_i)})$$

where $pa_K(X_i)$ is the set of parents of X_i in K. Note that $pa_K(X_i) \leq 1$ for all i.

The following termination condition is often used in practice [44, 45, 47]. Stop growing the OR tree if any of the following conditions are satisfied: (1) The number of examples is smaller than k; (2) The depth of the OR tree is larger than d (d is bounded by n). Hyperparameters d and k are tuned using the validation set; namely we search over possible choices of d and k and choose the combination that gives the highest log-likelihood score on the validation set.

In the conditioning step, the algorithm heuristically selects a variable X_i to condition on. The following heuristic is often used in practice [9, 44]. We select a variable having the following sum mutual information score with ties broken randomly:

$$\text{SumMIScore}(X_i) = \sum_{j:j \neq i} \text{MIScore}(X_i, X_j)$$

Once the variable (X_i) is selected, the algorithm induces an OR node o labeled by X_i (line 4). Then, the algorithm partitions the dataset D into two datasets, $D_{\overline{x}_i}$ and D_{x_i} where the former contains only those examples in D which X_i is assigned the value 0 while the latter contains only those examples in D which X_i is assigned the value 1. It then creates two child nodes l and r and recursively constructs a CN on l and r using $D_{\overline{x}_i}$ and D_{x_i} respectively. Finally, the algorithm estimates the conditional probability on the edges between l and o and between r and o (lines 11 and 12) and returns the OR node o.

4.2.2 Conditional Cutset Networks. Conditional cutset networks (CCNs) are a new framework that was recently proposed by Rahman et al. [46]. As the name suggests, they generalize the cutset networks framework to compactly represent conditional distributions of the form $P(Y|X)$ where X and Y are sets of variables. In CCNs, the OR tree and each tree Bayesian network is defined over variables in Y. The conditional probabilities in the OR tree and tree Bayesian networks are given by a calibrated probabilistic classifier [40]. These classifiers take as input an assignment x to a set of variables X and output a probability distribution over the class label $Y_i \in Y$. Tractability over each individual distribution is still maintained since the number of parameters for most calibrated classifiers scales polynomially with the number of input variables X. For example, when we using logistic regression, we have $P(Y_i = 1|X = x) = \sigma(w_0 + \sum_{X_i \in X} w_i x_{\{X_i\}})$ where w_i's are the weights (parameters) and σ denotes the sigmoid function.

Given an assignment x to all variables in X, a CCN yields a cutset network because each calibrated classifier yields a marginal probability distribution over the class variable. Thus, given x, CCNs yield a tractable probabilistic model over X. Fig. 4(b) shows an example of a conditional cutset network.

Structure and parameters of a CCN can be learned (see Algorithm 2) using the same top-down induction approach used for cutset networks. The differences between the two algorithms are:

(1) In LearnCCN, we learn the parameters on the edges of the OR tree and the conditional distributions at each node in each tree Bayesian network using a calibrated classifier $\sigma(X)$ (e.g., logistic regression, neural networks, random forests, etc.). The best classifier is chosen using cross-validation.
(2) In LearnCCN, we learn the tree Bayesian networks at each leaf node of the OR tree using *conditional mutual information scores*. Similarly, the splitting heuristic in the LearnCCN algorithm uses sum conditional mutual information scores as compared with sum mutual information scores in the LearnCNet algorithm. See Rahman et al. [46] for details.

4.2.3 Using CCNs to Predict Activity Labels. To use CCNs in our video activity recognition framework, we feed the output of GoogLeNet to the CCN. More formally, let X denote the set of output nodes of GoogLeNet and Y denote the set of true labels at a frame. We use the CCN to model $P(Y|X)$ and learn its structure and parameters using Algorithm 2. Given a set of videos V, the training

Algorithm 2 LearnCCN

Input: Dataset $D = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$, Sets of Variables Y and X
Output: Conditional cutset network C

```
1: if termination condition then
2:     return ConditionalChowLiuTree(D, Y, X)
3: Use a splitting heuristic to select a variable Y_i ∈ Y
4: Create new OR node o and label it by Y_i
5: Learn a calibrated classifier σ(X) with class label Y_i and X as input using D
6: D_{ȳ_i} ← {(x, y) ∈ D | Y_i = 0}
7: D_{y_i} ← {(x, y) ∈ D | Y_i = 1}
8: Let l and r denote the left and right child nodes of o
9: l ← LearnCCN(D_{ȳ_i}, Y \ {Y_i})
10: r ← LearnCCN(D_{y_i}, Y \ {Y_i})
11: Label the edge between l and o by 1 − σ(X)
12: Label the edge between r and o by σ(X)
13: return o
```

dataset D is constructed as follows. We have one training example in D for each frame in each video of V. Each example is composed of true labels (Y) and labels predicted by GoogleNet (X) 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 MAP inference over the cutset network to yield the final set of labels. In other words, the CCN treats the output of GoogLeNet as a noisy sensor (see Fig. 4(c)) and computes a conditional joint probability distribution over the true labels given the predicted (noisy) labels. A second benefit of CCNs, apart from improved accuracy, is that it can be used to generate high quality explanations.

4.2.4 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 Y^i and X^i be the set of true labels and predicted labels (evidence) respectively at frame i. Then, the DCCN represents the following probability distribution:

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

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

The representation given in Eq. (2) is not compact as the number of frames in a video (n) increases. To circumvent this issue, we adopt two standard assumptions widely used in temporal or dynamic probabilistic models—the 1-Markov and stationarity assumptions [43]. 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

Algorithm 3 GetPosteriorParticlesDCCN

Input: DCCN C, GoogleNet labels $X = \{x^1, .., x^n\}$ for n video frames, number of particles K
Output: Set of K particles from the approximate posterior distribution $\hat{P_C}(Y^n|x^1, .., x^n)$

```
1: particles ← Generate K samples from prior P_C(Y^1|x^1)
2: for i ∈ {2..n} do
3:     old_particles ← particles
4:     for k ∈ {1, ..., K} do
5:         particles[k] ← Sample from transition P_C(Y^i|x^{i-1}, old_particles[k])
6: return particles
```

can represent $P(y^{1:n}|x^{1:n})$ using

$$P(y^{1:n}|x^{1:n}) = P(y^1|x^1) \prod_{i=2}^{n} P(y^i|x^i, y^{i-1}), \tag{3}$$

where $P(y^1|x^1)$ and $P(y^i|x^i, y^{i-1})$ are conditional cutset networks and $P(y^i|x^i, y^{i-1})$ has the same parameters and structure for i (see Figure 4c).

We learn DCCNs using the following approach. The prior model $P(y^1|x^1)$ is the same as the CCN described in the previous section. To learn the structure and parameters of $P(y^i|x^i, y^{i-1})$, we construct the dataset as follows. Each frame in each video is a training example and is composed of true labels at frame i (Y^i), true labels at frame $i-1$ (Y^{i-1}) and labels predicted by GoogLeNet at frame i (X^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 [12]. Here, we generate K assignments $(y^{1,(1)}, \ldots, y^{1,(K)})$ uniformly at random from $P(y^1|x^1)$, then for each assignment $y^{1,(i)}$ we sample one assignment from $P(y^2|x^2, y^{1,(i)})$, and so on. At the end of the sampling process, we will have K particles from $P(y^{1:n}|x^{1:n})$.

Algorithm 3 uses this process to generate K particles that we will later use to generate the explanations. The main virtue of DCCNs is that, unlike widely used temporal models such as dynamic Bayesian networks [38], the particles in DCCNs are generated from the posterior distribution $P(y^i|x^{1:i}, y^{1:i-1})$ at each frame i. 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 [12].

4.2.5 Question Answering and Explanation Generation. As mentioned earlier, the main virtue of CCNs and DCCNs is that unlike GoogleNet, they can be used to generate the three different types of explanations described in Section 3.2. In this section, we describe how to generate the explanations.

To recap, in our proposed system, the user poses a selection type query to the system such as "Does the person in the video cut anything?" *(cut,∗,∗)* or "Does the person do anything with an orange in the sink?" *(∗,orange,sink)*. The fields with ∗'s can be matched with anything. The system then tries to search the video for any activity tuples that match the conditions of the selection query. If a complete match is found, then the system answers *Yes*. If, however, no complete match is found then the system answers *No*. The system then uses CCNs and DCCNs to generate the following types of explanations:

(1) **Video Explanations**: For each unlabeled video, we use Algorithm 3 to compute the joint probability distribution over all possible ground labels at time slice t. Then, we generate the most likely set of labels y^t at t by choosing the particle having the highest posterior probability, which is equivalent to performing MAP inference after computing the posterior distribution. Once we have the most likely set of labels for all the time slices, we can cluster

frames that have the same set of labels into a single segment. In the worst case, this process might exhibit high variance and result in segments spanning only a few frames; however, this issue can be somewhat circumvented by merging windows containing multiple frames instead of doing it on a frame-by-frame basis. Once this is done, we record the most probable activity triple associated with each segment. When the user submits a selection query, all segments whose activity triples completely match the parameters of the query are returned as video explanations. This helps the user to quickly navigate to the relevant segments of the video where the system believes that the activity took place.

More interesting, however, are the explanations generated when the system answers *No*. In such a scenario, the system returns all video segments that have partial matches. For example, for the question "Does the person take out a knife from the drawer" *(take out, knife, drawer)*, in the absence of a complete match, the system will first try to search for partial matches where at least two of the three components match. For example, if there are segments where the person takes out a peeler from the drawer *(**take out**, peeler, **drawer**)* or takes out a knife from the cupboard *(**take out**, **knife**, cupboard)*, these segments will be returned as explanations for the *No* answer. The rationale is that while the system found activities similar to the activity being queried, it did not find the exact activity being searched for and therefore answered *No*. If no such partial matches exist, the system then tries to match at least one component such as, say, *(wash, **knife**, sink)* or *(**take out**, carrot, fridge)*. If there are no single component matches either, then the system displays that no segments are found.

(2) **Ranked (action, object, location) Triples:** The particles/samples output by Algorithm 3 can also be used to generate ranked explanations for any given frame. In particular, if we want to compute the top-s most likely activities at time slice t, then we can select s particles having the highest likelihood at time slice t and display them to the user in descending order of likelihood scores; this is equivalent to performing s-MAP inference on the posterior distribution. Since video segments are returned as explanations to the system's answers and each video segment is associated with a single activity triple, we take the average of all the particles over a segment and return the top-s particles having the highest average likelihood as ranked triples.

(3) **Most Probable Entities:** Once again, these kinds of explanations can also be generated by Algorithm 3 using an approach similar to the one used for ranked triples. The only difference is that instead of generating s-MAP tuples, we wish to compute the marginal distribution $P(y^t|x^{1:t})$ that will tell us how confident the system is about a particular label at a given time slice. Since the last step of particle filtering involves generating K instantiated cutset networks, we can simply perform exact inference on these networks (which can be done tractably and in fact, linearly) to compute the posterior marginals and then average out over all K networks.

4.3 User Interface

Our 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 users in order to avoid overwhelming them 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 5 shows the interface.

The interface allows the users to select a video and a relevant query based on that video. These would serve as inputs for the model. The interface would then provide two types of output: (1) the model's answer to that query and (2) the explanations for why the answer is provided. For this purpose, the interface includes a video player with the selected video that would allow the users to go over the video if they wanted to review and analyze the system's answer to the query or try to

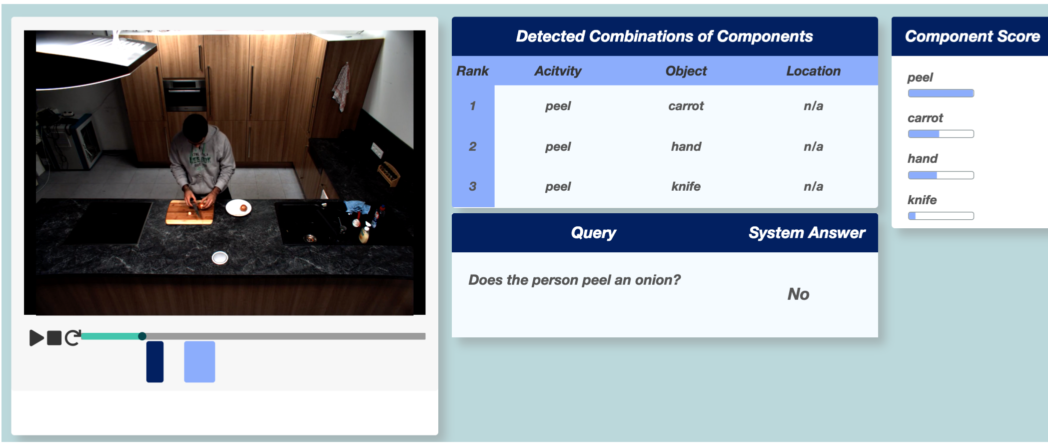


Fig. 5. The interactive visual interface allows users to load videos and ask queries. The interface shows the ML system's answer along with explanatory elements for the output. The most relevant portions of the video play time are shown by colored bars beneath the video, and the right side shows detected video components and combinations of components relevant to the video and query.

come up with their own answer for that query. The queries used in this system are in the form of yes/no questions, and hence, the answers are either *yes* or *no*.

The system also provides information to justify its answer to the selected query. For this purpose, it would first highlight the most relevant segments (a sequence of relevant and consecutive frames) in the video through visual annotations added under the video progress bar. These annotations are in the form of square-shaped blue buttons, as seen under the video in Fig. 4. These annotated segments are buttons, and when clicked, the video jumps to the start of that segment, and new information is loaded that explains why this segment is relevant to the query answer. By default, the first identified segment is selected.

Each segment includes detailed explanations as to why it is related to the system answer. We display this information on the right-side of the interface, as seen in Fig. 4. On the top right, the summary of detected video components (*action*, *objects*, and *locations*) for the given query is shown, which represent the highest ranking combinations of components the model detected in this segment. On the bottom right, the interface shows the component scores that summarize the marginal probabilities of single components in the selected segment. 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 MODEL EVALUATION

The model for the activity detection system was trained using a publicly-available video dataset, the Textually Annotated Cooking Scenes (TACoS) dataset [50], which consists of videos of several different cooking-related activities. For example, a typical video will have a person take out a vegetable from the refrigerator, wash it, cut it, and then cook it. The cooking context has the advantage of being easily understandable, even without particular domain expertise. The dataset includes hand-annotated labels of actions, objects, and locations for each frame of video. We isolate 28 such labels and use videos with only these labels for our experiments. Most videos are around 2 minutes in length (although videos as long as 15 minutes are also present). We used different sets of

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.

videos for training and testing. For training, we used 60313 frames and for testing we used 9355 frames.

5.1 Model (Machine Learning) Evaluation

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 predicting each activity correctly is a multi-label classification task, we use K-Group measures. Formally, a K-i measure where i is an integer is defined as the percentage of instances where i labels out of the total number of labels were predicted correctly. We report K-1, K-2, and K-3 since each activity is comprised of three entities: *action*, *object* and *location*. In addition, we also use standard measures such as the Hamming Loss and the Jaccard Index. Hamming loss is defined as the average fraction of incorrect labels (smaller the better). Jaccard index is the ratio between the cardinality of the intersection of the predicted labels and the true labels and the cardinality of the union of the predicted and true labels (higher the better). We observe that dynamic CCNs are more accurate than CCNs, which in turn are more accurate than GoogLeNet.

Thus, our results clearly show that reasoning about relationships between the various labels via CCNs and temporal constraints via DCCNs improves the accuracy of deep neural networks. Next, we evaluate the quality of explanations output by our system using human subjects studies.

6 HUMAN SUBJECTS EVALUATION

To evaluate the overall effectiveness of the explanations in our video activity recognition system, we designed and conducted a human-centered experiment.

6.1 Goals and Hypotheses

Video activity recognition (AR) systems are valuable and have many real-world applications, from fire detection [27] and airport security [60], to elderly care [23] and autonomous vehicles [48]. As alluded to in the introduction, many state-of-the-art models exist that yield high accuracy on the AR task. However, no matter how accurate the model, these kinds of models typically suffer from false positives, which may be highly undesirable in mission-critical applications. At the very least, as users of these systems, we would like to predict the circumstances under which the system would be likely to generate erroneous results and if it does, what these results might look like. Thus, human-AI collaboration plays an important role in identifying the weaknesses of such systems. For this purpose, it is crucial that human users maintain a proper understanding of the systems and how they work in order to understand when to rely on them. This is the problem we expect to be addressed by the Explainable Activity Recognition (XAR) framework that we defined earlier.

The goal of this study was to measure the degree to which the explanations generated by our system would benefit non-expert end users with little to no understanding of how such systems work. We hypothesized that both the speed and the accuracy of the decision-making process would increase significantly with the introduction of explanations. We also hypothesized that the user's level of agreement with the system's answers would vary significantly if explanations were shown. A user's answer is said to 'agree' with the system's when they are the same.

6.2 Experimental Design

The task used for evaluating user performance involved answering a series of questions (or queries) over a set of videos with the help of the system. The participants were divided into two groups: one *with* and the other *without* explanations. Participants from both groups had access to the video player and the system's answer to each question. Participants in the *with explanations* category used the interface with all the explanation components available (i.e., the video segments, the detected combination of components, and the component scores), while participants in the *without explanations* category did not have these components shown (i.e. they were only able to view the system's answers and nothing else). The experiment was conducted between-subjects in order to allow the same set of questions and videos to be used for both groups and also to avoid learning effects about knowledge of the model performance.

The TACoS dataset that was used for training and evaluating our system was used here as well since cooking videos typically do not require any domain-specific knowledge. Each user was presented with 20 queries spread out over 4 videos (i.e., 5 queries per video). Each query was a simple yes/no question about the video and had a single, unique answer (e.g., "Does the person cut a carrot?"). Participants of each group were presented with the system's answer to each of these queries and their task was to determine the true answer by watching the video and using the system's answers as point of reference.

Since the goal of the study was to evaluate whether the explanations helped the participants perform better than the model alone, it was necessary that the system made enough errors so that the explanations would actually be useful to determine the actual answer and that this improvement could be measured.

It was, therefore, imperative that the task include multiple queries where the system provided incorrect answers so that participants would have opportunities to recognize system errors. However, since the actual model had a high accuracy score (refer to Table 1), using a set of sample queries representative of the actual model accuracy would not have provided enough opportunities to view system errors since the system would have been too accurate for participants to see enough errors in the limited time of the study. To address this problem, we constrained the system accuracy to a constant 80% for this phase of the evaluation. This was done by controlling the composition of trials so that all participants experienced the system answering 80% of the queries correctly.

To assess task performance, we used the following three metrics: (1) error (2) time taken for task completion and (3) agreement. Error was calculated as the percentage of queries where the participant's answer was incorrect with respect to the (known) true answer. Agreement was measured as the percentage of queries where the participant's answer matched that of the system.

6.3 Procedure

The interface was a web-based application and the evaluation was conducted as an online study. We ran the experiment through the Amazon Mechanical Turk (AMT) crowdsourcing platform. The study was approved by our organization's Institutional Review Board (IRB), and participants were compensated at a fixed rate per hour. The experiment consisted of a single session with completion time of the participants varying from 25 minutes to 55 minutes.

The study opened with a consent form followed by a background questionnaire asking general information about participant demographics, education, and occupation. The participants were also provided with instructions on how to complete the task as well as a small tutorial prior to the main query review trials.

We slightly modified the interface described in Section 4.3 to (1) control the sequence of viewed videos and queries, and (2) include buttons for participants to provide their own answer for each query. Thus, instead of allowing participants to choose from the set of existing videos and queries, the interface only showed one video and one corresponding query at a time. For each trial, the system's answers were available, but the participants were asked to answer each query themselves. After providing their answer, the system showed participants the correct answer (which was sometimes different from the system's answer to simulate 80% accuracy as mentioned earlier) as feedback to help them estimate the system's simulated accuracy as well as build an understanding of how the system made its decisions which, in turn, would help them decide whether or not to rely on its answers. Participants were not allowed to change their response after submission.

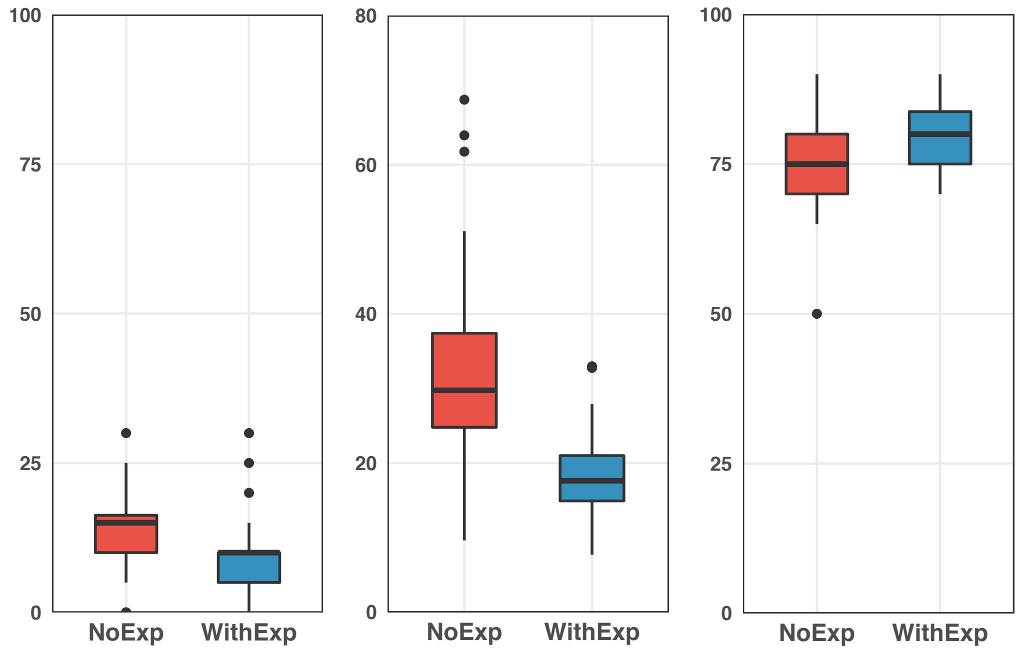
6.4 Participants

The experiment was completed online by 80 AMT workers. Of these participants, 40 of them were shown explanations while the other 40 were not. With the exception of a single participant who reported himself to be a programmer, all the others had occupations that were not related to data science, machine learning, and statistics; hence, the participants were non-experts with regards to machine learning knowledge. After pre-processing the data and removing outliers that did not fall within $1.5 \times$ IQR, we analyzed results from 38 participants for the *with explanations* category and 40 for the *without explanations* category.

6.5 Results

We analyzed the results using the Kruskal-Wallis non-parametric test to measure the difference between the two groups. The plots are shown in Fig. 6. We observed a significant difference on error per trial $\left(\chi^2(1,76) = 5.63, p < 0.05\right)$, showing that the participants *with explanations* had significantly less error than those *without explanations*. Our experiment also detected a significant difference on average time per trial $\left(\chi^2(1,76) = 28.1, p < 0.001\right)$. Participants *with explanations* were significantly faster. Together, these results support our hypothesis that the addition of our explanations significantly improves user task-performance in our system.

Additionally, the results from the user agreement with the system show that participants *with explanations* significantly agreed with the system more than participants *without explanations* $\left(\chi^2(1,76) = 8.00, p < 0.01\right)$. This might mean that providing more information helped participants understand the system and judge when it was correct. It would seem that the explanations encouraged participants to *correctly* trust its output. Since the *with explanations* category also had significantly better performance results, this suggests that the higher rate of agreement was not simply blind trust or *automation bias* [15], where humans tend to trust an intelligent system by virtue of its 'intelligence' alone. Rather, the results of this study suggest that agreement was appropriately aligned with the queries where the system provided the correct answer. However, it is to be noted that our study was not designed to specifically focus on the potential effects of explanations on automation bias, and therefore this still remains an open area for further research.



(a) **Average Error per Trial (Percentage)**

(b) **Average Time per Trial (Seconds)**

(c) **User Agreement with the System (Percentage)**

Metric	No Explanation	With Explanation
Accuracy	86.88% ± 7.4%	**90.26% ± 6.03%**
Speed (sec)	986.25 ± 561.35	**517.42± 198.1**
Agreement	74.38% ± 7.78%	**78.95%± 4.95%**

(d) Mean & standard deviation for performance and agreement of novice participants.

Fig. 6. The plots show the distribution of user responses based on user task-performance and agreement with the system among our two study conditions. Lower scores for both measures indicate better performance, i.e., lower errors and less performance time per each trial. The table is a summary of findings through mean & standard deviation of each metric. These findings align with the results from Kruskal-Wallis non-parametric test reported in section 6.5. We measured the average user error and time per trial and the fraction of instances on which their answer agreed with the system's answer. Bold results indicate significantly higher score.

7 DISCUSSION AND CONCLUSION

In this paper, we proposed a new explainable framework for activity recognition (AR), which we call explainable activity recognition (XAR). We surmised that such a framework would use Explainable Artificial Intelligence (XAI) techniques to provide enough model transparency to the users such that it will allow them to: (1) build a good mental model of how the system functions; (2) use and interact with the system more effectively, specifically understanding when it will succeed and when it is likely to fail; and (3) build trust in the system.

We then proposed a general approach for building an XAR/XAI system. Our approach uses the following pipeline:

(1) build an accurate model for activity recognition using deep neural network architectures and learning algorithms
(2) build a tractable probabilistic model over the interpretable random variables in the application domain using the output of the deep learning model as input treating the latter as a noisy sensor; the tractable model further improves the accuracy of the deep learning model
(3) answer queries posed by the user and generate explanations by performing probabilistic inference over the tractable model; tractability ensures that query answers and explanations are accurate and can be generated in real-time

We applied this general approach to build an XAR system for activity recognition in cooking videos, specifically on the TACoS video dataset [51] where activity is defined as a *(action, object, location)* triple. Our system had two-layers; the first layer used a deep convolutional neural network called GoogLeNet [58] and the second layer used a new tractable model called dynamic conditional cutset networks (DCCNs). The latter is a novel representation which extends and generalizes the recently proposed conditional cutset networks representation [46] to temporal domains.

In our system, GoogLeNet helped detect complex spatial patterns in each frame of each video while the DCCN helped capture the relationships between the various activities as well as temporal dynamics. The DCCN answered queries posed by the user and provided explanations via fast, accurate probabilistic inference. It also helped decipher the output of the black-box GoogLeNet architecture by summarizing and aggregating its decisions (see "Video explanations" in Fig. 5), suggesting alternative hypotheses that are likely to be true (see "Detected Combinations of components" in Fig. 5) and providing confidence on its detected components (see "Component Score" in Fig. 5).

We evaluated our system along two dimensions: prediction accuracy and explanation effectiveness. Via a thorough ablation based empirical evaluation, we found that the "explainable model" which combines a DCCN and a neural network is superior in terms of prediction accuracy to a "non-explainable model" which only uses a neural network. This verifies our hypothesis that DCCN corrects the errors made by the neural network by leveraging temporal information as well as relationships between activities. The usefulness of our explanations was also corroborated by the user studies where most of the users believed that the explanations helped them solve the question-answering task with greater ease and also gave them a better understanding of how the model made its decisions which, in turn, increased their trust in the system.

7.1 Future Work

Although our new dynamic conditional cutset network framework generates high-quality samples from the posterior distribution, both MAR and MAP inference over it are intractable (or NP-hard) in general. To this end, one line of future work is to investigate temporal models on which both MAP and MAR inference tasks can be solved in polynomial time. We are currently investigating specific structural constraints for achieving this objective.

In this paper, we only considered simple selection queries with yes/no answers for the sake of simplicity and brevity. An interesting direction to expand upon would be to use more complex kinds of queries such as counting queries (e.g., *how many carrots are there in the video?*) and time-based queries (e.g., *when was the first time the user washed his hands?*) which would require event ordering. In order to achieve this objective, we will have to develop a novel representation for activities and build an ontology to represent hierarchies of activities. This would allow for other interesting queries involving super-activities and sub-activities. A simple query of this type might be something like "Does the person in the video cook a potato?" which might consist of the sub-activities *(cut, potato, ∗)*, *(move, potato, pot)*, *(move, pot, stove)*, *(turn on, stove, ∗)* and *(turn off, stove, ∗)*. Since these are cooking videos, we might even ask the system if the person in the video follows the recipe correctly. Once our system is able to answer these kinds of complex queries, we could then think about more varieties of explanations that might be tailored to specific kinds of queries. Finally, we would re-design our human studies to accommodate these new queries and explanations and evaluate their usefulness to different categories of end users.

8 ACKNOWLEDGEMENTS

This work was supported by the DARPA Explainable Artificial Intelligence (XAI) Program under contract number N66001-17-2-4032, and by the National Science Foundation grants IIS-1652835, IIS-1528037, and IIS-1762268.

REFERENCES

[1] Francis R Bach and Michael I Jordan. 2002. Thin junction trees. In *Advances in Neural Information Processing Systems*. 569–576.

[2] Jessa Bekker, Jesse Davis, Arthur Choi, Adnan Darwiche, and Guy Van den Broeck. 2015. Tractable Learning for Complex Probability Queries. In *Advances in Neural Information Processing Systems*. 2242–2250.

[3] Aaron F Bobick and James W Davis. 2001. The recognition of human movement using temporal templates. *IEEE Transactions on Pattern Analysis & Machine Intelligence* (2001), 257–267.

[4] Aaron F Bobick and Andrew D Wilson. 1997. A state-based approach to the representation and recognition of gesture. *IEEE Transactions on Pattern Analysis & Machine Intelligence* (1997), 1325–1337.

[5] Hilary Buxton and Shaogang Gong. 1995. Visual surveillance in a dynamic and uncertain world. *Artificial Intelligence* (1995), 431–459.

[6] Lee W Campbell and Aaron F Bobick. 1995. Recognition of human body motion using phase space constraints. In *Proceedings of the IEEE International Conference on Computer Vision*. 624–630.

[7] C. K. Chow and C. N Liu. 1968. Approximating Discrete Probability Distributions with Dependence Trees. *IEEE Transactions on Information Theory* 14 (1968), 462–467.

[8] Adnan Darwiche. 2000. A Differential Approach to Inference in Bayesian Networks. In *Proceedings of the Sixteenth Conference in Uncertainty in Artificial Intelligence*. 123–132.

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

[10] Nicola Di Mauro, Antonio Vergari, and Floriana Esposito. 2016. Multi-label classification with cutset networks. In *Conference on Probabilistic Graphical Models*. 147–158.

[11] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE International Conference on Computer Vision & Pattern Recognition*. 2625–2634.

[12] Arnaud Doucet, Nando De Freitas, Kevin Murphy, and Stuart Russell. 2000. Rao-Blackwellised particle filtering for dynamic Bayesian networks. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*. 176–183.

[13] Xuguang Duan, Wenbing Huang, Chuang Gan, Jingdong Wang, Wenwu Zhu, and Junzhou Huang. 2018. Weakly supervised dense event captioning in videos. In *Advances in Neural Information Processing Systems*. 3059–3069.

[14] Georgia Gkioxari and Jitendra Malik. 2015. Finding action tubes. In *Proceedings of the IEEE International Conference on Computer Vision & Pattern Recognition*. 759–768.

[15] Kate Goddard, Abdul Roudsari, and Jeremy C Wyatt. 2011. Automation bias: a systematic review of frequency, effect mediators, and mitigators. *Journal of the American Medical Informatics Association* (2011), 121–127.

[16] Shaogang Gong and Tao Xiang. 2003. Recognition of Group Activities using Dynamic Probabilistic Networks.. In *Proceedings of the IEEE International Conference on Computer Vision*. 742–749.

[17] David Gunning. 2019. DARPA's Explainable Artificial Intelligence (XAI) Program. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*. ii–ii.

[18] Kevin Anthony Hoff and Masooda Bashir. 2015. Trust in automation: Integrating empirical evidence on factors that influence trust. *Human factors* 57, 3 (2015), 407–434.

[19] Robert R Hoffman. 2017. A Taxonomy of Emergent Trusting in the Human–Machine Relationship. *Cognitive systems engineering: The future for a changing world* (2017), 137–163.

[20] Timothy Huang, Daphne Koller, Jitendra Malik, G Ogasawara, Bobby S Rao, Stuart J Russell, and Joseph Weber. 1994. Automatic Symbolic Traffic Scene Analysis Using Belief Networks. In *Proceedings of the Twelfth AAAI Conference on Artificial Intelligence*. 966–972.

[21] Matthew J Johnson, David K Duvenaud, Alex Wiltschko, Ryan P Adams, and Sandeep R Datta. 2016. Composing graphical models with neural networks for structured representations and fast inference. In *Advances in Neural Information Processing Systems*. 2946–2954.

[22] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. 2014. Large-Scale Video Classification with Convolutional Neural Networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 1725–1732.

[23] Zafar A Khan and Won Sohn. 2011. Abnormal human activity recognition system based on R-transform and kernel discriminant technique for elderly home care. *IEEE Transactions on Consumer Electronics* (2011), 1843–1850.

[24] Daphne Koller and Nir Friedman. 2009. *Probabilistic Graphical Models - Principles and Techniques.* MIT Press.

[25] Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. 2017. Dense-captioning events in videos. In *Proceedings of the IEEE International Conference on Computer Vision*. 706–715.

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*. 1097–1105.

[27] Tai Yu Lai, Jong Yih Kuo, Yong-Yi Fanjiang, Shang-Pin Ma, and Yi Han Liao. 2012. Robust little flame detection on real-time video surveillance system. In *Third International Conference on Innovations in Bio-Inspired Computing and Applications*. 139–143.

[28] Ivan Laptev. 2005. On space-time interest points. *International Journal of Computer Vision* (2005), 107–123.

[29] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* (1998), 2278–2324.

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

[31] Yitao Liang, Jessa Bekker, and Guy Van den Broeck. 2017. Learning the Structure of Probabilistic Sentential Decision Diagrams. In *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence.*

[32] Min Lin, Qiang Chen, and Shuicheng Yan. 2013. Network in network. *arXiv preprint arXiv:1312.4400* (2013).

[33] Daniel Lowd and Pedro Domingos. 2008. Learning Arithmetic Circuits. In *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence.*

[34] R. Mateescu and R. Dechter. 2005. AND/OR Cutset Conditioning. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 230–235.

[35] Nicola Di Mauro, Antonio Vergari, and Teresa Maria Altomare Basile. 2015. Learning Bayesian Random Cutset Forests. In *Foundations of Intelligent Systems - 22nd International Symposium*. 122–132.

[36] Bonnie M Muir. 1994. Trust in automation: Part I. Theoretical issues in the study of trust and human intervention in automated systems. *Ergonomics* 37, 11 (1994), 1905–1922.

[37] Bonnie M Muir and Neville Moray. 1996. Trust in automation. Part II. Experimental studies of trust and human intervention in a process control simulation. *Ergonomics* 39, 3 (1996), 429–460.

[38] K. P. Murphy. 2002. *Dynamic Bayesian networks: representation, inference and learning.* Ph.D. Dissertation. University of California, Berkeley.

[39] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. 2015. Beyond Short Snippets: Deep Networks for Video Classification. In *Computer Vision and Pattern Recognition.*

[40] Alexandru Niculescu-Mizil and Rich Caruana. 2005. Predicting good probabilities with supervised learning. In *Proceedings of the Twenty-Second International Conference on Machine learning*. 625–632.

[41] Judea Pearl. 1988. *Probabilistic reasoning in intelligent systems: networks of plausible inference.* Morgan Kaufmann.

[42] Hoifung Poon and Pedro Domingos. 2011. Sum-product networks: A new deep architecture. In *IEEE International Conference on Computer Vision Workshops.*

[43] Lawrence R Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE* 77, 2 (1989), 257–286.

[44] T. Rahman and V. Gogate. 2016. Learning Ensembles of Cutset Networks.. In *AAAI.*

[45] Tahrima Rahman and Vibhav Gogate. 2016. Merging Strategies for Sum-Product Networks: From Trees to Graphs.. In *UAI.*

[46] Tahrima Rahman, Shasha Jin, and Vibhav Gogate. 2019. Cutset Bayesian networks: a new representation for learning Rao-Blackwellised graphical models. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. 5751–5757.

[47] Tahrima Rahman, Prasanna Kothalkar, and Vibhav Gogate. 2014. Cutset networks: A simple, tractable, and scalable approach for improving the accuracy of Chow-Liu trees. In *Joint European conference on machine learning and knowledge discovery in databases*. 630–645.

[48] Akshay Rangesh, Eshed Ohn-Bar, Kevan Yuen, and Mohan M Trivedi. 2016. Pedestrians and their phones-detecting phone-based activities of pedestrians for autonomous vehicles. In *Proceedings of the Nineteenth IEEE International Conference on Intelligent Transportation Systems*. 1882–1887.

[49] Cen Rao and Mubarak Shah. 2001. View-invariance in action recognition. In *Proceedings of the IEEE International Conference on Computer Vision & Pattern Recognition*. II–II.

[50] Michaela Regneri, Marcus Rohrbach, Dominikus Wetzel, Stefan Thater, Bernt Schiele, and Manfred Pinkal. 2013. Grounding action descriptions in videos. *Transactions of the Association for Computational Linguistics* 1 (2013), 25–36.

[51] Anna Rohrbach, Marcus Rohrbach, Wei Qiu, Annemarie Friedrich, Manfred Pinkal, and Bernt Schiele. 2014. Coherent multi-sentence video description with variable level of detail. In *German conference on pattern recognition*. 184–195.
[52] Amirmohammad Rooshenas and Daniel Lowd. 2014. Learning sum-product networks with direct and indirect variable interactions. In *Proceedings of the Thirty-First International Conference on Machine Learning*. 710–718.
[53] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision* 115, 3 (2015), 211–252.
[54] Eli Shechtman and Michal Irani. 2005. Space-time behavior based correlation. In *Proceeedings of the IEEE International Conference on Computer Vision & Pattern Recognition*. 405–412.
[55] Young Chol Song, Iftekhar Naim, Abdullah Al Mamun, Kaustubh Kulkarni, Parag Singla, Jiebo Luo, Daniel Gildea, and Henry A Kautz. 2016. Unsupervised Alignment of Actions in Video with Text Descriptions. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. 2025–2031.
[56] Thad Starner and Alex Pentland. 1995. Real-time american sign language recognition from video using hidden markov models. In *Proceedings of the International Symposium on Computer Vision*. 265–270.
[57] Christian Szegedy. 2014. Googlenet pre-trained model. http://dl.caffe.berkeleyvision.org/bvlc_googlenet.caffemodel.
[58] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE International Conference on Computer Vision & Pattern Recognition*. 1–9.
[59] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. 2015. Learning Spatiotemporal Features With 3D Convolutional Networks. In *The IEEE International Conference on Computer Vision (ICCV)*.
[60] Rajesh Kumar Tripathi, Anand Singh Jalal, and Subhash Chand Agrawal. 2018. Suspicious human activity recognition: a review. *Artificial Intelligence Review* 50, 2 (2018), 283–339.
[61] Limin Wang, Yu Qiao, and Xiaoou Tang. 2015. Action recognition with trajectory-pooled deep-convolutional descriptors. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*. 4305–4314.
[62] Junji Yamato, Jun Ohya, and Kenichiro Ishii. 1992. Recognizing human action in time-sequential images using hidden markov model. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*. 379–385.
[63] Yezhou Yang, Yi Li, Cornelia Fermuller, and Yiannis Aloimonos. 2015. Robot learning manipulation action plans by" Watching" unconstrained videos from the world wide web. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*.