Published: IEEE Transactions on Multimedia, (Early Access), December 2018.

Long Activity Video Understanding using Functional Object-Oriented Network

Ahmad Babaeian Jelodar, David Paulius, and Yu Sun

***Abstract*—Video understanding is one of the most challenging topics in computer vision. In this paper, a four-stage video understanding pipeline is presented to simultaneously recognize all atomic actions and the single ongoing activity in a video. This pipeline uses objects and motions from the video and a graph-based knowledge representation network as prior reference. Two deep networks are trained to identify objects and motions in each video sequence associated with an action and low Level image features are used to identify objects of interest in the video sequence. Confidence scores are assigned to objects of interest to represent their involvement in the action and to motion classes based on results from a deep neural network that classifies an ongoing action in video into motion classes. Confidence scores are computed for each candidate functional unit to associate them with an action using a knowledge representation network, object confidences, and motion confidences. Each action, therefore, is associated with a functional unit, and the sequence of actions is evaluated to identify the sole activity occurring in the video. The knowledge representation used in the pipeline is called the functional object-oriented network, which is a graph-based network useful for encoding knowledge about manipulation tasks. Experiments are performed on a dataset of cooking videos to test the proposed algorithm with action inference and activity classification. Experiments show that using a functional object-oriented network improves video understanding significantly.**

***Index Terms*—Video Understanding, Activity Understanding, Video Knowledge Representation.**

I. Introduction

VIDEO understanding is a challenging topic that requires completion of several difficult steps successfully, where each step is a challenging and active research topic by itself. It would usually require the video to be automatically split into atomic actions, the activities and objects in the atomic video clip to be successfully recognized, and a meaningful understanding inferred based on the activities and objects. For each step, extensive learning would be carried out for object recognition, activity recognition, and video splitting, but these are usually done individually.

We propose to learn the relationship between objects, actions, and activities and represent those relationships in a graph. We use the graph as structured prior information for video understanding when possible. For example, a video that demonstrates a chef who is cooking an omelet comprises multiple consecutive actions, and each action, such as mixing eggs in a bowl, employs multiple objects such as a bowl, a whisk, and eggs. To identify the actions, the structural information between the objects (bowl, whisk, eggs) and motions (mixing) can be useful. For instance, if we understand that eggs can be mixed using a whisk, we can associate the object whisk with the objects egg and bowl.

A. B. Jelodar, D. Paulius, and Y. Sun are with the Department of Computer Science and Engineering, University of South Florida, Florida FL, 33620 e-mail: (ajelodar,davidpaulius,yusun@mail.usf.edu).

The structural information between consecutive actions can be applied to interpret an activity in a video. For example, cracking eggs into a bowl occurs before mixing the eggs in the bowl. Consequently, this knowledge can help with predicting that the current ongoing action is mixing, knowing that the previous action was cracking eggs. Embedding these informative structures into a prior graphical structure and using the embedding for inference at test time can improve video understanding.

We use the coordination encoded in the object nodes (e.g., bowl or eggs) and motion nodes (e.g., stirring) of the knowledge-based graph presented in [1], [2] to recognize actions (such as stirring eggs) in videos. The knowledge-based network used for task inference, called the *functional object-oriented network* or FOON [1], encodes knowledge about the flow of actions coming one after another. Using this network, we present a powerful object-oriented inference algorithm for action and activity recognition.

We propose a pipeline that deploys object localities and their motion features to identify active objects within an action. We train a deep model for holistic motion recognition, which helps with cases in which the object (e.g., salt in a chef's hand) is not easily detectable. The identified objects and motion are fed to the inference stage with FOON to provide a list of candidate functional units that can be associated with the current ongoing action (e.g., cracking egg in a bowl). The consecutive predicted functional units are evaluated to understand the activity performed in the video (e.g., making an omelet). This work has four main contributions:

- Integrating object localities, object flow features, and their accordances from the functional object-oriented knowledge representation for action recognition.
- Deploying the prior structural information between objects and motions in the functional object-oriented network for functional action recognition in video (e.g., using the relationship between the objects egg, fork, and bowl to interpret and label the action as "stirring eggs in a bowl with a fork").
- Using the structural information of consecutive actions in the functional object-oriented network for task inference (e.g., recipe classification based on a list of consecutively predicted actions).
- Merging a deep neural network for motion recognition with the FOON knowledge representation for functional action recognition.

The remainder of this paper is organized as follows: in Section II, we discuss the related work, in Section III, we describe the functional object-oriented network [1], and in Section IV, we introduce the algorithm pipeline. In Section V, we explain how objects of interest are identified and in Section VI, we describe the procedure of functional unit recognition. In Section VII, we discuss experiments and results, and we conclude our findings in Section VIII.

II. Related Work

A. Knowledge Representation

Knowledge representations have been successfully applied to robotics and machine learning [3] and in natural language processing for Wordnet [4], Verbnet [5], and Framenet [6]. In [7], Carlson et al. propose a knowledge-based architecture to learn a language from web text. Some works introduce and use a knowledge base for answering queries [8], visual queries [9], and cuisine- and ingredient-oriented queries using deep features [10]. Knowledge-based methods have been also used in visual applications such as the ontological hierarchical knowledge base for image content retrieval and video event detection [11], scene understanding [12], description logics for scene interpretation [13], visual structured knowledge base for scene recognition and object detection [14], and a combination of various knowledge based representations using machine learning and statistical approaches [15]. In [16], the problem of object affordance reasoning is modeled using a knowledge base representation. In [17] a visual knowledge-based representation and dataset are introduced for modeling relationships in images. In [18], a knowledge representation-based method for food recognition from an image was proposed, which is close to our application. The lack of a structured knowledge representation for joint object and motion representation motivated applying the functional object-oriented network for video understanding in cooking videos.

B. Video Understanding

There is a broad area of work in video understanding. Some works deploy costly setups such as physical sensors or additional modalities (e.g., text) [19]–[21], and some research performs analysis on spatio-temporal features of a sequence in a holistic manner to label actions [22]–[25] or uses spatio-temporal features of a person (e.g., models of joints or pose) to classify actions [26], [27]. These methods are incapable of handling variations in view, zoom, and occlusion easily. Simultaneous video segmentation and understanding [28], [29], [30] is also a very common research area. These methods usually do not consider objects or variations in pose. Some approaches extract and analyze a selection of frames for video event summarization [31], and fast anomaly concentration and detection [32]. Jain et al. propose a method that embeds structure into a deep model [33] to incorporate knowledge with deep models for activity recognition. Other deep approaches proposed for activity recognition are [34], [35]. The motivation to incorporate FOON for video understanding is based on the group of research that use objects and their affordances and states in a video for action recognition [36]–[41].

Currently, there are various multi-view applications, especially in surveillance systems. Information from multiple cameras can enhance event summarization or task understanding. Several researchers have proposed methods for handling multi-camera scenarios. Event summarization in multi-view videos using a deep learning approach [35], detection and summarization of an event in multi-view surveillance videos by applying boosting [42], and a machine learning ensemble method [43] are instances of research in the area of multi-view video understanding. This aspect of video understanding has not been addressed in this work; however, the proposed framework can be deployed in multi-view systems. A discussion on the multi-view aspects of our video understanding pipeline is included in Section VII-E.

C. Knowledge Representation for Video Understanding

Various approaches have been proposed to use knowledge representation for video understanding, such as semantic-visual knowledge bases like FrameNet and Imagenet for modeling rich event-centric concepts and their relationships for video event detection [44], a knowledge- and probabilistic-driven framework for activity recognition [45], and semantic representations for event detection [46], [47]. Souza et al. deploy objects, actions and their bonds into graphs and use simulated annealing for event inference using temporal connections [48], [49]. Ren et al. [50] previously proposed a Bayesian framework that uses object motions and their relationships to improve object recognition reliability. This model enables robots to learn the interactive functionalities of objects from human demonstrations [51] [52].

Object information and analysis is an essential aspect for activity recognition. The method in [53] deploys spatial and functional constraints on the relationships between objects and motions to semantically interpret videos. Modeling the mutual context of human pose and objects using a random field model [54], modeling relationships between object parts and people in the scene using contextual scene descriptors and Bayesian learning [55], and encoding objects for action classification and localization are examples of work on video understanding using object information. These works all assume that a person is performing the act in the video, and, therefore, the human pose would be essential for their approaches. We follow the path of incorporating objects and extend it to the goal of action recognition and activity inference by deploying our previously proposed knowledge representation network [1]. Our work is different from the noted object-based activity recognition methods, in that our videos do not contain a person and its pose. We use only the human hand and its location, if available in the scene, as features to interpret the video.

III. Functional Object-Oriented Network

FOON is a knowledge representation for encoding knowledge about manipulation tasks and, in extension, object affordances. A FOON can also be used by a robot for solving manipulation problems given a target goal. Currently, FOON focuses on learning activities in the cooking and kitchen domain, but it can also be extended to other domains and environments.

A. FOON Basics

A FOON is a directed acyclic graph that contains two types of nodes (object and motion), making it a bipartite network [56]. Figure 1 depicts a sample functional unit, the basic building block of a FOON.

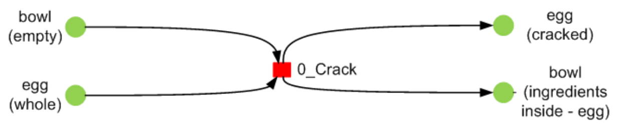


Fig. 1. Example of a functional unit with 2 input object nodes (green), 2 output object nodes (green), and 1 motion node (red).

Object nodes are defined as items that are being manipulated or acted upon by a demonstrator, and motion nodes describe the action being applied on objects such as cutting or mixing. An object node (N_O) is identified by its *object type*, an *object state*, and a *motion identifier*, which denotes whether the object is in motion during activity. Objects can also serve as *containers* of other objects, and each node can be described by a list of ingredients. Motion nodes are identified only by their *motion type*. Within this graph, as in regular bipartite networks, edges connect a pair of nodes; specifically, an edge in FOON connects an object-motion pair. The edge direction indicates the order in which an object may change in its state through a motion action similar to Petri Nets [57], which require transitions to activate or fire place nodes.

B. Functional Unit

A FOON consists of subcomponents or learning units called functional units. Each functional unit describes a single, atomic action as seen in an activity (an activity or subgraph can be considered as a series of actions). For instance, in the activity of cooking scrambled eggs, one functional unit may describe the action of cracking an egg, and another may describe the action of mixing the eggs in a bowl. A functional unit describes the transition of objects states before and after a manipulation motion occurs; this is described by input object nodes (objects before manipulation) and output object nodes (objects after manipulation). In this paper, our focus is generating these functional units directly from instructional videos for learning future instances of how tasks are executed. A collection of subgraphs (or activities) that are merged together to combine knowledge and remove duplicate units is called a *universal* FOON. Each functional unit has three components: input object nodes, output object nodes, and a motion node that describes the action that possibly causes a change in the input objects' states, possibly causing a state change, because an action may not always incur a change of state. Each functional unit is also described by the time frames at which they are observed in an activity.

C. FOON Construction

The graph shown in Figure 2 consists of nodes from 65 videos that were annotated in the form of subgraphs, which consist of functional units that reflect each individual step in a cooking procedure. Edges would be drawn between an object node and motion node pair, where the object nodes are those seen in an action within the cooking activity and the motion node describes the action occurring. As we created these subgraphs and parsed them, we compiled a list of objects and motions to create labels for the different node instances seen and to enforce consistency in labels (subgraphs were created by multiple volunteers). When adding new information (subgraphs) from other datasets, we only need to annotate them to conform to the format of our graphs and parse them to get the labels correct. The merging procedure will add these newly parsed functional units to the network to ensure that there are no duplicates. This merging procedure is detailed more in our previous work in [1]. This is where this proposed work fits in; the task of automatically generating subgraphs from videos (especially those from other datasets) is difficult to do, and manual annotation can be time-intensive.

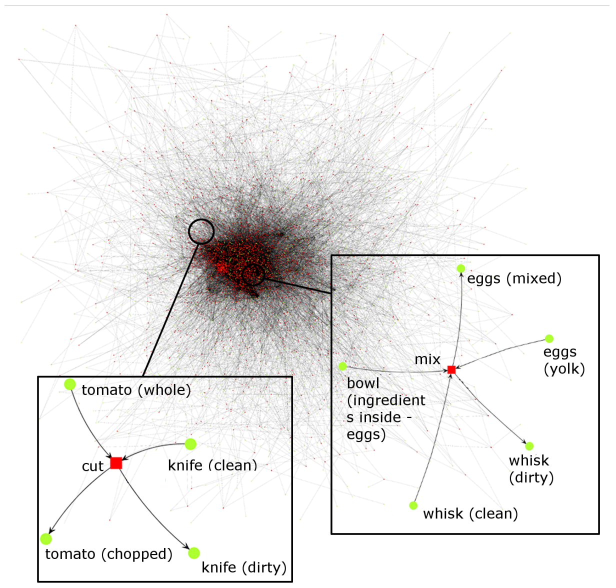


Fig. 2. Illustration of universal FOON with 4955 nodes (object and motion nodes). FOON comprises many functional units, such as those highlighted in image.

D. FOON Sources and Statistics

A FOON ideally is learned directly from human demonstrations, whether by video or from observation, and it is automatically generated from such demonstrations, although in the earlier phases of constructing FOON we opted to manually label YouTube videos as subgraphs. In the future, we will try to extend FOON using the method discussed in this paper. After recording all functional units for a video, we parsed the subgraph to ensure that all object and motion labels were consistent with all other subgraphs.

Each subgraph was then merged into a single network, referred to as a *universal FOON*. The merging procedure is as follows: using a list of all functional units in G_{FOON}, compare each functional unit in all subgraphs to this list and append those units of a subgraph which are not present in G_{FOON}.

In total, the network contains 1853 object nodes, 3102 motion nodes, and 15656 edges. Figure 2 illustrates the network described by these statistics.

E. FOON vs. Other Knowledge Representations

FOON is not the first knowledge representation to address video understanding. In this subsection, we discuss the main differences between FOON and previous work. Previous works in knowledge representation do not consider the joint representation of both objects and motions. Our work is inspired by the theory of affordance originally proposed in [58]. Many follow-up studies show that there is a link between manipulations and objects. Our objective is to create a graphical representation of manipulations where objects and motions describe affordance. In terms of graphical representations, previous works capture knowledge using probabilistic graphical methods or semantic graphs/trees. However, they do not create a knowledge base of activity from demonstrations that could then be used for performing (possible) new manipulations. In addition, for affordance studies, they would instead try to model the relationship between objects and simple actions to predict the effect or impact it has on them. A more general form of representation akin to FOON is Petri Nets, where place nodes are like object nodes and transition nodes are like motion nodes. Certain input places are needed to fire or execute a transition node, much like input object nodes must be available to execute a given manipulation motion.

IV. Video Understanding Pipeline

We propose a four-stage pipeline for video understanding. The pipeline identifies the objects and motions in a video sequence (associated with an action) and uses them together with the knowledge representation to assign a functional unit label to the event in action. An *action* refers to a single, atomic event, and a *sequence* of actions represents an entire activity. Consecutive identified actions will be analyzed as a whole to understand the activity (recipe) being executed in the video. The steps to the pipeline are as follows: 1) *functional object recognition*, 2) *functional motion recognition*, 3) *functional unit recognition*, and 4) *task graph inference*.

In the first stage of the pipeline, the *functional object recognition* stage, all objects are identified and scores are assigned to objects based on their usefulness in the scene. In the second stage, *functional motion recognition*, each action (a split of the video) is classified into its corresponding motion class. Using the results from the first two stages and their FOON accordances, each action is analyzed and associated with a functional unit in the *functional unit recognition* stage. Each recognized action (in the video) from a single video is assigned a functional unit and looked up in the FOON graph for them to eventually be classified as a whole activity (recipe). This last stage is referred to as *task graph inference*. An illustration of the video understanding pipeline is depicted in Figure 3.

A. Functional Object Recognition

We apply the well-known Faster R-CNN algorithm for localizing and labeling objects in the scene [59]. Faster R-CNN is a two-part convolutional network that detects object proposals and performs object classification simultaneously. The output of the Faster R-CNN network is a set of bounding boxes and their corresponding object class labels. We further identify the used objects in the video sequence, which we call objects-in-action, using three metricsthe closeness of the human hand to the object, the magnitude of optical flow, and the frequency in which the objects have been observed in the video. We explain the functional object recognition stage more thoroughly in Section V-A.

B. Functional Motion Recognition

In some cases, FOON is not able to correctly identify the action in video using only object features. For example, knowing that the objects *bowl* and *egg* are objects-in-action could lead to multiple FOON inferences, because various functional units contain the object nodes *bowl* and *egg* but have different motion nodes (e.g., pouring or cracking). In another example, when sprinkling salt with the hand, it is difficult to visually discern that the object *salt* is being used, but the hand motion will suggest the action of sprinkling.

To address these issues, we fine-tune the deep (CNN+LSTM) network by Donahue et al. [23] with 10 classes in the last layer. This network comprises a CNN portion and an LSTM portion. The frames of a sequence are, one by one, given as input to the CNN and the output of the CNN is given as input to the LSTM layer. The outputs of the LSTM layer are averaged to provide a final prediction for the class of the motion in action. The architecture of the CNN network contains five convolutional layers and two fully-connected layers. The initial five convolutional layers and a single fully-connected layer on top is fed to one layer of a recurrent LSTM layer. The output of the LSTM layer is followed by the classification layer. We modified the last layer so that the number of neurons in the last layer of the network contains ten neurons to reflect the 10 motion types we have selected for training. We train the CNN architecture and the CNN + LSTM architectures separately. We use the trained weights from [23] and perform training only for the last layer of classification. We report only the better results from the CNN+LSTM architecture.

Each motion class in this deep architecture is associated with a set of motion nodes in FOON. The network assigns confidence scores to each of the motion classes. A confidence score reflects the probability of a class being assigned as a label to the action happening in the video. For more details on the approach, readers are referred to the algorithm described in [23]. The output from this deep network is used to calculate confidences for each candidate functional unit in the functional unit recognition stage.

C. Functional Unit Recognition

We determine the meaning of a video by associating actions with functional units. Objects-in-action are looked up in the

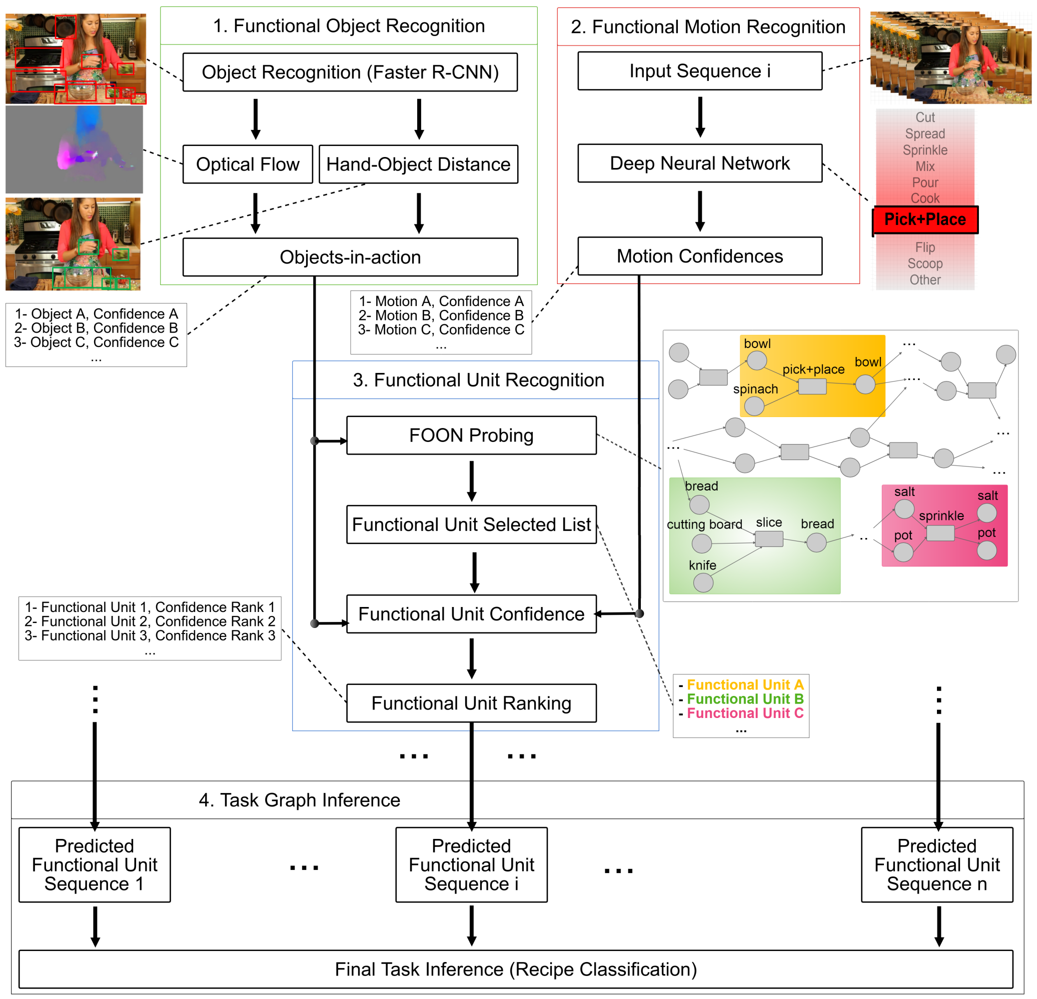

1. Functional Object Recognition
Object Recognition (Faster R-CNN)
Optical Flow
Hand-Object Distance
Objects-in-action
1- Object A, Confidence A
2- Object B, Confidence B
3- Object C, Confidence C
2. Functional Motion Recognition
Input Sequence i
Deep Neural Network
Motion Confidences
Cut
Spread
Sprinkle
Mix
Pour
Cook
Pick+Place
Flip
Scoop
Other
1- Motion A, Confidence A
2- Motion B, Confidence B
3- Motion C, Confidence C
3. Functional Unit Recognition
FOON Probing
Functional Unit Selected List
Functional Unit Confidence
Functional Unit Ranking
1- Functional Unit 1, Confidence Rank 1
2- Functional Unit 2, Confidence Rank 2
3- Functional Unit 3, Confidence Rank 3
bowl
pick+place
bowl
spinach
bread
cutting board
slice
bread
knife
salt
sprinkle
salt
pot
pot
- Functional Unit A
- Functional Unit B
- Functional Unit C
4. Task Graph Inference
Predicted Functional Unit Sequence 1
Predicted Functional Unit Sequence i
Predicted Functional Unit Sequence n
Final Task Inference (Recipe Classification)

Fig. 3. The pipeline for automatic functional unit identification.

universal FOON to identify candidate functional units. Candidate functional units are evaluated based on a confidence score computed in this stage and thoroughly discussed later in the paper. This consolidated confidence score incorporates both object confidences produced from the functional object recognition stage and motion confidences resulting from the functional motion recognition stage. The confidence score estimates how related each candidate functional unit is to the ongoing action in the present sequence. The list of candidate functional units is further sorted based on their confidences. Functional units with the highest confidences are associated with the current action.

D. *Task Graph Inference*

To identify the activity (sequence of actions) in a video, the identified actions throughout the video are used together with FOON look-up to predict the most likely activity label for that video.

V. Functional Object Recognition

We recognize and localize all objects in a video sequence (associated with an action) using the well-known Faster R-CNN algorithm [59]. We then quantify the involvement of each object in the current action by extracting optical flow features and calculating hand-object distances in each frame of the video sequence. A list of the most used objects is created and named objects-in-action.

A. *Recognizing Objects-in-action*

In this stage of the pipeline, we use the bounding box associated with each object for our computations. After localizing objects, the less frequent objects in the video are excluded. The center point of the bounding boxes resulting from the Faster R-CNN algorithm are used to calculate the *object's average distance from the hand*. The distances are further normalized using a Gaussian distribution. The *optical flow of objects* within the video sequence are computed. The proposed method in [60] is used to estimate the optical flow between two frames. The estimated optical flow and the objects' positions are incorporated to estimate the flow of each object. Objects with higher magnitude of optical flow are assigned a higher confidence valuea higher value conveys a higher chance that the object is moving and, hence, a higher probability that the object is being used in the video sequence. Equation 1 shows how these metrics are integrated to estimate a confidence for each object.

$$conf_{object} = \alpha.c_{flow} + \beta.c_{dist} + \gamma.c_{freq} \tag{1}$$

In Equation 1, c_{flow}, c_{dist}, and c_{freq} are the optical flow confidence, distance to hand confidence, and frequency confidence of each object, respectively; $conf_{Object}$ is the final calculated confidence of the object. Coefficients α, β and γ are tuned manually and represent how much each factor contributes to the final confidence of each object. Figure 4 depicts the procedure of identifying objects-in-action for a simple action of whisking eggs, using the three metrics mentioned in Equation 1.

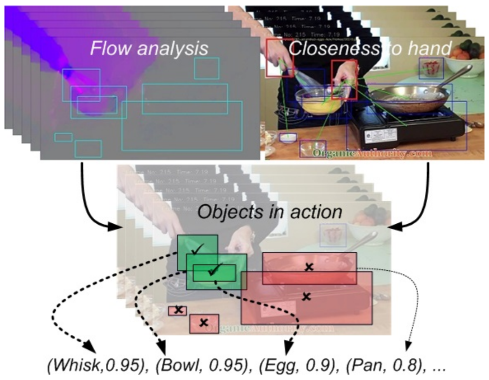


Fig. 4. Example showing procedure of identifying objects-in-action. Items such as *egg* and *whisk* would be possible candidates for participating in an egg-whisking motion.

In the example of Figure 4, we observe an egg-whisking motion occurring in which objects *egg*, *whisk*, and *bowl* are at the top of the list of objects-in-action and objects *pan* and *stove* have lower confidences.

VI. Functional Unit Recognition

Each action in the video is associated with the closest functional unit from FOON. To associate the correct functional unit with an action, unrelated functional units are filtered out. Filtering is performed using *functional unit confidence* estimation and *probing*, as discussed in this section.

A. Functional Unit Confidence

The pipeline recommends a list of in-use objects from the current action, named objects-in-action (Section V-A). Objects from the list are looked up in FOON, and functional units containing them are identified. The identified functional units are suggested as candidate functional units that can be associated with the current action in the video. Every functional unit contains several object nodes that may or may not be included in the list of objects-in-action. The overlap between the object nodes (of a functional unit) and the objects-in-action are named as the used set, and the remainder of the object nodes is tagged as the unused set. These two sets of objects are used to determine whether we should support or penalize a candidate functional unit. Equation 2 shows how the confidence of a candidate functional unit is estimated.

$$conf_{FOON} = \frac{\sum_{n=1}^{N_{used}} conf_n}{N_{used}} - penalty + \kappa.bonus \tag{2}$$

In this equation, $conf_{FOON}$ is the estimated confidence, N_{used}, is the number of object nodes in the used set of a candidate functional unit, and $conf_n$ is the confidence of each of those objects (subsection V-A). The $bonus$ term is estimated based on the pixel-wise overlap of all objects used in a candidate functional unit. This term represents the extent of interaction between the objects. The $penalty$ term calculated by Equation 3, represents the penalty applied to the estimated confidence.

$$penalty = \sum_{m=1}^{N_{notused}} \lambda.conf_m + \sum_{k=1}^{N_{extra}} \eta.conf_k \tag{3}$$

The confidence of the objects listed in the list of objects-in-action but not used in the candidate functional unit, $conf_m$, together with the confidence of the objects not listed as objects-in-action but used in the candidate functional unit, $conf_k$, contribute to the penalty. In this equation, $N_{notused}$ is the number of unused objects, and N_{extra} is the number of objects not listed but used in the candidate functional unit. In Equation 2, the constant κ tunes the effect of bonus and penalty. The constant λ in Equation 3 tunes the effect of unused objects on the penalty term, and the constant η tunes the effect of objects used but not listed. Figure 5 illustrates the procedure of confidence estimation for a candidate functional unit. The algorithm for confidence calculation is shown in Algorithm 1.

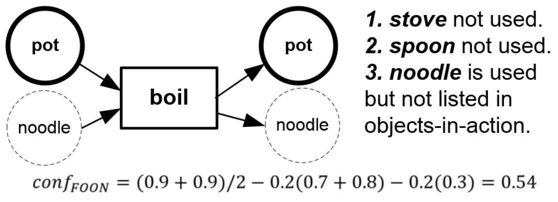

pot
noodle
boil
pot
noodle
1. stove not used.
2. spoon not used.
3. noodle is used but not listed in objects-in-action.
$conf_{FOON} = (0.9 + 0.9)/2 - 0.2(0.7 + 0.8) - 0.2(0.3) = 0.54$

Fig. 5. Illustration of functional unit confidence estimation. In this example, identified objects-in-action are *pot*, *spoon*, and *stove*, with confidences 0.9, 0.8, and 0.7 respectively (λ=η=0.2).

The confidence calculated in Equation 2 focuses solely on object interaction and their functional accordances. We believe motion can introduce additional information for confidence calculation. To include functional motion for estimating the confidence, we incorporate the outputs from the trained deep architecture (CNN+LSTM) for motion classification. We fuse the output of the CNN+LSTM network with the confidence estimated solely based on object interaction, $conf_{FOON}$. The output of the CNN+LSTM network for motion recognition has 10 confidence scores representing the probability of each of the motion classes happening. We rank the motion classes based on their resulted confidence scores. Finally, the confidences of functional units in Equation 2 are combined with the results

Algorithm 1 Confidence Calculation

```
1:  $list = \emptyset$ // holds objects and their confidences
2:  for $object \in sequence$ do
3:      $c_{dist} = abs(object - hand)$
4:      $c_{freq} = \textbf{frequency}(object)$
5:      $c_{flow} = \textbf{opticalFlow}(object)$
6:      $conf_{object} = \alpha.c_{flow} + \beta.c_{dist} + \gamma.c_{freq}$
7:      $list.append((object, conf_{object}))$
8:  end for
9:
10: $list.sort()$
11: $topK = list.selectTopK()$ // objects in action
12: // Find all candidate functional units associated with the top K objects
13: $candidates = \textbf{FOONLookUp}(topK)$
14:
15: for $c \in candidates$ do
16:     $nodes = c.getObjects()$
17:     $overlap = \textbf{objectOverlap}(nodes.objects, topK.objects)$
18:     $N_{used} = size(overlap)$
19:     $bonus = \textbf{pixelOverlap}(nodes.objects)$
20:     $unused = (topK - overlap) + (nodes - overlap)$
21:     $penalty = average(unused.confidences)$
22:
23:     $conf_{FOON}(c) = \frac{\sum_{n=1}^{N_{used}} conf_{object}(n)}{N_{used}} - penalty + \kappa.bonus$
24: end for
```

from the CNN+LSTM network to extract a final confidence for the functional units as shown in Equation 4.

$$conf_{motion} = conf_{FOON} + \alpha.conf_{LSTM} \tag{4}$$

In Equation 4, $conf_{FOON}$ is the confidence calculated in Equation 2, and $conf_{LSTM}$ is the confidence calculated based on results from the CNN+LSTM network. Coefficient α balances the effect of each of those parameters.

B. Probing

Each object is individually looked up in FOON, and all functional units containing that object are identified. A list of candidate functional units containing the object is acquired. The list contains candidate functional units that may associate with the current action. We exclude the objects with lower confidences, $conf_{object}$, from the list, to reduce the number of potential objects-in-action and, as a consequence, the number of probed objects and candidate functional units. To illustrate, assume that the filtered list of objects seen in the sequence or probed objects are *egg*, *bowl*, and *fork* and the ground truth functional unit associated with the sequence that includes the motion node *mix* with the objects *bowl*, *egg*, and *fork* as input nodes and *egg*, and *fork* as output nodes. Individually probing functional units in FOON using the list of objects produces a list of candidate functional units that contain those objects. Table I shows some of the 674 candidate functional units that contain the objects-in-action for this specific example. Any other functional unit that is not identified does not contain the objects.

Each probed functional unit from FOON contains object nodes that may or may not have been observed in the current video sequence (associated with an action). The last column of Table I depicts the overlap between the objects included in a probed functional unit with the identified objects in the video sequence. The probed functional units with an overlap value less than a specific threshold are excluded. Confidence values for the remaining functional units are computed, and those

TABLE I
RESULTS OF PROBING OBJECTS IN FOON BASED ON THE EXAMPLE IN SECTION VI-B. THE OBJECTS-IN-ACTION ARE SHOWN IN BOLD.

	Input Nodes	Motion	Output Nodes	Overlap
1	mixer, **bowl**	mix	mixer, **bowl**	0.5
2	**fork**, **egg**, cup	stir	**fork**, **egg**, cup	0.67
:	:	:	:	:
674	**bowl**, pan, pasta	pour	pan	0.25

with the highest confidence values are selected. The selected functional units are the most likely to be associated with the ongoing action.

VII. Experiments and Results

In our experiments, we used the annotated videos used for the creation of the universal FOON in [1] and the videos from the MPII Cooking Activities Dataset, summing to a total of 338 videos [61] [1]. At the time of writing this paper, the universal FOON consisted of data from 338 instructional videos and a total of 3102 functional units. This includes a subset of instructional videos from YouTube and videos from the MPII Cooking Activities Dataset [61]. For the current experiments, we also manually labeled some of the video sequences in FOON with object bounding boxes and their categories.

We used 11 of the 338 cooking videos as our test dataset, which included an overall amount of 55 functional units. We performed our tests in 11 iterations in a leave-one-out manner: in each iteration, one video was entirely left out, and the remainder of the videos were used to create a FOON. The lack of training data for object recognition, lack of labeled ground truth data for the videos, and a shortage of instances of each kind of functional unit in the dataset forced us to only use 11 videos for the experiments.

For testing the pipeline, we conducted three different experiments based on both manually- and automatically-labeled objects. 1) comparing functional unit recognition using only FOON look-up with functional unit recognition using the fusion of FOON and motion recognition, 2) video understanding for functional unit recognition with and without FOON, and 3) task inference or recipe classification.

A. Object Overlap Metric

The overlap between a candidate functional unit and its corresponding ground truth functional unit was used to evaluate the results. This overlap metric was calculated for each action in the video separately. The metric used was fairly simple: if the motion node of the candidate functional unit was equivalent to the motion node of the ground truth functional unit, the overlap between their object nodes was counted. Consequently, precision and recall were computed using the object overlap. *Precision* was measured as overlap divided by the number of object nodes in the candidate functional unit, and *recall* was measured as overlap divided by the number of object nodes in the ground truth functional unit. If the motion

[1]The videos and graphs of FOON are available at: http://www.foonets.com

nodes were different, precision and recall were assumed to be 0. Figure 6 illustrates how precision and recall were calculated.

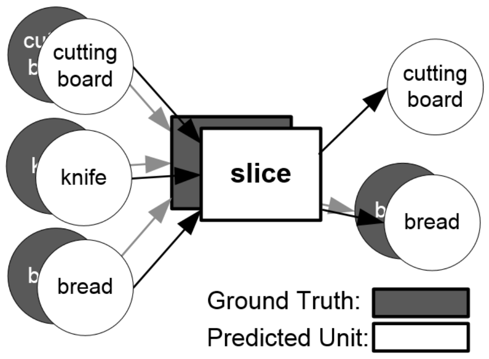


Fig. 6. Illustration of how overlap is measured. In this example, overlap is equal to 4 nodes, precision is 100 percent and recall is 80 percent. If the ground truth motion node was anything but *slice*, precision and recall would be 0 percent.

B. Functional Unit Recognition

We used the time stamp labels in the universal FOON to split the videos in the dataset into its comprising actions. For example, a video demonstrating a cook making scrambled eggs was split into several atomic actions such as cracking eggs, pouring eggs into a bowl, and mixing eggs with a whisk.

1) Functional Unit Recognition using FOON: Each action sequence in a video was fed into the algorithm that identified the functional unit best fit for that action based on the metrics discussed in Section V. In each iteration, we used a single video for evaluation and the other 337 videos to create an iteration-specific FOON. Functional units corresponding to an action sequence in a video were identified by processing the iteration-specific FOON. After identifying functional units, precision and recall were computed as defined in Section VII-A for all candidate functional units for the top 10 results, as shown in Figure 7.

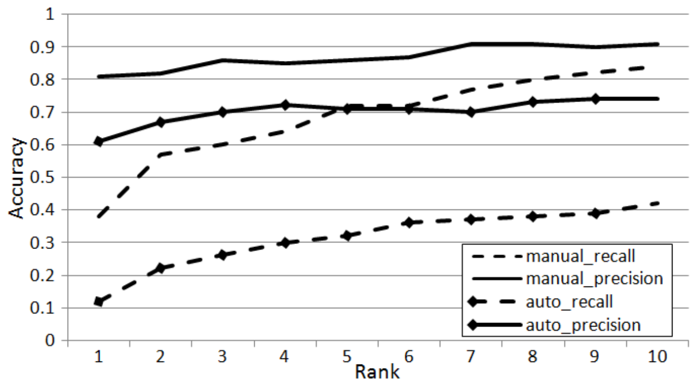


Fig. 7. Precision and recall, as observed in manual and automatic object recognition for top 10.

In Figure 7, the horizontal axis represents the number of best functional units analyzed for precision and recall calculation. The solid curves show precision, and the dashed curves show recall calculated on 55 functional units for both manually- and automatically-labeled objects. Figure 7 shows that the algorithm can potentially improve with additional procedures. We can also see that precision in Figure 7 is always higher than 80 percent, showing that our algorithm sometimes missed the objects in the video; however, when it assumed an object was being used in a functional unit, it usually identified the functional unit correctly. In Figure 8 snapshots of three sequences of a cooking video are depicted with their predicted functional units. In this example, the correct functional unit is always included in the top three identified functional units.

2) Functional Unit Recognition with Motion Recognition and FOON: We fused motion recognition with FOON look-up to improve the recognition procedure. We created motion classes by selecting the nine most frequent motion types from the FOON motion nodes (e.g., *pour*, *pick+place*, and *cook*) [1]. To accommodate the other types of motions not included in the nine most frequent motions, we designed a class labeled as the "other" class. We extracted optical flow features from each sequence in the video, applied the CNN+LSTM network on RGB and optical flow sequences of each event, and performed an averaging of the outputs from the two networks. The architecture returned 10 values representing confidences for the 10 classes. The motion confidence values were used in the computation of confidences for the candidate functional units. Table II shows the top 1, top 3, top 5, and top 10 accuracy of prediction for functional unit recognition using both FOON and motion recognition.

TABLE II
TOP 1 TO 10 ACCURACY OF PREDICTION FOR FUNCTIONAL UNIT RECOGNITION USING FOON AND MOTION RECOGNITION.

	Using FOON	Using FOON + Motion Recognition
Top 1	56%	64%
Top 3	75%	84%
Top 5	80%	89%
Top 10	89%	98%

The accuracy of prediction for an action was computed by comparing the identified functional units with the ground truth functional units. If the motion node of the identified functional unit was equivalent to the motion node of the ground truth and the overlap of object nodes was higher than 80 percent, we determined the prediction to be correct. We counted the number of correct predictions over all functional units in the test set and calculated the accuracy. In some cases, the motion node of the ground truth varied in text with the motion node of the identified functional unit, while having the same interpretation (e.g., *whip* vs. *stir* or *slice* vs. *cut*). These cases of motion nodes were considered equivalent.

As shown in Table II, the accuracy of functional unit recognition when motion recognition was combined with FOON look-up was higher than functional unit recognition without motion recognition. This shows adding automatic motion recognition to the pipeline improves the motion node recognition and leads to better identification of functional units. The deep network guesses the motion node in only 47 percent of the cases. The complexities of the videos, such as background variations, different camera views, and moving

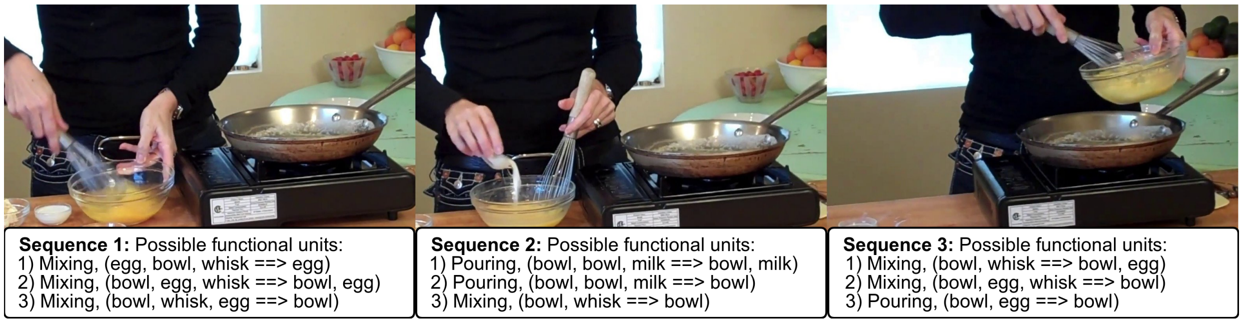

Sequence 1: Possible functional units:
1) Mixing, (egg, bowl, whisk ==> egg)
2) Mixing, (bowl, egg, whisk ==> bowl, egg)
3) Mixing, (bowl, whisk, egg ==> bowl)
Sequence 2: Possible functional units:
1) Pouring, (bowl, bowl, milk ==> bowl, milk)
2) Pouring, (bowl, bowl, milk ==> bowl)
3) Mixing, (bowl, whisk ==> bowl)
Sequence 3: Possible functional units:
1) Mixing, (bowl, whisk ==> bowl, egg)
2) Mixing, (bowl, egg, whisk ==> bowl)
3) Pouring, (bowl, egg ==> bowl)

Fig. 8. Example of functional unit recognition using labeled and manually-split sequences for scrambled egg recipe.

cameras, prevented it from producing the desired accuracy. In experiments, we set α in Equation 4 to less than 0.2, so the results from the neural network would not adversely influence the final results.

3) Analysis: To see the effect of each part of the pipeline on the results, we looked deeper into each part. The automatic motion recognition by itself achieves 67 percent accuracy, whereas the functional unit recognition without motion recognition achieves 61 percent accuracy (top 2). There are two differences in these two evaluations that make them incomparable. First, for automatic motion recognition, the number of classes of motion was generalized and reduced to 10 classes, whereas for functional unit recognition, there were more than 50 types of motion nodes. Second, functional unit recognition identifies the action with a focus on both the objects and the motion occurring, whereas the aim of motion recognition is to recognize the motion class in an action. Although not comparable, motion recognition is a good feature to combine with FOON for optimal functional unit recognition.

We calculated the overlap between objects-in-action and the identified functional units as 84 percent. This shows that although the majority of objects were identified correctly, the accuracy of functional unit recognition was lower than expected due to mistakes in identifying the motion nodes.

In another experiment, we applied the pipeline combined with motion recognition for automatically recognized objects and report its top 10 results in Figure 7. Although object recognition is an important stage of the pipeline that can be improved, we do not address it further, as that is not our specific goal in this paper. Snapshots of various sequences with their ground truth representation and identified functional units are depicted in Figure 9.

C. Video Understanding

The pipeline was evaluated based on the extent it understands a video using the overlap metric. Precision and recall were calculated for both object and motion nodes for all actions of each video individually, and the average precision and recall was calculated for all videos over the top 10 results. Figure 10 shows the calculated results.

The results show that the pipeline is capable of perceiving an understanding of the video, especially when the top 5 results are used. The lower values for recall may be due to errors made in identifying objects-in-action. We calculated the F-Score metric using recall and precision, as discussed in [62].

The video understanding F-Score was calculated for the pipeline in two instances: 1) when FOON was used, and 2) when FOON was not used; the results are depicted in Figure 11. When using FOON, we calculated the F-Score by using the overlap metric for ground truth and identified functional units. When not using FOON, we calculated the overlap metric between the highest-ranked objects and the objects in the ground truth functional unit, and we calculated the overlap between the highest-ranked motion classes with the motion nodes in the ground truth. The sum of these two overlaps was used to calculate the precision and recall and F-Score. Using FOON achieved higher F-Scores than not using FOON, as object and motion nodes in a video are perceived much better when using FOON as reference.

D. Task Inference (Recipe Classification)

We deployed our algorithm for recipe classification of unseen cooking videos. We used eight videos, including one salad recipe, two omelette recipes, two bread recipes, one cake recipe, one noodle recipe, and one sandwich recipe for the test. We classified all the recipes in FOON into 13 classes of recipesCake, Pizza, Bread, Omelette, Soup, Barbecue, Sandwich, Smoothies, Pasta, Coffee/Tea, Salad, Mashed potatoes, and Other.

Task inference was performed after all functional units in a video were identified. All the identified objects-in-action used in the video and the identified functional units equally contributed to the task inference stage. To classify a video to a recipe, clusters of recipes were created using all videos in the train set. The similarity distance between the current video and every (recipe) cluster was calculated, and the closest cluster was selected as the recipe associated with the video. To calculate the similarity distance between the current video and a cluster, the similarity of the video with each of the videos in the cluster was calculated and was averaged. The similarity distance between a video and a recipe was computed as the similarity of functional units in the video with the similarity of functional units in the recipe aggregated with the similarity of the used objects in the video with the similarity of the object nodes in the recipe. In the similarity comparison, we did not check the order of functional units. The recipe class with the highest similarity was assigned to the video. Figure 12 shows

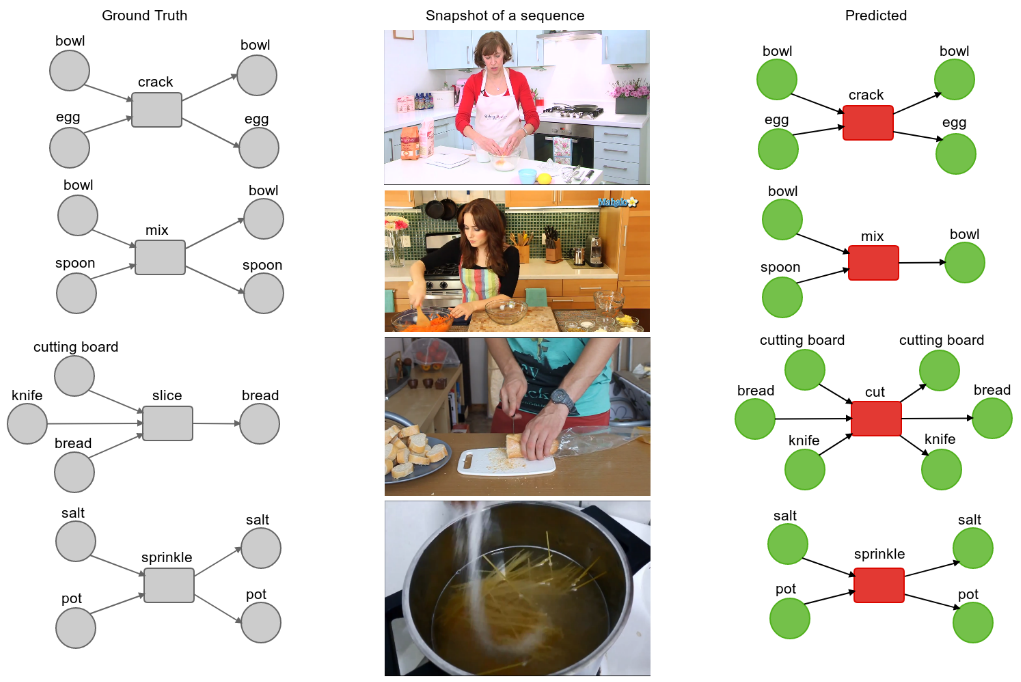


Fig. 9. Snapshots of events, their ground truth functional unit representation and predicted functional unit.

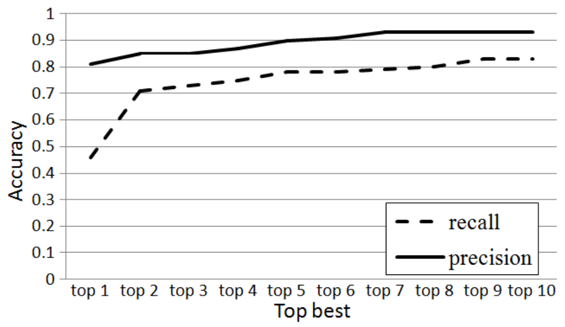


Fig. 10. Graph showing results of overlap metrics for video understanding for top 10.

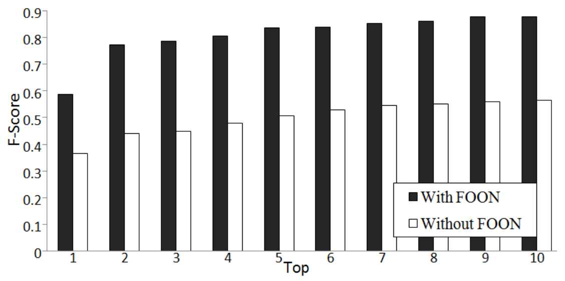


Fig. 11. Graph showing calculated F-Scores for video understanding with and without FOON.

the identified functional units of a video demonstrating a cook making noodles.

The top 2 results of recipe classification are shown in Table III. The recipe classification algorithm returned the predicted class names based on their confidence scores. If the class name with the highest confidence is equivalent to the ground truth class name, the classification is assumed as correct.

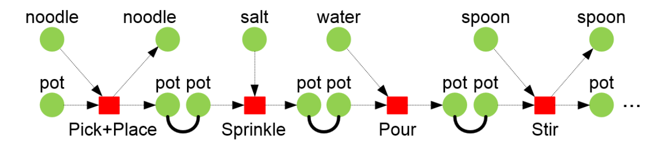


Fig. 12. Illustration of identified functional units for noodle-cooking video (identified as noodle by proposed pipeline)

TABLE III
RECIPE CLASSIFICATION RESULTS.

Used Procedure	Top 1	Top 2
Manually labeled Objects	37.5%	100%
Automatically labeled Objects	25%	75%

As shown in Table III, the algorithm using FOON can approximately guess what recipe is being cooked in the video, assuming that all objects in the video sequence are identified correctly. The motion of the objects can also insinuate the type of recipe activity that is occurring.

E. Discussion

Several studies have worked on activity recognition using either knowledge bases or other methods, but they represent a video with a sentence or a label for the activity. Our work outputs sub-graphs representing short activities for each part of the video. As such, our work is incomparable to other work. We analyzed our work through the overlap metric and compared two approaches for video understandingpipeline using FOON and pipeline not using FOON. It is clear that some methods in the literature can be substituted with the method we use to integrate with FOON, but our current focus is to prove that FOON is a powerful knowledge representation that can understand video and would be able to semi-automatically build itself in the future.

Although the proposed framework uses information from multiple views throughout the video, it applies information from only one camera at each specific time. However, due to the importance of simultaneous multi-view applications, we discuss briefly a few ways that the framework can be integrated into a multi-view system. The proposed framework can be applied individually to multiple videos in a multi-view system. Individual predictions can be gathered from multiple deployments of the framework. The predictions can be further combined to reach to a final prediction of the actions and activity in the video. We can also combine the proposed framework from a multiple view aspect at the confidence level. Confidences of objects can be extracted at each view and fused to reach a final confidence for the objects. The framework can further run as proposed.

The goal of the proposed framework is to identify the actions and tasks in a video. The framework can be used as the vision system of a robot chef or in any robotic system that deploys and manipulates utensils, such as a robot carpenter, robot waiter, etc.

VIII. Conclusion and Future Work

The main objective of this paper was video understanding with the help of the FOON knowledge representation. We proposed a pipeline for video understanding using the functional object-oriented network (FOON) and deep neural networks and make use of low-level image features together with deep networks to identify objects of interest. Using objects of interest (objects-in-action) and deep motion understanding, we associate the actions in a video with the correct functional units in the knowledge representation (FOON). We demonstrated that using FOON significantly improves the performance of video understanding in comparison to not using FOON.

Our current pipeline is a big step towards automatically extending the knowledge representation graph, which presents a significant improvement to network applications, such as robots solving manipulation problems given a target goal. In future work, we would like to explore other methods of identifying objects-in-action, incorporate object recognition confidences to handle misidentified objects, utilize states of objects [63], and incorporate history of events for inference using FOON. We are also working on generalizing the knowledge contained within a FOON to achieve more generic video inferences from an expanded version of FOON [2].

References

[1] David Paulius, Yongqiang Huang, Roger Milton, William D. Buchanan, Jeanine Sam, and Yu Sun. Functional object-oriented network for manipulation learning. In *IROS*, pages 2655–2662. IEEE, 2016.

[2] D. Paulius, A. B. Jelodar, and Y. Sun. Functional object-oriented network: Construction & expansion. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–7, May 2018.

[3] David Paulius and Yu Sun. A survey of knowledge representation and retrieval for learning in service robotics. *arXiv preprint arXiv:1807.02192*, 2018.

[4] Christiane Fellbaum, editor. *WordNet: an electronic lexical database*. MIT Press, 1998.

[5] K.K. Schuler. *Verbnet: A Broad-coverage, Comprehensive Verb Lexicon*. 2005.

[6] Collin F. Baker, Charles J. Fillmore, and John B. Lowe. The berkeley framenet project. In *Proceedings of the 17th International Conference on Computational Linguistics - Volume 1*, COLING '98, pages 86–90, Stroudsburg, PA, USA, 1998. Association for Computational Linguistics.

[7] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka, Jr., and Tom M. Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, AAAI'10, pages 1306–1313. AAAI Press, 2010.

[8] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 62(1):107–136, Feb 2006.

[9] Qi Wu, Peng Wang, Chunhua Shen, Anton van den Hengel, and Anthony R. Dick. Ask me anything: Free-form visual question answering based on knowledge from external sources. *CoRR*, abs/1511.06973, 2015.

[10] W. Min, B. K. Bao, S. Mei, Y. Zhu, Y. Rui, and S. Jiang. You are what you eat: Exploring rich recipe information for cross-region food analysis. *IEEE Transactions on Multimedia*, 20(4):950–964, April 2018.

[11] Christopher Town. Ontological inference for image and video analysis. *Machine Vision and Applications*, 17(2):94, Mar 2006.

[12] Nicolas Eric Maillot and Monique Thonnat. Ontology based complex object recognition. *Image Vision Comput.*, 26(1):102–113, January 2008.

[13] Bernd Neumann and Ralf Möller. On scene interpretation with description logics. *Image Vision Comput.*, 26(1):82–101, January 2008.

[14] X. Chen, A. Shrivastava, and A. Gupta. Neil: Extracting visual knowledge from web data. In *2013 IEEE International Conference on Computer Vision*, pages 1409–1416, Dec 2013.

[15] Feng Niu, Ce Zhang, Christopher Ré, and Jude Shavlik. Elementary: Large-scale knowledge-base construction via machine learning and statistical inference. *Int. J. Semant. Web Inf. Syst.*, 8(3):42–73, July 2012.

[16] Yuke Zhu, Alireza Fathi, and Li Fei-Fei. Reasoning about object affordances in a knowledge base representation. In *ECCV 2014*, pages 408–424, Cham, 2014. Springer International Publishing.

[17] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123(1):32–73, May 2017.

[18] L. Herranz, S. Jiang, and R. Xu. Modeling restaurant context for food recognition. *IEEE Transactions on Multimedia*, 19(2):430–440, Feb 2017.

[19] Pei-Yu Chi, Jen hao Chen, Hao-Hua Chu, and Jin-Ling Lo. Enabling calorie-aware cooking in a smart kitchen. In *PERSUASIVE*, volume 5033 of *Lecture Notes in Computer Science*, pages 116–127. Springer, 2008.

[20] Xiatian Zhu, Chen Change Loy, and Shaogang Gong. Learning from multiple sources for video summarisation. *International Journal of Computer Vision*, 117(3):247–268, 2016.

[21] Cornelia Fermller, Fang Wang, Yezhou Yang, Konstantinos Zampogiannis, Yi Zhang, Francisco Barranco, and Michael Pfeiffer. Prediction of manipulation actions. *CoRR*, abs/1610.00759, 2016.

[22] Patrick Perez and Ivan Laptev. Retrieving actions in movies. In *ICCV*, pages 1–8. IEEE Computer Society, 2007.

[23] Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. *CoRR*, abs/1411.4389, 2014.

[24] Michele Merler, Bert Huang, Lexing Xie, Gang Hua, and Apostol Natsev. Semantic model vectors for complex video event recognition. *IEEE Trans. Multimedia*, 14(1):88–101, 2012.

[25] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *ICCV*, pages 3551–3558. IEEE Computer Society, 2013.

[26] Guilhem Chron, Ivan Laptev, and Cordelia Schmid. P-cnn: Pose-based cnn features for action recognition. *CoRR*, abs/1506.03607, 2015.

[27] Bingbing Ni, Pierre Moulin, and Shuicheng Yan. Pose adaptive motion feature pooling for human action analysis. *International Journal of Computer Vision*, 111(2):229–248, 2015.

[28] Cuiwei Liu, Xinxiao Wu, and Yunde Jia. A hierarchical video description for complex activity understanding. *International Journal of Computer Vision*, 118(2):240–255, 2016.

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

[30] Yuki Matsumura, Atsushi Hashimoto, Shinsuke Mori, Masayuki Mukunoki, and Michihiko Minoh. Clustering scenes in cooking video guided by object access. In *ICME Workshops*, pages 1–6. IEEE Computer Society, 2015.

[31] Krishan Kumar, Deepti D. Shrimankar, and Navjot Singh. Eratosthenes sieve based key-frame extraction technique for event summarization in videos. *Multimedia Tools and Applications*, 77:7383–7404, 2017.

[32] Krishan Kumar, Anurag Kumar, and Ayush Bahuguna. D-cad: Deep and crowded anomaly detection. In *ICCCT-2017*, 2017.

[33] Ashesh Jain, Amir Roshan Zamir, Silvio Savarese, and Ashutosh Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. *CoRR*, abs/1511.05298, 2015.

[34] M. Hasan and A. K. Roy-Chowdhury. A continuous learning framework for activity recognition using deep hybrid feature models. *IEEE Transactions on Multimedia*, 17(11):1909–1922, Nov 2015.

[35] K. Kumar and D. D. Shrimankar. F-des: Fast and deep event summarization. *IEEE Transactions on Multimedia*, 20(2):323–334, Feb 2018.

[36] A. Gupta and L. S. Davis. Objects in action: An approach for combining action understanding and object perception. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2007.

[37] A. Fathi and J. M. Rehg. Modeling actions through state changes. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2579–2586, June 2013.

[38] Jianxin Wu, Adebola Osuntogun, Tanzeem Choudhury, Matthai Philipose, and James M. Rehg. A scalable approach to activity recognition based on object use. In *ICCV*, pages 1–8. IEEE Computer Society, 2007.

[39] Jingjing Chen and Chong-Wah Ngo. Deep-based ingredient recognition for cooking recipe retrieval. In *ACM Multimedia*, pages 32–41. ACM, 2016.

[40] Yang Zhou, Bingbing Ni, Richang Hong, Meng Wang, and Qi Tian. Interaction part mining: A mid-level approach for fine-grained action recognition. In *CVPR*, pages 3323–3331. IEEE Computer Society, 2015.

[41] Xiaoyang Wang and Qiang Ji. Hierarchical context modeling for video event recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(9):1770–1782, 2017.

[42] Krishan Kumar and Deepti D. Shrimankar. Deep event learning boost-up approach: Delta. 03 2018.

[43] K. Kumar, D. D. Shrimankar, and N. Singh. Event bagging: A novel event summarization approach in multiview surveillance videos. In *2017 International Conference on Innovations in Electronics, Signal Processing and Communication (IESC)*, pages 106–111, April 2017.

[44] Xishan Zhang, Yang Yang, Yongdong Zhang, Huanbo Luan, Jintao Li, Hanwang Zhang, and Tat-Seng Chua. Enhancing video event recognition using automatically constructed semantic-visual knowledge base. 17:1–1, 09 2015.

[45] C. F. Crispim-Junior, V. Buso, K. Avgerinakis, G. Meditskos, A. Briassouli, J. Benois-Pineau, I. Y. Kompatsiaris, and F. Bremond. Semantic event fusion of different visual modality concepts for activity recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(8):1598–1611, Aug 2016.

[46] X. Chang, Z. Ma, Y. Yang, Z. Zeng, and A. G. Hauptmann. Bi-level semantic representation analysis for multimedia event detection. *IEEE Transactions on Cybernetics*, 47(5):1180–1197, May 2017.

[47] M. Mazloom, X. Li, and C. G. M. Snoek. Tagbook: A semantic video representation without supervision for event detection. *IEEE Transactions on Multimedia*, 18(7):1378–1388, July 2016.

[48] Fillipe D. M. de Souza, Sudeep Sarkar, Anuj Srivastava, and Jingyong Su. Temporally coherent interpretations for long videos using pattern theory. In *CVPR*, pages 1229–1237. IEEE Computer Society, 2015.

[49] Fillipe D. M. de Souza, Sudeep Sarkar, Anuj Srivastava, and Jingyong Su. Spatially coherent interpretations of videos using pattern theory. *International Journal of Computer Vision*, 121(1):5–25, 2017.

[50] Yu Sun and Shaogang Ren. Human-object-object-interaction affordance. In *WORV*, pages 1–6. IEEE Computer Society, 2013.

[51] Yu Sun, Shaogang Ren, and Yun Lin. Object-object interaction affordance learning. *Robotics and Autonomous Systems*, 62(4):487–496, 2014.

[52] Yu Sun, , and Yun Lin. Modeling paired objects and their interaction. *New Development in Robot Vision*, 23:73–87, 2015.

[53] A. Gupta, A. Kembhavi, and L. S. Davis. Observing human-object interactions: Using spatial and functional compatibility for recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(10):1775–1789, Oct 2009.

[54] B. Yao and L. Fei-Fei. Modeling mutual context of object and human pose in human-object interaction activities. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 17–24, June 2010.

[55] Dong Han, Liefeng Bo, and Cristian Sminchisescu. Selection and context for action recognition. *2009 IEEE 12th International Conference on Computer Vision*, pages 1933–1940, 2009.

[56] M. Newman. *Networks: An Introduction*. OUP Oxford, 2010.

[57] C. Adam Petri and W. Reisig. Petri net. *Scholarpedia*, 3(4):6477, 2008. revision 91646.

[58] Gibson J J. *The theory of affordances, in Perceiving, Acting, and Knowing. Towards an Ecological Psychology.* Number eds Shaw R., Bransford J. Hoboken, NJ: John Wiley & Sons Inc., 1977.

[59] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, pages 91–99, 2015.

[60] T Brox, A Bruhn, N Papenberg, and J Weickert. High accuracy optical flow estimation based on a theory for warping. In *European Conf. on Computer Vision*, volume 3024, pages 25–36, Prague, Czech Republic, 2004.

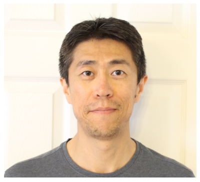
[61] Marcus Rohrbach, Sikandar Amin, Mykhaylo Andriluka, and Bernt Schiele. A database for fine grained activity detection of cooking activities. In *CVPR*, pages 1194–1201. IEEE Computer Society, 2012.

[62] Tom Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861 – 874, 2006. ROC Analysis in Pattern Recognition.

[63] A. B. Jelodar, M. S. Salekin, and Y. Sun. Identifying object states in cooking-related images. *arXiv preprint arXiv:1805.06956*, May 2018.

Ahmad Babaeian Jelodar received the BSc degree in software engineering, in 2008, and MSc degree in computer engineering, in 2011, from the Amirkabir University of Technology, Tehran, Iran. He has been a PhD student in the Department of Computer Science and Engineering at the University of South Florida, Tampa, FL, USA, since Spring 2016. His research interests include computer vision, deep learning, image processing, and machine learning.

David Paulius received his B.S. degree in Computer Science from the University of the Virgin Islands, St. Thomas, VI, USA, in 2014. Since Fall 2014, he has been a Ph.D. student in the Department of Computer Science and Engineering at the University of South Florida and is presently working as a member of the Robot Perception and Action Lab (RPAL). His research interests include knowledge representations, artificial intelligence and graph theory, as pertaining to service robotics.



Yu Sun is an Associate Professor and Associate Chair of Graduate Affairs in the Department of Computer Science and Engineering at the University of South Florida. He is an Associate Editor of the IEEE Transactions on Robotics. He was a Visiting Associate Professor at Stanford University and a Postdoctoral Associate at Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA. He received his Ph.D. degree in Computer Science from the University of Utah, and his B.S. and M.S. degrees in Electrical Engineering from Dalian University of Technology. His research interests include robotics, intelligent systems, deep learning, computer vision, virtual reality, and medical applications.