Abstraction Hierarchy and Self Annotation Update for Fine Grained Activity Recognition

Song Cao, Kan Chen and Ram Nevatia
University of Southern California, Institute for Robotics and Intelligent Systems
Los Angeles, CA 90089, USA
songcao@usc.edu

Abstract

Fine-grained activity recognition focuses recognition on sub-ordinate levels. This task is made difficult due to low inter-class variability and high intra-class variability caused by human motion and objects. We propose that recognition of such activities can be significantly improved by grouping and decomposing them into a hierarchy of multiple abstraction layers; we introduce a Hierarchical Activity Network (HAN). Recognition in HAN is guided by classifiers operating at multiple levels; furthermore, descriptions of different levels of abstraction are also generated from HAN, which may be useful for different tasks. We show significant improvements in accuracy of recognition compared to earlier methods. Besides, annotation for fine grained activity is challenging, and inaccurate annotation influences classification performance. We explore an automatic solution for improving the classification results while auto enhancing the annotation quality.

1. Introduction

Activity recognition has been a popular topic of research in computer vision. The range of activities can vary broadly in terms of their specificity. In this paper, we focus on fine-grained activities that take place in a fixed environment, which has potential applications in intelligent home, elder people care and for daily living assistance. Several activities of daily life datasets have been introduced. A recent one is given in [23] which contains 65 different but similar kitchen activities. The characteristic that these activities take place in a fixed (and possibly known) environment can be helpful but also prevents environmental context from being a key distinguishing feature. Furthermore, the activities of interest can be very similar, e.g. recognizing *cutting stripes* vs *cutting slices* in kitchen tasks. Considerable progress has been made in classifying such videos, [23][24] but accuracy remains low for many of the activities.

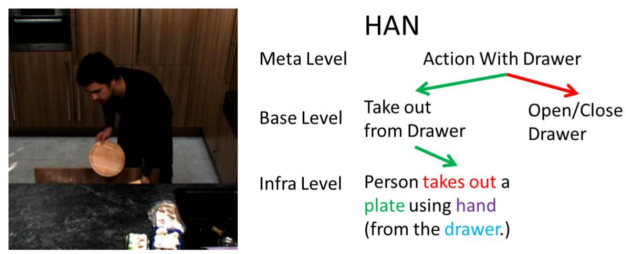

Figure 1. Three levels of representation in HAN for an example video.

There have been many approaches to the problem of activity recognition, many of which are hard to apply to the stated problem of fine-grained activity recognition. The human locomotion in this problem is of low inter-class variability so not very discriminative. Human pose is of obvious importance but the current pose estimation algorithms are not robust enough to provide discriminative pose configurations [23]. Hence, we resort to use of low-level features which has proved effective for many tasks. However, the distinctions between various activities in the fine-grained task are small so similar activities may be difficult to discriminate. We propose that recognition of such activities can be significantly improved by grouping and decomposing them into a hierarchy of multiple abstraction layers. The recognition can then be guided by classifiers operating at multiple levels; furthermore, descriptions of varying abstractness are also generated which may be useful for different tasks. Use of such a hierarchy has been successfully explored for image classification [5] problems but not for activities, to the best of our knowledge.

We describe an abstraction hierarchy model named Hierarchy Activity Network (HAN), which includes Meta level(s), Base level and Infra level(s), as shown in Figure 1. Base level is the level at which annotations are provided in the training data. Meta levels group some of the base level activities into one, resulting in more abstraction activities. Infra levels decompose a set of base level activities based on Infra distinctions. In the example, and in our experiments, we use only one meta and one Infra level but the technique easily extends to the multiple levels. Note that this hierarchy is based on abstraction levels rather than the more common hierarchy derived from temporal decompositions as in, for example, [22][28].

Construction of Meta and Infra levels from Base level is by unsupervised clustering based on similarity using SVM classifiers. As we lack annotations at the Meta and Infra levels, the nodes may not have semantic names. For the Meta levels, we can create a name by composition of terms in the Base level. For Infra levels, it is difficult to provide a semantic name in unsupervised way, but we can relate the activities to attributes to construct meaningful phrases.

Independent classifiers are trained for each node in the hierarchy. We use the score of a path in the network to compute the confidence of labels for a video. We find that this not only improves recognition accuracy but also gives us a better attribute prediction and description for the video. With the annotated attributes provided by [24], each Infra level node is associated with an attribute distribution learned from the training data. To generate Infra level descriptions, we use pre-trained attribute detectors in the video level. Thus, the attribute prediction in HAN for video is not only based on the binary attribute classifiers, but also the Infra level node priors. In activity classification task, we do not use attribute annotations.

We find that annotation for fine grained activity recognition has two limitations. First, the activity labels can be confusing; for example, it is hard to tell the difference between *put in pan* or *pour* in annotated video clips. Also, people have different opinions about when an activity starts, and when it ends. For example, it is hard to determine whether the activity *open the fridge* includes *walking towards the fridge*. We find that a better annotated video will help improve the classification performance. So we propose an automatic way to improve the annotation. We first prepare a video feature dictionary based on previous annotations. Then we evaluate the feature dictionary using trained model, replace the original feature with a candidate feature with highest confidence score, into a new training dataset. Then we iterate this process to get the final result.

In summary, our **contributions** are:

(1) Modeling of the mutual and hierarchy information within fine grained activities.

(2) HAN's hierarchy is of abstraction type which is different from and complementary to the more common temporal hierarchy (from atomic action to activity to events).

(3) Demonstration that activity hierarchy helps improve fine grained activity recognition and attribute predictions.

(4) A self updated annotation method improves both the classification performance and annotation quality.

The paper is organized as follows. Section 2 discusses related work. Then, we introduce HAN in Section 3, including construction and inference using HAN. We present experimental results in Section 4 and give our conclusions in Section 5.

2. Related Work

A considerable number of approaches have been proposed in activity recognition problem. In complex datasets such as UCF101 [26] and HMDB51 [12], low and middle level features [29, 27, 11] are used commonly. Among them, the bag of spatial-temporal interest points [15] representation, patches [10], attributes [18], dynamic poselet[30], have been widely adopted for human action recognition. It can be combined with various models such as discriminative classifiers [16][21], unsupervised generative models [32], and semi-latent topic models [31]. Besides this holistic representation, researchers also work on integrating feature sequence arrangement and temporal ordering information. Some researchers try to construct plausible temporal structures [9] for different actions agents, but they ignore the temporal composition within the movements of a single subject under the framework of holistic representation. On the other side, temporal context based discriminative models can be applied in coarse grained activities' classification tasks [4]. Besides, contextual information, such as background scene context [21] and object interactions [9][33], can also improve the performance of activity recognition. Our paper uses an abstraction hierarchy which incorporates contextual information in both the training and testing stages.

Fine grained image or activity classification poses further challenges. One is that the training data is limited due to the difficulty of acquiring fine grained annotations. Besides, unlike categorization, the differences between fine grained classes may be subtle and only some key features may be useful. One approach to address these problems is to apply specialized domain knowledge [13], which uses botany information to help classify leaves. However, this requires developers to have a deep understanding of the specific field. Another method is to use crowdsourcing in the loop [7][3], asking humans to either label or propose parts and attributes [2][1], which achieves promising results. The difficulty lies in designing annotation tasks effectively: two ways are proposed to solve this problem. One is ask humans to label pre-defined parts and attributes [7][3], which requires careful design; the other is to use open-ended tasks [19]. But the quality in this method may be hard to control and descriptors may be highly varied. The recent improvement for ImageNet is [5], which introduces Hierarchy and Exclusion (HEX) graphs to captures semantic relations between any two labels applied to the same object: mutual exclusion, overlap and subsumption. Similarly, [8] propose a solution when we cannot find an accurate prediction for a pre-trained model, it finds a less specific answer that is also plausible from a pragmatic standpoint.

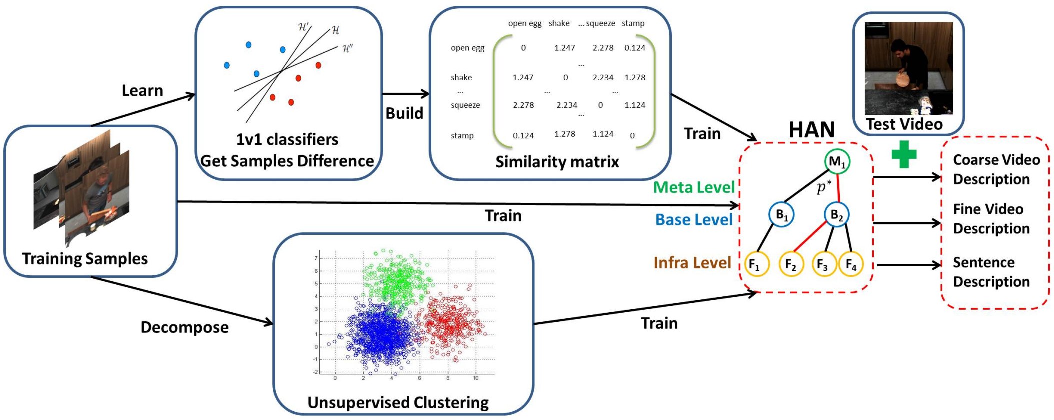

Figure 2. HAN Flow Chart and Representation. We first build Meta and Infra level in an unsupervised way, then integrate these nodes into an abstraction hierarchy model. HAN can be used to classify activities and predict attributes in a given input video.

Recently, [34] shows large improvements on the [23] fine grained activity dataset using pre-detection of objects. They take advantages that most of the objects are stationary, and the video is captured with a fixed camera (strong prior integrated). However, it is difficult to generalize this approach as reliable object detectors which are not always available. In our HAN solution, we never use the pre-detection objects, no extra annotations are needed. [25] discusses about how to find the most discriminative portion of videos, while we focus on auto improving the fine grained activity annotations and achieving better recognition performance.

3. Hierarchy Activity Network

In this section, we give a description of how to model HAN and apply it for activity recognition. First, we define HAN model and activity similarity in detail. Next, we show how to encode the semantic similarity information to construct HAN model in an unsupervised way. In the third part, we show how to apply HAN in fine-grained activity recognition and use HAN to generate description in different hierarchy activity levels. The flow chart of HAN is shown in Figure 2.

3.1. HAN Representation

HAN is a multiple level network which integrates information from not only the Base level, but also Meta (super) level and Infra (sub) level. The design of Meta level and Infra level follows two intuitions in the fine grained activity recognition problem. On one hand, some similar base level activities with limited training samples need a meta classifier to help distinguish from other uncorrelated activities. For example, *Cut Something* helps distinguish finer cutting activities (e.g. cut stripes) from other non-cut activities. On the other hand, for certain activity, there may still exist different sub-actions like *take&put in fridge, spread*. These sub-actions may be hard for one general classifier to recognize. Thus, we need to build Infra level to further select discriminative sub clusters. Here we give the definition for these three levels specifically.

Base Level: This is the level where the main video labels are provided and where the activity recognition tasks are defined.

Meta Level: This is the more abstract level of the base level. It groups some of the activities in base level by combining similar base level activities.

Infra Level: This is the more detailed level below the base level. Each activity can be divided into several Infra level nodes in an unsupervised way. If the dataset provides attribute annotations, each Infra node can own discriminative attributes distribution in this level.

We denote B_i for the nodes in the Base level, M_j for each synthesized node in the Meta level, and F_k for the nodes in the Infra level. The confidence for B_i is denoted as $f(B_i|\phi(\mathbf{x}))$, where $\phi(\mathbf{x})$ denotes input feature for sample $\mathbf{x}$. An example representation of HAN model is shown in Figure 2.

Similarity Measure: HAN automatically generates the Meta and Infra levels. To build such a network, we need a similarity metrics. As no similarity annotation is provided, we choose to use one versus one SVM distance for the similarity measure. We define a *similarity matrix* S on the training dataset. S_{ij} represents the similarity between Activities B_i and B_j.

We use training samples from activity B_i and activity B_j in base level as an example. To get S_{ij}, we train SVM using B_i as positive and B_j as negative. For each labeled B_i or B_j pair, we get a confidence score. We then compute the average for these two groups of samples. Last, we compute their difference from the average score. The distance between activity B_i and B_j is computed by the following formula.

$$S_{ij} = \frac{1}{T}\Big|\sum_k \left(f(B_i|\phi(\mathbf{x}_k)) - f(B_j|\phi(\mathbf{x}_k))\right)\Big| \quad (1)$$

where $\{x_k\}$ denotes the training samples with size T.

3.2. HAN Construction

We first analyze the benefits of building meta level and Infra level using SVM classifier [20] as an example. Then we state the detailed method of constructing HAN model.

Generating the Meta level: If we use hinge loss function for optimization, the primal form of objective function can be written as:

$$\mathcal{L}(\mathbf{w}) = C\sum_n \xi_n + \frac{1}{2} \parallel \mathbf{w} \parallel_2^2 \tag{2}$$

where $C \geq 0$, $\{\xi_n\}$ are slack variables, which also represent the hinge loss function:

$$\ell(\mathbf{w}) = \max(0, 1 - y[\mathbf{w}^\top \phi(\mathbf{x}) + b]) \tag{3}$$

For misclassified samples, the hinge loss is proportional to the difference between estimation probability of true label class i and wrong estimated label class j:

$$\ell(\mathbf{w}, \phi(\mathbf{x}_n)) \propto f(B_j|\phi(\mathbf{x}_n)) - f(B_i|\phi(\mathbf{x}_n)) \tag{4}$$

Summing the hinge loss function for class i, we get:

$$\ell(\mathbf{w})_i = \sum_n \ell(\mathbf{w}, \phi(\mathbf{x}_n^i)) \propto -\sum_j S_{ij} \tag{5}$$

where $\{x_n^i\}$ denote the sample set with true label i.

From Equation 5, we observe that $\mathcal{L}(\mathbf{w})$ in Equation 2 can be reduced by dropping the loss generated from the first t smallest elements in i th column of similarity matrix S. Thus, we can use S to *merge* different similar activities classes into one Meta level node M_k, which is the corresponding node in Meta level. In training stage for certain class B_i, we eliminate similar samples from different classes B_j sharing the same M_k.

After we generate S using Equation 1, we apply a greedy method to select most similar pairs of activities, as shown in Algorithm 1 below.

Algorithm 1 Similar Activity Pair Selection

Input: similarity matrix S

Label all B_i "available"
while there is activity classes available **do**
  Find $(i, j)^* = \arg\min_{i,j} S_{ij}^{\text{available}}$
  Change activity B_i, B_j's labels as "unavailable"
  Add (B_i, B_j) to merge list L
end while
for k th pair in L **do**
  Assign M_k two children node: B_i, B_j
  Set $path(B_i, M_k) = 1, path(B_j, M_k) = 1$
end for
return M_k

With SVM, similar examples are still hard to be separated, while the not similar examples are easy to be separated.

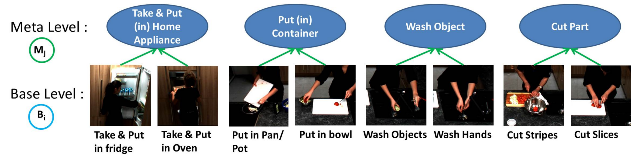

Figure 3. Meta Level Generation Examples.

It is intuitive that the similar categories will still have less difference than the not similar categories. Using the SVM metrics, we find that screw open & screw close, wash objects & wash hands, cut stripes & cut slices belong to the same meta level nodes. Both intuition and experimental results shows the SVM metrics is more reasonable than equal-weight distance metrics.

Naming the Meta Level Nodes: The semantic names for M_k are synthesized from the activity names B_i and B_j. Typically, the activity name of B_i is composed like *Do Object (verb + noun)* or *Do (verb)*. For an input activity name, we first use WordNet to decide which is verb, and noun in the name of B_i automatically. Then, we generate Meta level names using a synthesis rule as $(v_i + n_i) + (v_j + n_j) = p(v_i, v_j) + p(n_i, n_j)$. Here $p(v_i, v_j)$, $p(n_i, n_j)$ mean the least shared parent of the verb and noun. For $p(n_i, n_j)$, we use the WordNet Hierarchy provided by ImageNet [6]. For example, we need to find the least share parent for pan and bowl. A path goes up to top from pan is, pan - container - instrumentality. The path goes up from bowl is bowl - dish - container - instrumentality. Therefore, the least shared parent of pan and bowl is container. For the verb $p(v_i, v_j)$, we temporally use human to name their parent verb due to the challenges in building verbs mutual relations. In this way, we can synthesis the Meta level node names like *put in pan + put in bowl = put in containers*.

Generating the Infra Level: The function of Infra level is to provide more detailed description for each activity class by clustering them, making a Infra sub class. To achieve this goal, we prepare two protocols to constrain the division of fine grained training samples: first, the samples need to be roughly equal distributed; second, the intra-class distance needs to be as small as possible. We use $K-means$ clustering to select the most discriminative feature groups for representing different sub-activities. The two protocols are encoded in constraints of clusters' sizes balancing and minimizing the total distance sum:

$$D(K, \{\mathbf{x}_i\}) = \sum_{k=1}^{K} \left(\frac{1}{N_k} \sum_{n=1}^{N_k} y_{nk} d(\phi(\mathbf{x_n}), C_k) \right) \tag{6}$$

where $d(\phi(\mathbf{x_n}), C_k)$ denotes the distance for each feature $\phi(\mathbf{x}_n)$ to its center C_k. $y_{nk} = 1$ if $\phi(\mathbf{x}_n)$ belongs to kth cluster, otherwise $y_{nk} = 0$.

For building Infra level, we use a loop to find the best Infra level separation with K-means. Our algorithm balances

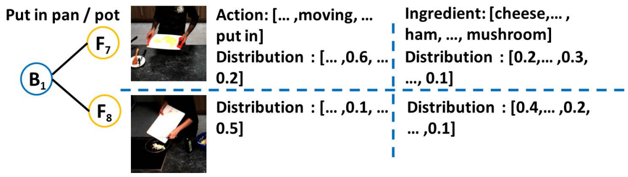

Figure 4. Combining an Infra level Node with Attributes.

the training sample number for each sub class so that possible solutions for unsupervised classification will be limited.. Then, we use SVM to train the Infra level nodes according to the separated data.

Naming the Infra Level Nodes: It is difficult to automatically give the Infra level nodes a semantic name (as the Infra level nodes are generated in an unsupervised way.) Therefore in Infra level, instead, we attach a template of semantic attributes for each input video, using predicted attributes as explained below. In the kitchen dataset, the attribute annotations are given [24] and can be used in training.

Boosting Infra Level with Attributes: HAN can optional integrate activity attributes in a supervised way. This allows the infra nodes to contain attribute priors which can then be combined with attribute detectors for improved descriptions. We generate sentence descriptions by filling in predicted attributes into pre-defined sentence templates. Four attributes are provided: actions (e.g. take out), tools (e.g. hand, knife), ingredients (e.g. apple), and containers (e.g. baking tray, bowl). Given the attribute values, we can compose a sentence: for example, given Action $=$ taking out, tool $=$ hand, ingredient $=$ bottle, containers $=$ fridge, we can generate the sentence : *Person takes out a bottle using hand (from the fridge).*

Summary: To give difference level descriptions, for the Meta level, we use the synthesis name of predicted node M_j. For the Base level, we use name of predicted base node B_i. For the Infra level, we use the predicted attributes to compose a sentence. We introduce how to predict the attributes in next section.

3.3. HAN Inference

Activity Recognition: For each input video x, we calculate a coarse to fine path p in HAN. Every path is composed of a node from the Base level B_i, a node from the Meta level M_j, and a node from the Infra level F_k. It can be written as $p = (B_i, M_j, F_k)$. Given input video x, we define a vector $\psi(p,x)$, which $\psi(p,x) = [\phi(B_i), \phi(M_j), \phi(F_k), \varphi(B_i, M_j), \varphi(B_i, F_k)]$. The $\phi(B_i), \phi(M_j), \phi(F_k)$ are the unary confidences of selecting B_i, M_j, F_k to represent the videos. Specifically, $\phi(B_i)$ is the output of a single binary video classifier. $\varphi(B_i, M_j), \varphi(B_i, F_k)$ are the pairwise term that shows how likely the path is. It is set equally in our implementation. The optimal path p^* is given by maximizing the output of $\omega \cdot \psi(p,x)$:

$$p^\star = \arg\max_p \omega \cdot \psi(p,x) \qquad (7)$$

where the ω is the weight for $\psi(p,x)$, to balance each term. ω is set equally at the initialization, then we search these parameter and choose the weight by cross validation on training dataset.

Attribute Prediction: Within the Infra level, activity attributes are predicted as follows. We first train the video level attribute detectors using middle level features and an SVM [20]. Assuming a base level node B_k is decomposed into Infra nodes F_{ki}, we predict attributes for node B_k by the following formula:

$$a^\star = \arg\max_{a_i} \phi(a_i) \sum p(a_i|F_{kj})\phi(F_{kj}) \qquad (8)$$

where $\phi(a_i)$ is the confidence for the trained attribute detector, $\phi(F_{kj})$ is the confidence for the Infra node F_{kj}. $p(a_i|F_{kj})$ is the prior where $p(a_i|F_{kj}) = \frac{N_{a_i,F_{kj}}}{N_{F_{kj}}}$. $N_{a_i,F_{kj}}$ is the number of samples that belong to node F_{kj} with the attribute a_i, while $N_{F_{kj}}$ is the total number of samples that belongs to node F_{kj}. Note that this combines the prior expectations of attributes for a specific node with the actual detector outputs.

3.4. Auto Annotation Updating

The annotation for fine grained activity needs refinement. We propose an auto annotation updating method. For the training dataset T, we break them into N subset T_i. For each T_i, we leave most of the original training data (where $T_j, j \neq i$) in training data set T_1, and we use the T_i for improvement in set T_2. For each example in T_2, we prepare a feature expansion dictionary D, by re cropping the sample from T_2. For example, if we have a sample X_i from T_2, we generate the X_{ij} by modifying the boundary of sample X_i, put all X_{ij} into our dictionary. Then, we train a detector using the samples from T_1, and test them on T_2. We sort the results and find the best cropped video clips in D, then update the original X_i in T_2. Using a similar strategy, after we updates the whole training dataset, we train a new fine grained activity detector. The specific algorithm is shown in Algorithm 2.

3.5. Discussion on Hierarchy

Building hierarchy has been used in image classification problem. [17] builds the tree from the top down: the leaf level is the recognition target. Our idea is to extend the target layer, by combining nodes into meta level, and decompose nodes into finer level. The semantic hierarchy idea is novel for the activity recognition problem.

In activity recognition problem, [14] has a different hierarchy concept than our paper. They consider viewpoints

Algorithm 2 Auto Annotation Update

Input: Feature dataset $\mathbf{T} = \bigcup_{i=1}^{N} \mathbf{T}_i$, $(\mathbf{T}_i \cap \mathbf{T}_j = \varnothing, i \neq j)$

for each subset $\mathbf{T}_i$ **do**
  Generate specific training set $\mathbf{C}_i = \bigcup_{j=1, i \neq j}^{N} \mathbf{T}_j$
  Generate feature dictionary $\mathbf{D}_i = f(\mathbf{T}_i)$, where $f(.)$ denotes the time-shift and time-truncation operations on the samples $\{\mathbf{x}_j^i\}$ in $\mathbf{T}_i$
  Train activity detector $\mathbf{M}_i$ on $\mathbf{C}_i$
  for $j = 1$ to $|\mathbf{T}_i|$ **do**
    Find expanded feature set $\mathbf{d}_j^i = f(\mathbf{x}_j^i)$ for $\mathbf{x}_j^i$
    Apply detector $\mathbf{M}_i$ on $\mathbf{d}_j^i$ to generate scores $\mathbf{s}_j^i$
    Select the optimal feature: $\mathbf{x}_j^{i*} = \arg\max_{\mathbf{d}_j^i} \mathbf{s}_j^i$
    Replace the original feature: $\mathbf{x}_j^i \leftarrow \mathbf{x}_j^{i*}$
  end for
end for
return $\mathbf{T}$

and poses to create a hierarchy; in other words, an appearance hierarchy whereas ours is a semantic hierarchy. Also, our predicted multi-level description is not based on the basic-level category annotations, which is different from their method.

4. Experimental Results

We test HAN on the most widely used fine grained activity dataset [23]. We first introduce the dataset specifications and experiment setup, and then provide the results.

4.1. Dataset

This dataset contains 12 different actors, 44 videos, 5609 clips, 881755 frames. It contains 65 fine-grained activities (listed in a results table later, except one is background activity). It includes challenging tasks such as recognizing the difference between *cut dice* and *cut slices*, *take out from drawer* and *open/close drawer*.

HAN focuses on fine-grained activity recognition which is not the same type dataset as Hollywood and HMDB. In non fine-grained activity recognition dataset, clustering high inter-class variance activities will help less.

4.2. Experiment Setup

We use the middle video level features provided by [23]. These include Histogram of Oriented Gradients (HOG), Histogram of Oriented Flow (HOF) and Motion Boundary Histograms (MBH). The features were generated around densely sampled points, which are tracked for 15 frames in a dense optical flow field. For feature histograms, 4000 words, obtained by K-means clustering over a million sampled features, were used for each type of feature.

We use the same train and test split as in [23]. We use one actor's data as testing; the rest (11) for training. Thus, we use leave-one-person-out cross-validation but only for the 7 actors that are specified to be test subjects to be consistent with the experiments settings in [23]. For accuracy of classification, we use the same evaluation criteria and code provided by [23]. This consists of measuring average precision (AP) for each class and a mean average precision (mAP) across all the classes.

4.3. Meta and Infra Level Generation

We use the unsupervised learned similarity measure to build the Meta level (described in Section 3.1). In this evaluation, we combine 64 Base level nodes (we neglect the background activity) into 32 Meta level nodes. Figure 3 shows part of HAN structure which demonstrates the similarity pairs explored, and the synthesized nodes by combining them into Meta level nodes.

For the Infra level nodes, we use the K-means clustering to cluster nodes, as described in Section 3.2. We do not decompose Base level nodes that have much less than 10 samples. For others, we divide them into 2 to 4 Infra nodes each, depending on how many training samples each base node has; we maintain around or greater than 10 training samples per Infra node. In the kitchen dataset, we generate a total of 128 Infra nodes.

4.4. Classifier Training

We train classifiers for nodes at each level by using libsvm with Histogram Intersection Kernel (HIK) [20]. For the base level nodes, we use similarity pairs to help refine positive and negative training examples. If we are training the base level node for activity B_i (the most similar class is activity B_j), we need to use examples with label B_i, and the negative examples should exclude label B_j's examples. Similarly, we do not use examples from sibling nodes for the Infra level nodes as well. Note that this distinction is made possible by the use of HAN structure.

4.5. Classification Results

We compute scores for classifiers trained separately for HOG, HOF and MBH features. We combine the results of the three features by averaging their confidence values. Table 1 shows the mAP results for the three features and their combination for various configurations. The first two lines, showing the published baseline results and just the Base layer of HAN are almost the same. The next two lines show that each of the two levels improves accuracy over the baseline. The last line shows the results using the entire HAN structure. Note that using MBH features alone, HAN improves results by 8.5% compared to the baseline; combined improvement is 5.4% in mAP. The dataset also includes pose information but we did not experiment with this feature, as the baseline [23] shows these features to give a low mAP (34.6%)

Method	[23]	HAN	Method	[23]	HAN	Method	[23]	HAN	Method	[23]	HAN
change temperature	73.6	**75.2**	cut apart	38.4	**39.4**	cut dice	35.9	**38.5**	cut in	21.9	*21.9*
cut off ends	16.3	**21.1**	cut out inside	46.8	**52.1**	cut slices	54.0	51.0	cut stripes	34.4	**56.2**
dry	97.4	96.2	fill water from tap	67.2	**100**	grate	72.6	70.5	lid: put on	20.1	6.5
lid: remove	14.5	11.1	mix	37.6	**73.5**	move from X to Y	54.8	54.5	open egg	67.4	63.6
open tin	79.5	**88.5**	open/close cupboard	80.4	79.9	open/close drawer	83.7	79.6	open/close fridge	96.8	95.5
open/close oven	100	*100*	package X	47.6	**65.5**	peel	84.2	79.0	plug in/out	61.9	**69.8**
pour	70.7	70.5	pull out	61.9	**100**	puree	75.9	**100**	put in bowl	34.5	**37.7**
put in pan/pot	24.6	**35.0**	put on bread/dough	51.5	48.8	put on cutting-board	29.5	**35.1**	put on plate	26.9	25.9
read	55.4	53.2	remove from package	42.8	38.2	rip open	16.7	12.3	scratch off	6.4	**28.7**
screw close	50.3	48.8	screw open	58.7	58.3	shake	95.2	93.9	smell	87.4	*87.4*
spice	47.3	46.9	spread	10.0	**38.0**	squeeze	98.3	**100**	stamp	64.6	**100**
stir	77.7	74.4	strew	43.5	**61.6**	take & put in cupboard	46.0	*46.0*	take & put in drawer	43.2	**53.9**
take & put in fridge	56.9	**60.5**	take & put in oven	100	*100*	t. & put in spice holder	89.8	89.1	take ingredient apart	42.1	**50.7**
take out from cupboard	90.8	**93.1**	take out from drawer	93.8	93.3	take out from fridge	94.0	80.2	take out from oven	100	*100*
t. out from spice holder	87.4	75.4	taste	42.4	41.0	throw in garbage	95.9	**96.4**	unroll dough	45	**100**
wash hands	56.1	51.6	wash objects	80.6	79.1	whisk	94.4	93.8	wipe clean	12.8	**41.6**

Table 2. Average Precision for Individual Base Level Activities Recognition.

Method	HOG	HOF	MBH	Combine
[23]	52.9	53.4	52.0	59.2
Base	48.7	49.2	54.4	59.1
Base+ Meta	**54.1**	53.4	58.8	62.5
Base+ Infra	53.5	54.3	59.7	63.5
HAN	53.9	**54.9**	**60.5**	**64.6**

Table 1. Mean Average Precision numbers for different features and network configurations. Note that complete *HAN framework with combined features* achieves the best result.

In recent work, [34] has reported mAP of 70% on the same dataset. However, they use knowledge of pre-detected objects, such as the cabinets and refrigerator; hence, we do not consider these numbers to be directly comparable. One can argue that such objects could be known in a fixed environment but, nonetheless, this reduces flexibility. Without the use of knowledge of fixed objects, accuracy of [34] is 59.2%, which is lower than our result using HAN. For the activity classification task, we do not use the attribute annotation provided by [23].

In Table 2, we show mAP performance for individual activity types. Within the 64 fine grained activities, HAN gives better results than [23] in 28 activities. Notice that improvements for several activities are quite large and in 5 classes, AP rises to 100%. Performance is improved for many challenging activities such as *cut*. The Meta level node *Cut Something* helps distinguish finer cutting activities (e.g. *cut stripes*) from other non-cut activities. Similar improvements are also observed for the base activities *put in bowl* and *put in pan/pot*.

Infra level nodes can also help improve precision for decomposable activities. We can see that the *take& put in drawer* and *take& put in fridge* both benefit from decomposing the complex actions (take and put in.) *Spread* involves varying ingredients (bread, butter and oil), different tools (spoon, knife and hand) which cause high intra-class variance. Decomposing these types of base nodes into Infra nodes helps improve the video classification results.

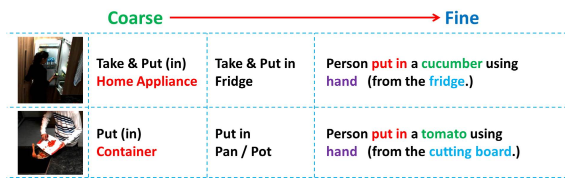

Figure 5. Output of HAN on two test videos. Coarse to fine descriptions are shown.

To show the performance of HAN, we use the feature provide by our baseline paper. All the experiment settings are the same except we add in HAN hierarchy (We use the exactly same feature provided by [23]). Second, kitchen dataset is very challenging. Few paper use this dataset because the result in [23] is hard to exceed.

4.6. Infra Level Attribute Experiments

In this section, we describe performance of attribute prediction in Infra Level. The Infra level description is composed by four attribute types: actions, tools, ingredients and containers. The attribute annotations are provided from [23] and used during training and for evaluation.

Single attribute detectors are trained using a linear SVM and features provided from [23], which are applied in both Base and Infra level nodes. The train and test splits are the same as for video level classification. We use HAN to predict attributes as described in Section 3.2 for each category. Then we use them to fill in a sentence template to describe the activity; some of the outputs are shown in Figure 5.

For quantitative evaluation of generated attributes, we calculate the mAP for each category. We find that among the 218 attributes provided from [23], several come with very few annotation examples; precision for these attributes is not reliable so we only include attributes that have at least 50 annotated samples. Under this criteria, there are 20 such attributes in the *action* category, 5 in the *tool* category, 7 in the *container* category, and 19 in the *ingredients* category.

The baseline uses outputs of the attribute detectors only,

Method	Action	Container	Tool	Ingredient
Attribute	59.69	47.38	71.48	29.97
Att + HAN	**65.03**	**49.95**	**72.36**	**32.41**

Table 3. Attribute mAP evaluation.

Attribute Name	Attribute	Attribute + HAN
add *(Action)*	42.72	**49.56**
change temperature	51.22	**54.38**
put in	59.05	**66.51**
take apart	27.41	**46.92**
baking tray *(Container)*	**21.82**	14.28
bowl	30.03	**35.39**
counter	30.64	**35.18**
cutting board	81.78	**85.52**
drawer	64.85	**67.35**
pan	66.61	**78.37**
pot	**35.91**	33.57
apple *(Ingredient)*	28.42	**29.81**
bottle	**56.25**	55.10
carrot	9.60	**11.47**
cheese	6.67	**10.84**
onion	13.43	**23.37**
orange	23.14	**29.05**
paper box	17.57	**23.60**
spice shaker	73.26	**86.98**

Table 4. Average Precision for part of Action Attributes, Container Attributes, and Ingredients attribute.

while our method integrates HAN into the prediction. The results are given in Table 3. HAN improves attribute prediction by 5.4% for action attributes, 2.6% for Container attributes, 1% for the tool category and 2.5% in ingredient category. In total, we improve for about 3% in average.

In detail, we show the accuracy of attribute results for four actions in Table 4. We observe that HAN improvements are higher in actions that have high intra-class variability. In Base level, *take ingredient apart* is of high variance, because the ingredients can be *tomato, orange, dough*, the tools can be *hand and knife*; also each actor has different ways of performing this action. Decomposition of base level node helps increase accuracy by clustering similar examples. *Take & put in* activity is also high in intra-class variations as it involves different containers such as *drawer, fridge and cupboard*; again, decomposition shows high improvements. On the other hand, *change temperature* has few variations and decomposition does not improve accuracy by much.

Table 4 shows attribute prediction results for containers. HAN improves results for 5 of the 7 container attributes. Table 4 shows attribute prediction results for ingredients. HAN improves results for 13 of the 19 ingredient attributes. The improvements come about, because HAN brings in activity/objects context: for example, cheese is often related to the *strew* activity, and orange is often related to the *squeeze* activity.

Method	HOG	HOF	MBH	Combine
HAN	53.9	54.9	60.5	64.6
HAN + Expand=9	**54.3**	56.4	63.6	66.4
HAN + Expand=16	54.1	**56.8**	**64.7**	**67.8**

Table 5. Self Annotation Update Improvement

4.7. Annotation Update Experiment

As we have 12 subjects, we use 10 subjects as a training set (T_1 set), 1 subject for annotation improvement (T_2 set), and 1 subject as test data. Using the method described in Section 3.4, the annotations are updated in T_2. Similarly, we update the annotations for videos in subset in T_1. After one round of iteration, we get improvement of 3.2%, achieving 67.8% mAP in total based on HAN. We compare to the original result in Table 5 with two experiment set expansion =16/interval =0.1, expansion =9/interval =0.1 (using the same feature provided in [23]). In detail, for each feature in T_2 (when per feature expansion = 16, interval=0.1, total video length = t), we pick a start time point from candidate pool $[0, 0.1t, 0.2t, 0.3t]$, and pick a end time point from candidate pool $[0.7t, 0.8t, 0.9t, t]$. In this way, one origin feature will expend to 16 features.

Annotation Quality: In our experiment, we find that around 23.2% video clips need annotation boundary refinement. After sorting the results, we find that the annotation for Fridge / Drawer / Cupboard related activities are ambiguous. That might be because of the *open fridge* includes walking towards the fridge, *Take out from fridge* includes partial *open/close fridge*, *Take out from drawer* includes partial *open/close drawer*. Also, whether *throw in garbage* should include collect garbage before throwing. Annotation for fine grained activities is challenging, and auto annotation updates help improve classification performance.

5. Conclusion

We described a new semantic hierarchy model for activity recognition named HAN. Unlike previous temporal level hierarchy models, HAN is based on an abstraction hierarchy. It includes three different levels which can describe the videos from coarse grained to fine grained. We demonstrate that the use of HAN improves the accuracy of activity recognition and also provides improved attributes for description. HAN can be easily extended to multiple levels of hierarchy. It would be also interesting to explore incorporating these levels into a temporal hierarchy for longer duration activities.

Acknowledgment: This research was supported, in part, by the Office of Naval Research under grant N00014-13-1-0493.

References

[1] L. Bourdev, S. Maji, T. Brox, and J. Malik. Detecting people using mutually consistent poselet activations. In *Proc. ECCV*, 2010. 2

[2] L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3d human pose annotations. In *Proc. ICCV*, 2009. 2

[3] S. Branson, P. Perona, and S. Belongie. Strong supervision from weak annotation: Interactive training of deformable part models. In *ICCV*, 2011. 2

[4] S. C., Kanaujia, and A. Metaxas. Conditional models for contextual human motion recognition. In *CVIU*, 2006. 2

[5] J. Deng, N. Ding, Y. Jia, A. Frome, K. Murphy, S. Bengio, Y. Li, H. Neven, and H. Adam. Large-scale object classification using label relation graphs. In *ECCV*, 2014. 1, 2

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2012. 4

[7] R. Farrell, O. Oza, N. Zhang, V. Morariu, T. Darrell, and L. D. Birdlets. Birdlets: Subordinate categorization using volumetric primitives and pose-normalized appearance. In *ICCV*, 2011. 2

[8] S. Guadarrama, N. Krishnamoorthy, G. Malkarnenkar, S. Venugopalan, R. Mooney, T. Darrell, and K. Saenko. Youtube2text: Recognizing and describing arbitrary activities using semantic hierarchies and zero-shot recognition. In *ICCV*, 2013. 2

[9] J. S. Gupta A., Srinivasan P. and D. L.S. Understanding videos, constructing plots learning a visually grounded storyline model from annotated videos. In *CVPR*, 2009. 2

[10] A. Jain, A. Gupta, M. Rodriguez, and L. S. Davis. Representing videos using mid-level discriminative patches. In *CVPR*, 2013. 2

[11] V. Kantorov and I. Laptev. Efficient feature extraction, encoding and classification for action recognition. In *CVPR*, 2014. 2

[12] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011. 2

[13] N. Kumar, P. Belhumeur, A. Biswas, D. Jacobs, W. Kress, I. Lopez, and J. Soares. Leafsnap: A computer vision system for automatic plant species identification. In *ECCV*, 2012. 2

[14] T. Lan, T.-C. Chen, and S. Savarese. A hierarchical representation for future action prediction. In *ECCV*, 2014. 5

[15] I. Laptev. On space-time interest points. In *IJCV*, 2005. 2

[16] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008. 2

[17] M. M. lek and C. Schmid. Constructing category hierarchies for visual recognition. In *ECCV*, 2008. 5

[18] J. Liu, B. Kuipers, and S. Savarese. Recognizing human actions by attributes. In *Proc. CVPR*, 2011. 2

[19] S. Maji. Discovering a lexicon of parts and attributes. In *ECCV Workshop*, 2012. 2

[20] S. Maji, A. C. Berg, and J. Malik. Classification using intersection kernel svms is efficient. In *CVPR*, 2008. 4, 5, 6

[21] L. I. Marszalek, M. and C. Schmid. Actions in context. In *CVPR*, 2009. 2

[22] H. Pirsiavash and D. Ramanan. Parsing videos of actions with segmental grammars. In *CVPR*, 2014. 2

[23] M. Rohrbach, S. Amin, M. Andriluka, and B. Schiele. A database for fine grained activity detection of cooking activities. In *CVPR*, 2012. 1, 3, 6, 7, 8

[24] M. Rohrbach, M. Regneri, M. Andriluka, S. Amin, M. Pinkal, and B. Schiele. Script data for attribute-based recognition of composite activities. In *ECCV*, 2012. 1, 2, 5

[25] S. Satkin and M. Hebert. Modeling the temporal extent of actions. In *ECCV*, 2010. 3

[26] K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human action classes from videos in the wild. In *CRCV-TR-12-01*, 2012. 2

[27] C. Sun and R. Nevatia. Large-scale web video event classification by use of fisher vectors. In *WACV*, 2013. 2

[28] C. Sun and R. Nevatia. Discover: Discovering important segments for classification of video events and recounting. In *CVPR*, 2014. 2

[29] H. Wang, A. Klaser, C. Schmid, and C.-L. Liu. Action recognition by dense trajectories. In *CVPR*, 2011. 2

[30] L. Wang, Y. Qiao, and X. Tang. Video action detection with relational dynamic-poselets. In *ECCV*, 2014. 2

[31] Y. Wang and G. Mori. Human action recognition by semilatent topic models. In *TPAMI*, 2009. 2

[32] K. T. Wong, S.F. and R. Cipolla. Learning motion categories using both semantic and structural information. In *CVPR*, 2007. 2

[33] B. Yao and L. Fei-Fei. Modeling mutual context of object and human pose in human-object interaction activities. In *CVPR*, 2010. 2

[34] Y. Zhou, B. Ni, S. Yan, P. Moulin, and Q. Tian. Pipelining localized semantic features for fine-grained action recognition. In *ECCV*, 2014. 3, 7