SBGAR: Semantics Based Group Activity Recognition

Xin Li Mooi Choo Chuah
Department of Computer Science and Engineering, Lehigh University
xil915@lehigh.edu chuah@cse.lehigh.edu

Abstract

Activity recognition has become an important function in many emerging computer vision applications e.g. automatic video surveillance system, human-computer interaction application, and video recommendation system, etc. In this paper, we propose a novel semantics based group activity recognition scheme, namely SBGAR, which achieves higher accuracy and efficiency than existing group activity recognition methods. SBGAR consists of two stages: in stage I, we use a LSTM model to generate a caption for each video frame; in stage II, another LSTM model is trained to predict the final activity categories based on these generated captions. We evaluate SBGAR using two well-known datasets: the Collective Activity Dataset and the Volleyball Dataset. Our experimental results show that SBGAR improves the group activity recognition accuracy with shorter computation time compared to the state-of-the-art methods.

1. Introduction

Automatically recognizing human activities in videos is one of the core tasks in the field of computer vision. It has many potential applications. For example, Lao et al. [1] present an automatic video surveillance system by analyzing human motion in videos, Rautaray et al. [2] design a new human-computer interaction method based on real time hand gesture recognition, and Davidson et al. discuss the video recommendation system in use at YouTube in [3].

In the modern era, dramatic progress has been made in recognizing human activities within videos. For example, Wu et al. [4] and Li et al. [5] present models to recognize human activities based on RGB frames (or optical flow images). In addition, Du et al. [6] and Veeriah et al. [7] recognize human activities using 3D representation (body joints). Even though all of these approaches yield a good performance, they have limitations. For RGB-based approaches, only a few of them were evaluated on datasets which contain complex activities, like group activities. Compared to the single-person activity recognition task, group activity recognition requires a more robust scheme that can differentiate higher level classes of activities, e.g. how different persons interact with one another in a group activity. For 3D-based approaches, they rely on specific hardware sensors or some time-consuming algorithms to detect and locate the body joints. In addition, the sensors and the methods cannot work well when people are far away from the cameras/sensors or if the resolution of frames is low.

The contributions of this paper are as follows. First, we present a novel solution, namely SBGAR, for group activity recognition. It can be used to recognize single-person activity and group activity. Second, our proposed scheme is semantics-based. Specifically, we analyze the videos to generate sentences describing the video frames, and then recognize the activities based on the semantic meaning of these sentences. To the best of our knowledge, this is the first work that uses semantics to recognize human activities in videos. Finally, our solution yields significantly better performance compared to the state-of-the-art approaches using two well-known datasets.

The rest of this paper is organized as follows. In Section 2, we briefly discuss related work, followed by the introduction of some important building blocks in Section 3. In Section 4, we describe our proposed group activity recognition scheme and implementation details. We report our experimental results in Section 5. Finally, we conclude this paper in Section 6.

2. Related Work

In recent years, some research has been done to recognize group activities from videos. Lan et al. [8] believe that the contextual information of what other people in the scene are doing provides a useful clue for understanding high-level activities. Thus, they present a solution to recognize group activities by exploring group-person interaction and person-person interaction information.

Based on the similar intuition that a strong correlation exists between a person's action and the actions of other nearby people, Choi et al. [9] exploit the spatial distribution of pedestrians in the scene as well as their pose and motion to achieve a robust action classification result. Next, they present a solution in [10] for simultaneously tracking multiple people and estimating their collective activities. They

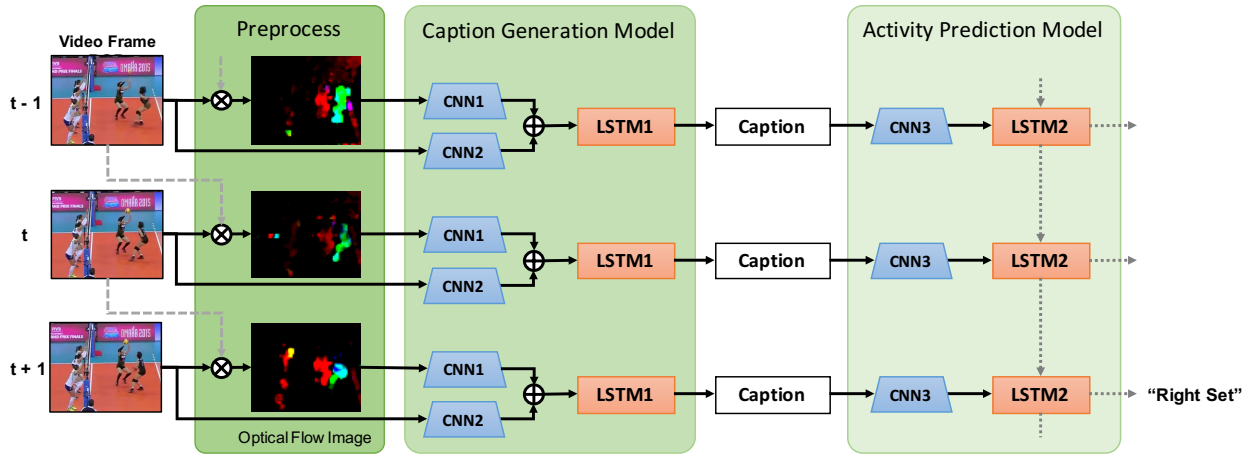

Figure 1: The architecture of the proposed Scheme. Caption Generation Model generates a caption to describe the corresponding frame. Activity Prediction Model is used to predict the group activity based on generated captions of a continuous sequence of frames. Symbol $\otimes$ indicates the operation of computing the dense optical flow image using two continuous frames, while symbol $\oplus$ indicates the operation of concatenating two CNN feature vectors into one single vector. In order to simplify the figure, the details of models are not shown here. Please refer to Figure 2 for more details of the Caption Generation Model, and Figure 3 for the Activity Prediction Model.

introduce a hierarchy of activity types which leads a specific person's action to the activity of the group.

Moreover, Ibrahim et al. [11] propose a hierarchical deep temporal model to infer group activities. Given a set of detected and tracked people, they run temporal deep networks (LSTMs) to analyze each individual person. They then aggregated these LSTMs over individual persons into a higher level deep temporal model. This allows the deep model to learn the relations between the people that contribute to recognizing a particular group activity.

Although all approaches discussed above achieve good performance in the task of group activities recognition, they are tracking based approaches. The biggest weakness of such approaches is their high computation time. These approaches have to first identify individual person in video frames, track their individual activities, and then later infer the relationships of their activities before they can predict the group activity label and thus incur much computation.

3. Important Building Blocks

Before we present the detailed description of our newly designed scheme, we first give a brief introduction about some building blocks.

1. Image Feature Extraction Via CNN: Convolutional Neural Network (CNN) [12] is a type of feed-forward artificial neural network. Recently, with the availability of efficient GPU computing, researchers have been able to train larger CNN-based networks. This has allowed CNNs to be widely used in solving several tough tasks such as image recognition [13], natural language processing [14], etc. In [15], Fischer et al. prove that CNN features contain more representative information of an image than other manually designed features, e.g. SIFT. In addition, CNN features perform well in the task of scene classification [16], which provides us a way to extract useful information of the scene from an image.

2. Long Short Term Memory: Long Short Term Memory (LSTM) model was first proposed by Hochreiter et al. [17]. It is a particular type of Recurrent Neural Network that works slightly better in practice, owing to its more powerful update equation and some appealing backpropagation dynamics. Donahue et al. [18] proposed a scheme which yields a good performance in the tasks of activity recognition, image description, and video description by using a LSTM model. Furthermore, Vinyals et al. [19] propose a Neural Image Caption (NIC) model based on LSTM to automatically describe the content of an image. They show that their model generates a better caption compared to other existing state-of-the-art approaches.

3. Dense Optical Flow: Dense Optical Flow was first proposed by Baker et al. [20]. It describes how each point in the scene moves from a frame to the next. In [21], Tao et al. propose an improved method, named SimpleFlow, which significantly reduces the computing time.

4. Proposed Scheme

Here, we present a novel model for recognizing group activities in videos. The intuition of our scheme is that people can easily infer an activity from a sequence of sentences. For example, given the following three sentences describing a volleyball game: *"There is a player jumping on the right side, while others are standing. There is one player spiking on the right side and three players blocking on the left side, while others are standing. All players are standing"*, a person can easily infer that the right team is performing an offensive action (spiking) while the left team is playing a defensive action (blocking). Thus, we design a model which generates a caption for each frame in a video and then pre-

dicts the activity based on a sequence of generated captions. Figure 1 shows the architecture of our scheme which consists of three steps: input preprocessing, caption generation, and activity prediction.

4.1. Preprocessing

We believe that both the scene features extracted from the original frames, and the movement features extracted using dense optical flow method [22] contribute towards group activity recognition. The original video frames contain more information about the environment, e.g. indoor or outdoor, while the derived optical flow images provide motion information. Thus, we use both types of features.

During preprocessing, we generate an optical flow image for every single video frame (except the first frame in a video). Given a video frame (frame t) as well as its previous one (frame $(t-1)$), we compute the displacement (direction and distance) of each pixel point in the frame. Then, in HSV color space, we set the direction and distance corresponding to the Hue and value plane correspondingly, and set the saturation value to be a constant value, e.g. 255. The generated optical flow images are illustrated in Figure 1.

4.2. Caption Generation Model

After preprocessing, at time t, we have an original video frame and its corresponding dense optical flow image. We extract CNN features from both original frames (CNN2 in Figure 1) and optical flow images (CNN1 in Figure 1). Then, we concatenate CNN1 and CNN2 into a single vector.

Next, we build a Language Model using the LSTM model. There are two reasons why a LSTM model is used here: (1). A LSTM model can generate good captions using the CNN feature vector as its input [19]. (2). A LSTM model also helps us handle some scenarios in which we need to split the scene into different groups, e.g. a left and a right team in a volleyball game. Figure 2 shows the details of our model for caption generation.

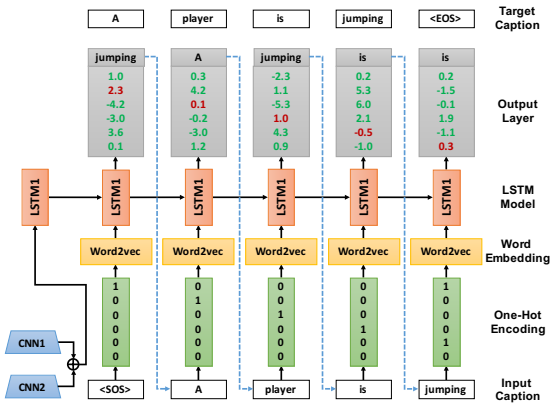

Figure 2: Caption generation model. <SOS> and <EOS> are symbols used to indicate the beginning and the ending of a caption correspondingly.

During the training process: The inputs of the Caption Generation Model consist of (i) concatenated CNN Features, (ii) Input Captions, and (iii) Target Captions (Ground Truth). In this paper, we encode each word of the Input Caption into a vector using One-Hot encoding. Considering One-Hot code is a high dimensional sparse feature which costs large storage and inefficient computation, we employ the word2vec model [23] to convert the One-Hot code into a continuous vector with a much lower dimension. We then feed the CNN Feature as well as the word2vec vector into a LSTM model (LSTM1) to generate the probability distribution of the next word in the sequence. Finally, the probability distribution will be compared to the Target Caption (Ground Truth) to tune the parameters of the model such that the predicted probability of the correct next word is higher than others.

Figure 2 shows the process when our model is fed the CNN Feature and the Input Caption "<SOS> A player is jumping" as the input, assuming the vocabulary is {"<SOS>", "A", "player", "is", "jumping", "<EOS>"}. The Output Layer contains the predicted probabilities that the LSMT1 assigns to the next word. The predicted result is "jumping A jumping is is", while the Target Caption is "A player is jumping <EOS>". It is obvious that such a prediction is not our expectation. Thus, we tune the parameters to increase the probability of the correct word (in red color) and decrease the probabilities of all other words (in green color). The process is repeated multiple times until the model converges and it can perform a good prediction.

During the testing process: The inputs of our model only consist of (i) CNN Features and (ii) Input Captions (initialized with a single starting symbol, <SOS>). The trained model, LSTM1, generates a probability distribution over what words are likely to come next. We then choose one word with the highest predicted probability and feed it right back into the model (blue dashed-line in Figure 2). This process is repeated many times until the predicted word with highest probability is the ending symbol, <EOS>, or the length of the generated caption is longer than a pre-determined threshold (e.g. 20).

4.3. Activity Prediction Model

The final step of SBGAR is to predict the activity label based on a sequence of generated captions using a LSTM model (LSTM2 in Figure 1). Instead of taking the captions as the input of the LSTM2 directly, we first employ a CNN model (CNN3 in Figure 1) to extract feature vectors from captions. The reason is threefold: 1. The lengths of generated captions vary while the input size of all cells in LSTM2 is the same. 2. A CNN model can generate vectors with the same dimension even if the lengths of input captions vary. 3. Kim et al. [24] show that a simple CNN model achieves excellent results in the task of sentence classification.

Figure 3 shows the details of our Activity Prediction Model. In this paper, we use a similar network as in [24] which originally consists of 4 layers. We remove the last layer of the network in [24] and concatenate its first three layers with a LSTM Model (LSTM2) by taking the output of Layer 3 as the input of the LSTM2. Using a LSTM model to analyze a sequence of captions makes intuitive sense, considering how such a model resembles the way we process language: reading sequentially. The first three layers of the network in [24] are:

Layer 1: In this layer, we employ word2vec model [23] to convert an input caption into a matrix. Each row of the matrix corresponds to one word. In Figure 3, we show two input captions. One caption consisting of 8 words and another consisting of 4 words. The dimension of word2vec is set to 5, thus these two input captions are represented by two matrices (8*5 and 4*5 correspondingly).

Layer 2: The second layer performs convolutions over the word matrix using multiple filter sizes. In vision, the filters slide over local patches of an image, while in the field of National Language Processing (NLP), we typically slide the filters over the full rows of the word matrix considering each row represents a word. Thus, we set the dimension of the filters equals to the dimension of the word matrix. In Figure 3, we only show 2 filter sizes (2*5 and 3*5). The 2*5 filter will slide over 2 words each time, while the 3*5 filter will slide over 3 words each time. We perform convolution operation on both word matrices using two filters and end up with two feature maps for each word matrix.

Layer 3: In this layer, max-pooling is performed on each feature map. As shown in Figure 3, after max-pooling, both input captions (different lengths) are represented as two dimensional features.

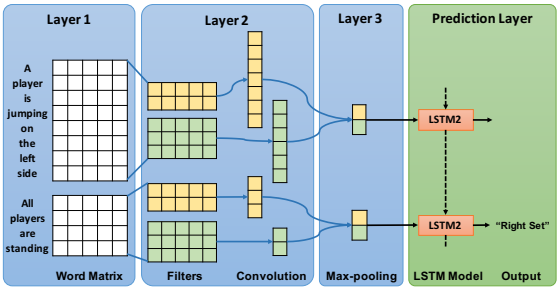

Figure 3: Activity prediction model.

In our SBGAR scheme, one caption is generated for each video frame and a sliding window of size $time\_steps$ frames is used to feed $time\_steps$ captions to LSTM2. For example, the window size shown in Figure 3 is 2 which means LSTM2 predicts an activity result based on 2 continuous input captions. By sliding the window, our model can analyze videos with varying number of frames.

During the training process: The inputs of our model consist of (i) a sequence of captions, more precisely $time\_steps$ captions, and (ii) their corresponding activity labels (Ground Truth). Given $time\_steps$ captions, our model first extracts CNN features from these captions and then feed the CNN features into the prediction layer (LSTM2) to generate a probability distribution for all potential labels. We then compute the mean value of the cross-entropy loss, as shown in Equation (1). The purpose of the training process is to minimize such a loss function L, where N is the size of the training set, y are the ground truth labels, and p are the predicted probabilities. During each training iteration, we tune the parameters of our model based on the value of this loss function L. We repeat feeding training captions and their corresponding labels to train our model until the value of L becomes smaller than a pre-determined threshold or the number of iterations is larger than another pre-determined threshold.

$$L(y,p) = -\frac{1}{N}\sum_{i=1}^{N} y_i \log p_i \quad (1)$$

During the testing process: The inputs of our model will only consist of $time\_steps$ captions. After the model generated a probability for all activity labels, we choose the one with the highest probability as the final result.

4.4. Implementation Details

We implement our scheme using Python Programming Language and Tensorflow [25]. Tensorflow is an open source software library for machine learning released by Google. We report the implementation details of our scheme and the settings of important parameters as follows.

CNN1 and CNN2: To extract CNN features from images, we use an Inception-v3 model [26] pre-trained on ImageNet [27] as a feature extractor. Specifically, we use the output of the final pooling layer (pool_3) in Inception-v3 model as the CNN feature of an image. Thus, the dimension of the extracted CNN feature is 2048 and the dimension of concatenated CNN1 and CNN2 features is 4096.

LSTM1: The LSTM1 is a 2-layer LSTM Model having 1024 hidden units. Before feeding the captions into the LSTM1, we use word2vec model [23] to convert each caption into a dense representation with a low dimension. We set the embedding size to 1024, thus the size of embedded captions is $n_w * 1024$, where n_w is a length of a caption. Because the size of all cells in a LSTM model are the same, so we use a transformation matrix $(4096 * 1024)$ and a bias vector $(1 * 1024)$ to transform a $1 * 4096$ CNN feature into $1 * 1024$ (1024 is the dimension of embedded captions). To do so, we only need to multiply the CNN feature $(1 * 4096)$ with the transformation matrix $(4096 * 1024)$ and add the bias vector $(1 * 1024)$. Then, we concatenate the transformed CNN feature with the embedded caption and feed them$((n_w + 1) * 1024)$ into LSTM1. **During the training process:** We set the learning rate to $1e^{-4}$ initially and reduce the learning rate every epoch until it reaches $1e^{-6}$. In order to reduce overfitting, we use the dropout technique

[28] and set the input & output keep probabilities to 0.75. **During the testing process:** The input caption is initialized with a starting symbol (<SOS>). We set the maximum length of the generated caption to be 20. The input and output keep probabilities are set to 1 to disable dropout.

CNN3: We first embed the generated captions into a dense representation using word2vec model before feeding them into the CNN model (CNN3). We set the embedding size to 5, so the size of the embedded caption is $n_w * 5$, where n_w is the length of a caption. Instead of using a pretrained CNN model, we implement a simple CNN model which only performs convolution and max-pooling operations with a generated caption as its input. Four filter sizes [3*5, 4*5, 5*5, 6*5] are used with 5 filters for each size. Thus, there is a total 20 filters in this CNN model. Each filter slides over the whole embedded caption using a VALID Padding Method (VALID padding means there will be no zero padding outside the edges when we do max pool). Once we have all the max-pooled outputs from each filter size, we combine them into one long feature. Thus, the length the feature generated by CNN3 is 20.

LSTM2: The LSTM2 is a 2-layer LSTM model. The sequence length of LSTM2 is set to 10, which means the LSTM2 will analyze 10 captions each time. **During the training process:** The learning rate is set to $1e^{-4}$ initially and reduced each epoch until $1e^{-6}$. We use the Adam algorithm [29] to minimize the cost function. To avoid overfitting, we employ the dropout method [28] and set the input and out keep probabilities to 0.75. **During the testing process:** The input and output keep probabilities are set to 1.

5. Experiments

We run our scheme on a desktop running Ubuntu 14.04 with 4.0GHz Intel Core i7 CPU, 16GB Memory, and NVIDIA Geforce GTX 1080 Graphics Card.

5.1. Datasets

We evaluate our scheme using two datasets: Collective Activity Dataset [9] and Volleyball Dataset [11].

Collective Activity Dataset: The Collective Activity Dataset has been widely used to evaluate the performance of group activity recognition schemes. It consists of 44 videos clips acquired using a low resolution hand-held camera. The location, action, and pose of each person in the videos is labeled. The five action categories include: crossing, waiting, queuing, walking, and talking while the pose categories include: right, front-right, front, front-left, left, back-left, back, and back-right. Thus, we trained the classifier to predict these five group activity categories depending on what the majority of the people included in the videos are doing: crossing, waiting, queuing, walking, and talking. Pose information is not used in our scheme.

Volleyball Dataset: The Volleyball Dataset was released by Ibrahim et al. [11] to evaluate the performance of group activity recognition schemes on sport footage. All videos related to volleyball games are collected from YouTube. In total, there are 1525 frames labeled with seven player action labels (waiting, setting, digging, falling, spiking, blocking, and others) and six group activity labels (right set, right spike, right pass, left pass, left spike, and left set). The location of each player is also labeled and that information is not used in our scheme.

5.2. Metrics

In order to compare our scheme with Ibrahim et al. [11], we use the same metrics used in [11].

Classification Accuracy: The accuracy is the percentage of the correct predictions.

Confusion Matrix: A confusion matrix [30] contains information about actual and predicted classifications generated by a classification system. In a confusion matrix, each column represents the instances of an actual class, while each row represents the predicted classes.

5.3. Baselines & SGBAR

In this paper, we want to compare the following baselines and SGBAR with some existing schemes proposed by other researchers.

B1. Single Frame Classification: B1 fine-tunes the Inception-v3 model for group activity recognition based on a single frame.

B2. Temporal Model with Image Features: B2 is the solution proposed by Donahue et al. in [18] where the image feature is extracted from the final pooling layer (pool 3:0) of Inception-v3 model and fed directly to a 2-layer LSTM model to recognize group activities.

B3. SBGAR (RGB Frame Only): B3 is a variant of our SBGAR scheme which only considers the RGB frames as the input ignoring any extracted optical flow information.

B4. SBGAR (Optical Flow Image Only): B4 is another variant of our SBGAR scheme which only considers the optical flow information while ignoring the information extracted from the RGB frames.

SBGAR (RGB Frame & Optical Flow Image): SBBAR considers information from both the RGB frame and optical flow image.

Comparing B1 & B2 allows us to see how much improvement can be obtained using a group of frames for group activity recognition. Similarly, comparing B3, B4 & SBGAR allows us to evaluate the improvement that can be achieved by combining both the scene and the motion related information.

5.4. Experiments on the Collective Activity Dataset

In this subsection, we report our experimental results using the Collective Activity Dataset. In order to train the caption generation model (LSTM1), we manually labeled a

caption for each training frame. Instead of generating complete sentences, we generate captions only using important keywords. The reasons are twofold : (1). Our purpose is to recognize group activities based on captions rather than generating complete sentences. Thus, our scheme will work as long as LSTM1 can generate several useful words. (2). Training a language model which can generate complete sentences incurs longer time, because it needs to learn the grammar which is not useful for activity recognition. Considering that this dataset contains the location and individual action of every person in each video frame, we can easily label captions for the actions of all players in the training frames as follows:

"<SOS> Walking Crossing Crossing Crossing <EOS>"

"<SOS> Waiting Waiting Waiting Crossing Walking <EOS>"

In Table 1, we report our experimental results (accuracy) using the Collective Activity Dataset and compare our SBGAR related and baseline methods with other existing methods. In [11], the authors compare their scheme with Contextual Model [8], Deep Structured Model [31], and Cardinality kernel [32] using the Collective Activity Dataset. Thus, we include the results they reported in Table 1. We follow the same experimental settings as used in [11], i.e., 1/3rd of the video clips were selected for testing and the rest for training. During the SBGAR related training process, we use 500 epochs to train the LSTM1 model and 300 epochs to train the LSTM2 model. For the LSTM2 model, we predict the final activity result based on a window size of 10 framse (5 before, current and 4 after frames)(the same setting as [11]).

Methods	Accuracy (%)
B1 - Single Frame Classification	67.2
B2 - Temporal Model with Image Features	68.5
B3 - SBGAR (RGB Frame Only)	83.7
B4 - SBGAR (Optical Flow Image Only)	70.1
Contextual Model [8] *	79.1
Deep Structured Model [31] *	80.6
Two-stage Hierarchical Model [11] *	81.5
Cardinality kernel [32] *	83.4
SBGAR (RGB & Optical Flow)	**86.1**

Table 1: Comparison of our scheme with baseline methods and previously published works on the Collective Activity Dataset. The results for "*" were extracted from [11].

The experimental results in Table 1 show that our proposed scheme outperforms the baseline methods as well as other existing schemes. It is worth pointing out that even when we only use a single feature (baseline B3), our proposed scheme can still achieve a higher accuracy than the state-of-the-art method in [32].

The baseline method B3 achieves a higher accuracy than B4 because most people in the videos in this dataset hardly move while they are talking, waiting, or queuing, which means not much useful information can be extracted from the optical flow analysis of these videos for activity recognition. B3 uses the information extracted from RGB frames and hence performs better.

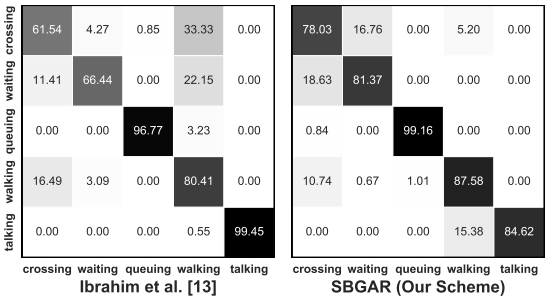

Figure 4: Comparison between [11] (left) and SBGAR (right) on the Collective Activity Dataset.

Figure 4 shows the comparison of the confusion matrices between the scheme in [11] and SBGAR using the Collective Activity Dataset. From this figure, one can see that [11] predicts some instances belonging to "crossing" and "waiting" as "walking", while SBGAR reduces this error. However, both [11] and SBGAR can not easily distinguish between "crossing" and "waiting". There are two reasons: 1. "crossing" and "waiting" often happen in the same scene, e.g. "at a cross road". 2. These two activities often happen sequentially, e.g. one waits at a cross road first, and then crosses. We notice that, comparing to [11], SBGAR predicts some "talking" instances as "walking". We discover that some video clips contain both activities and SBGAR believes that "walking" activity is more obvious than "talking" in these video clips.

5.5. Experiments on the Volleyball Dataset

In this subsection, we report our experimental results on the Volleyball Dataset. Based on the target activity labels (Left pass, Left set, Left spike, Right pass, Right set, Right spike), we notice that the labels contain information regarding whether the players are in the left or right side of the court, which means that we need to divide players into two groups. To handle this application scenario, we adjust the captions. We manually labeled captions for all training frames as follows:

"<SOS> Left: waiting moving blocking Right: standing spiking <EOS>"

"<SOS> Left: standing blocking Right: standing spiking <EOS>"

The order of the words describing the actions of each individual team is arbitrary. To make the training phase more efficient, we keep the order of the actions taken by both sides static (i.e. actions from the left are listed first). In Table 2, we report our experimental results (accuracy) using the Volleyball Dataset and compare the baseline and SBGAR related methods with existing methods. Two third of the video frames are used for training, and the remaining 1/3rd for testing (the same setting as Ibrahim et al. [11]). For SBGAR related methos, we use 500 epochs to train the LSTM1 model and 300 epochs to train the LSTM2 model. For the LSTM2 model, we predict the final activity result based on an observation window of 10 frames(5 before, current, and 4 after frames) (the same setting as in [11]).

Methods	Accuracy (%)
B1 - Single Frame Classification	41.9
B2 - Temporal Model with Image Features	44.3
B3 - SBGAR (RGB Frame Only)	38.7
B4 - SBGAR (Optical Flow Image Only)	54.3
Two-stage Hierarchical Model [11]	51.1
SBGAR (RGB & Optical Flow)	**66.9**

Table 2: Comparison of our scheme with baseline methods and previously published works on the Volleyball Dataset.

The experimental results show that our proposed SBGAR scheme outperforms the baseline methods and the state-of-the-art methods [11] on this dataset. It is worth pointing out that B4 (only a single feature is used) achieves a better result than [11].

For this dataset, B4 performs better than B3 by 15.6% in terms of achieved accuracy because the videos in the Volleyball dataset have the same scene (same viewpoint, similar background, similar color, etc) and hence fewer distinguishing features can be extracted in B3. However, B4 can extract more meaningful features (motion information) from the optical flow images.

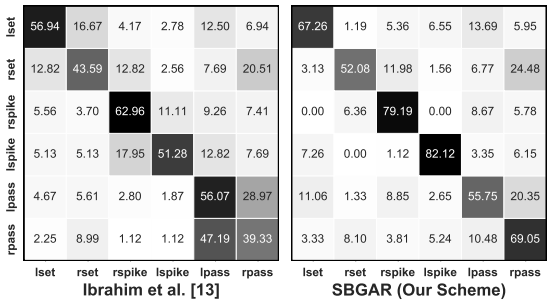

Figure 5: Comparison between [11] (left) and SBGAR (right) on the Volleyball Dataset.

Figure 5 shows the comparison of the confusion matrices between [11] and SBGAR on the Volleyball Dataset. It is clear that SBGAR achieves a better result in distinguishing activities which take place at the left and right side of the court, especially "lspike" versus "rspike". This improvement comes from the fact that our proposed model can generate captions for both the left and right parts. To a certain extent, the experimental results prove that our Caption Generation Model has the ability to consider the spatial information and represents such information in the generated captions correctly. However, we notice that both [11] and SBGAR predict some "lset" and "rset" samples as "lpass" and "rpass" correspondingly. This is because those "set" and "pass" activities are similar and often appear in the same region within a court from the view of the camera.

5.6. Impact of Key Parameters

The settings of parameter values have an impact on the predicted results of a Machine Learning model. Thus, we evaluated the impact of two key parameters:

Epochs: Each epoch is defined as the process of feeding the whole training set to a model. In SBGAR, we use two models, Caption Generation Model (LSTM1) and Activity Recognition Model (LSTM2). Thus, we will evaluate the impact of the number of epochs on their accuracy during the training of both models.

Observation Window Size of LSTM2: The observation window size is defined as the number of video frames that are used to generate a prediction. If the window size is 5, it means that LSTM2 will generate a prediction based on 5 consecutive frames.

We discuss the details as follows:

1. Epochs for LSTM1: In Figure 6, we report the accuracy of SBGAR on both datasets as we fix the number of training epochs of LSTM2 to 300 while varying the number of training epochs of LSTM1. The solid curve in blue color is the result using the Collective Activity Dataset, while the dashed curve in green color is the result using the Volleyball dataset. One can observe that larger epochs lead to higher accuracy. The accuracy becomes stable when the number of epochs exceeds 500 for both datasets. Figure 7 shows the training loss as we varies the number of epochs during the training process of LSTM1. The blue line with "*" marker shows the training loss, while the solid red line shows the testing loss. The training and testing losses decrease as the number of epochs increases and approach a stable value after 400 epochs.

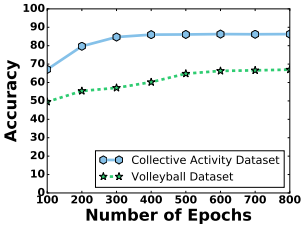

Figure 6: Activity recognition accuracy as the number of training epochs of LSTM1 is varied.

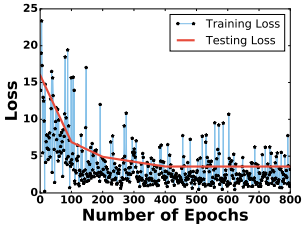

Figure 7: LSTM1 training loss as the number of training epochs is varied using the Collective Activity Dataset.

Based on both observations, we choose 500 as the number of epochs for training LSTM1.

2. Training epochs of LSTM2: In Figure 8, we report the testing accuracy of SBGAR on both datasets as we fix the number of training epochs for LSTM1 to 500 while varying the number of training epochs for LSTM2. The solid curve in blue color is the result using the Collective Activity Dataset, while the dashed curve in green color is the result using the Volleyball dataset. One can see the accuracy increases as the number of epochs increases and becomes stable after 200 epochs. Figure 9 shows the training loss on the Collective Activity dataset as we increase the number of epochs during the training process of LSTM2. The training and testing losses decrease as the number of epochs increases and become stable after 300 epochs.

Based on both above observations, we choose 300 as the default number of epochs for training LSTM2.

3. Observation Window Size of LSTM2: For video based activity recognition, only using frames before the

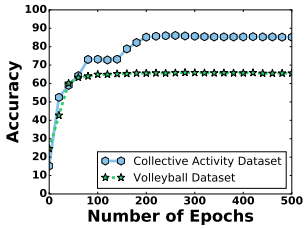

Figure 8: Activity recognition accuracy as the number of training epochs of LSTM2 is varied using both datasets.

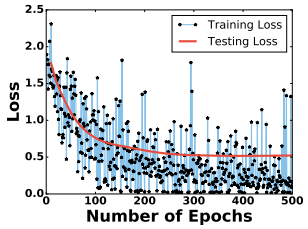

Figure 9: LSTM2 training loss as the number of training epochs is varied using the Collective Activity datasets.

current frame seems to make more sense in real life, considering that one can not access the frames after the current frame. A model which predicts a correct result only based on the previous frames may have the capability of early detection. Such a model is more useful for early-warning systems. However, adding some frames after the current frame may improve the prediction performance because more frames means more useful information can be used in the prediction process. Taking the volleyball sport as an example, assuming that a player is jumping, it is hard to say whether the player is "blocking" or "spiking" only based on this observation. By observing more frames, one can predict a more accurate result. Even though a model using "future" frames incurs additional delay, such a model may be more useful in some application scenarios.

In order to evaluate the performance of SBGAR with varying length of input frame sequence, we consider the following types of input frame sequences:

Before(x): x frames before the current frame are used as the input sequence.

After(x): x frames after the current frame are used as the input sequence.

Before(x)After(y): x frames before and y frames after the current frame are used as the input sequence.

Frame Sequences	Accuracy (%)	
	Collective Activity Dataset	Volleyball Dataset
Before(10)	85.7	64.7
Before(5)	84.1	64.7
After(5)	83.6	65.1
After(10)	84.7	65.1
Before(5)After(5)	86.1	66.9
Before(5)After(10)	85.9	67.4
Before(10)After(5)	86.3	67.1
Before(10)After(10)	**86.4**	**67.7**

Table 3: Accuracy on both datasets by taking variant input frames.

We report the experimental results in Table 3. One can easily notice that using a larger window size helps to improve the accuracy. In addition, comparing to only using frames before or after the current one (top 4 rows), using frames before and after the current frame (tail 4 rows) achieves a higher accuracy on both datasets. If we focus on the results of "Before(10)" and "After(10)", we can discover that "Before(10)" produces a better result on the Collective Activity Dataset, while "After(10)" performs better on the Volleyball Dataset. The same observation can be made between "Before(5)After(10)" and 'Before(10)After(5)". The reason of this is threefold: 1. The activities in the Collective Activity Dataset are more constant, which means there is no big differences between two continuous frames if they share the same activity, e.g. walking or queuing. 2. A video clip in the Collect Activity Dataset contains several activities, e.g. crossing and walking may happen alternately. 3. Activities in the Volleyball Dataset may involve the same action in their beginning frames, e.g. both blocking and spiking involve jumping. Thus, adding some frames after the current frame may cause a wrong prediction result in the Collective Activity Dataset, while it helps in the Volleyball Dataset.

5.7. Computation Time

For some application scenarios, e.g. sport analytics, it is highly important to be able to predict a group activity label in real time. Thus, we are interested in comparing the computation time of our scheme and the scheme in [11]. In Table 4, we report details of the computation time of our scheme. All data are averaged over 5 runs on the Volleyball dataset. With a sliding window of 10 frames, our scheme can predict on the average a group activity label within 108.5ms. If we use non-overlapping window of 10 frames, our scheme only takes about (22.19+27.78*2+28.63)*10+2.15=1065.95 ms (1.066sec). Running the code released by the authors in [11] using the same machine, the prediction time takes 4.22 seconds without including the time it takes to detect individual players. Thus, our scheme will be more useful for real-time prediction of group activity.

Process (Based on Single Frame)	Computation Time (ms)
Optical Flow Image	22.19
Extract CNN1 Feature (Inception-v3)	27.78
Extract CNN2 Feature (Inception-v3)	27.78
Caption Generation	28.63
Activity Recognition (Based on 10 Frames)	2.15
In Total	108.53

Table 4: Computation time of SBGAR.

6. Conclusion

In this paper, we propose a novel scheme (SBGAR) to recognize group activities in videos. The proposed method generates a caption for each video frame first, and then predicts the final activity categories based on these generated captions. The experimental results on two well-known datasets demonstrate the effectiveness and accuracy of our proposed method. Compared to the existing state-of-the-art methods, our scheme achieves a higher recognition accuracy with a shorter computation time.

7. Acknowledgement

This work is partially supported by a NSF CSR grant 1217379, a gift from Qualcomm and a GPU donated by NVIDIA.

References

[1] W. Lao, J. Han *et al.*, "Automatic video-based human motion analyzer for consumer surveillance system," *IEEE Transactions on Consumer Electronics*, vol. 55, no. 2, pp. 591–598, 2009.

[2] S. S. Rautaray and A. Agrawal, "Real time hand gesture recognition system for dynamic applications," *International Journal of UbiComp*, vol. 3, no. 1, p. 21, 2012.

[3] J. Davidson, B. Liebald, J. Liu, P. Nandy, T. Van Vleet, U. Gargi, S. Gupta, Y. He, M. Lambert, B. Livingston *et al.*, "The youtube video recommendation system," in *Proceedings of the fourth ACM conference on Recommender systems*. ACM, 2010, pp. 293–296.

[4] Z. Wu, X. Wang, Y.-G. Jiang, H. Ye, and X. Xue, "Modeling spatial-temporal clues in a hybrid deep learning framework for video classification," in *Proceedings of the 23rd ACM international conference on Multimedia*. ACM, 2015, pp. 461–470.

[5] Q. Li, Z. Qiu, T. Yao, T. Mei, Y. Rui, and J. Luo, "Action recognition by learning deep multi-granular spatio-temporal video representation," in *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*. ACM, 2016, pp. 159–166.

[6] Y. Du, W. Wang, and L. Wang, "Hierarchical recurrent neural network for skeleton based action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1110–1118.

[7] V. Veeriah, N. Zhuang, and G.-J. Qi, "Differential recurrent neural networks for action recognition," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4041–4049.

[8] T. Lan, Y. Wang, W. Yang, S. N. Robinovitch, and G. Mori, "discriminativeminative latent models for recognizing contextual group activities," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 8, pp. 1549–1562, 2012.

[9] W. Choi, K. Shahid, and S. Savarese, "What are they doing?: Collective activity classification using spatio-temporal relationship among people," in *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 1282–1289.

[10] W. Choi and S. Savarese, "A unified framework for multi-target tracking and collective activity recognition," in *European Conference on Computer Vision*. Springer, 2012, pp. 215–230.

[11] M. S. Ibrahim, S. Muralidharan, Z. Deng, A. Vahdat, and G. Mori, "A hierarchical deep temporal model for group activity recognition." in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[12] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[13] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: A convolutional neural-network approach," *IEEE Transactions on Neural Networks*, pp. 98–113, 1997.

[14] P. Blunsom, N. de Freitas, E. Grefenstette, K. M. Hermann *et al.*, "A deep architecture for semantic parsing," in *Proceedings of the ACL 2014 Workshop on Semantic Parsing*, 2014.

[15] P. Fischer, A. Dosovitskiy, and T. Brox, "Descriptor matching with convolutional neural networks: a comparison to sift," *arXiv preprint arXiv:1405.5769*, 2014.

[16] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition." in *ICML*, 2014, pp. 647–655.

[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 2625–2634.

[19] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: Lessons learned from the 2015 mscoco image captioning challenge," *IEEE transactions on pattern analysis and machine intelligence*, 2016.

[20] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," *International Journal of Computer Vision*, vol. 92, no. 1, pp. 1–31, 2011.

[21] M. Tao, J. Bai, P. Kohli, and S. Paris, "Simpleflow: A non-iterative, sublinear optical flow algorithm," in *Computer Graphics Forum*, vol. 31, no. 2pt1. Wiley Online Library, 2012, pp. 345–353.

[22] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Scandinavian conference on Image analysis*. Springer, 2003, pp. 363–370.

[23] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[24] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[25] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[26] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *arXiv preprint arXiv:1512.00567*, 2015.

[27] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 248–255.

[28] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[29] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[30] R. Kohavi and F. Provost, "Glossary of terms. editorial for the special issue on applications of machine learning and the knowledge discovery process," *Machine Learning*, vol. 30, no. 2-3, pp. 271–274, 1998.

[31] Z. Deng, M. Zhai, L. Chen, Y. Liu, S. Muralidharan, M. J. Roshtkhari, and G. Mori, "Deep structured models for group activity recognition," *arXiv preprint arXiv:1506.04191*, 2015.

[32] H. Hajimirsadeghi, W. Yan, A. Vahdat, and G. Mori, "Visual recognition by counting instances: A multi-instance cardinality potential kernel," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 2596–2605.