Active Vision for Early Recognition of Human Actions

Boyu Wang[1], Lihan Huang[1], Minh Hoai[1,2]
[1]Stony Brook University, [2]VinAI Research Vietnam
{boywang, lihahuang, minhhoai}@cs.stonybrook.edu

Abstract

We propose a method for early recognition of human actions, one that can take advantages of multiple cameras while satisfying the constraints due to limited communication bandwidth and processing power. Our method considers multiple cameras, and at each time step, it will decide the best camera to use so that a confident recognition decision can be reached as soon as possible. We formulate the camera selection problem as a sequential decision process, and learn a view selection policy based on reinforcement learning. We also develop a novel recurrent neural network architecture to account for the unobserved video frames and the irregular intervals between the observed frames. Experiments on three datasets demonstrate the effectiveness of our approach for early recognition of human actions.

1. Introduction

We propose a method for early recognition of human actions using multiple video cameras. Early recognition aims to recognize an action as soon as possible. This task arises in many situations, and the ability to make early and reliable decisions will enable a wide range of applications, from robotics to surveillance and health care.

Early recognition is an emerging research area, and several methods have been proposed in the last few years [1, 3, 6, 10, 13, 14, 16, 17, 21, 24, 30, 41, 42, 49–51, 53, 57]. However, existing methods only consider scenarios where temporal events can be wholly observed by a single camera at a fixed viewpoint. Unfortunately, human actions usually spread out in time and space, and a single camera with a fixed viewpoint cannot fully capture the progression of action due to limited view and coverage of the camera.

The limitations of a single camera can be overcome by using multiple cameras. Compared to systems with a single camera, the advantages of having multiple cameras are obvious: wider coverage and multiple perspectives. Unfortunately, these obvious benefits of multiple cameras might be difficult to realize in practice due to the lack of a method that can handle the constraints of the physical infrastructure. Multiple cameras will require more communication bandwidth and processing power. In many situations, communication bandwidth and processing power cannot be increased, putting an upper limit on the processing throughput. Although many cameras can be installed and the cameras might have high frame rates, not all the frames captured by the cameras can be transmitted and analyzed. Thus, temporal downsampling, i.e., frame dropping, is unavoidable. A simple approach is to apply the same downsampling factor to all cameras to satisfy the overall throughput limit. However, this uniform resource allocation strategy is unlikely to be optimal because some cameras might provide more information about the ongoing human action than other cameras, as illustrated in Fig. 1.

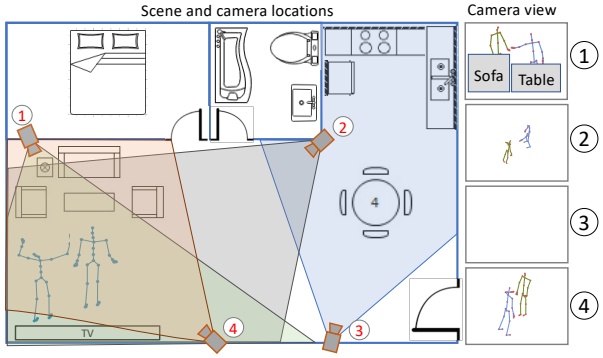

Figure 1: **Early recognition of human actions with multiple cameras.** Camera 1 is frontal but occluded by a sofa and a coffee table. Camera 2 shows a top-down view from far away. Camera 3 does not observe the action. Camera 4 only sees from the side. Due to the limited network and processing bandwidths, only one camera can be analyzed at a time. *Which camera should be used at each time step*, if no camera is always superior to the others?

If processing power is not constrained and if limited communication bandwidth is the only issue that we need to address, one may wonder if we can use low-resolution image or high compression factor instead. However, spatial downsampling and compression can reduce the accuracy of the system especially when the camera is far away from the scene of human action. Furthermore, to save com-

munication bandwidth, compression must be done prior to transmission using extra hardware equipment. This might be bulky, expensive, and not applicable to all cameras.

In this paper, we consider a scenario where there are k cameras, each with the maximum frame rate of l frames per second. However, the total processing throughput of the entire system is limited to l frames per second. Thus, the duration between two time steps is $1/l$ second; and at each time step, only one frame from a camera can be analyzed. Considering the importance of this scenario, we propose a framework for camera selection and early recognition of human actions. Our framework is developed based on reinforcement learning, treating camera selection as the decision of an artificial agent that is interested in maximizing its ability to recognize human actions as early as possible. The agent maintains a belief state based on the history of its observations from multiple cameras. We use a set of Recurrent Neural Networks (RNNs) [40] to model the belief state of the agent, from which the policy of the agent is parameterized and learned.

Reinforcement learning is a well-established framework for sequential decision making. But this framework is very general with numerous design choices, and defining the right state space or choosing the right reward function is not trivial. **Our first contribution** is a well-developed solution for camera selection and early recognition. We explicitly address the need for integrating observations over time for early recognition of human actions. We use recurrent networks for modeling the dynamics of human actions, but our situation requires a network architecture that is robust to the variable frame rate of an input video sequence (due to unobserved video frames). To this end, **the second contribution** of our work is the development of a novel recurrent network architecture that can estimate the missing values based on the last observed values and the elapsed time from the last observation. **Another contribution** is the approach for integrating information from multiple cameras for view selection and action recognition. Our framework and its components have been empirically validated on three multi-view or multi-modality datasets. The experiments show that the proposed recurrent network can robustly account for unobserved frames, and the learned policy for camera selection improves the early recognition ability of the camera system.

2. Related Work

Using multiple views for human action and activity recognition has been studied before [5, 18, 19, 33, 35, 45, 48, 54, 56]. These studies can be divided into two broad categories: explicitly building 3D models [48, 54] and integrating different 2D views. However, most prior studies were for offline recognition, and they assumed the cameras could be simultaneously used. They neither considered the early recognition problem nor addressed the need for camera selection. View-invariant features have been proposed in [2, 25, 44] for cross-view action recognition. But having a view-invariant representation is insufficient. Due to factors such as distance and occlusion, some views provide little information, and it is important not to select bad views.

Active sensing is an important research area in robotics. However, most works in robotics are for search-and-rescue or static object recognition, e.g., [4, 12, 27, 37, 59], and they are not applicable to early recognition of human actions.

Most similar to our work are the view selection methods [11, 46, 47]. Spurlock et al. [47] used a keyframe classifier for view selection, while Darrell and Pentland [11], Spurlock and Souvenir [46] also used reinforcement learning. However, these methods considered simpler types of human actions, where the classifier and the view selection policy can be defined based on *individual frames*. These methods are unsuitable for complex human actions, where the dynamics of the actions cannot be recognized accurately without integrating multiple observations over time. Empirically, these methods do not work well for complex human actions, as will be seen in Sec. 5. In this paper, we explicitly address the need for integrating observations over time for early decision making. This, in turn, requires a novel RNN architecture that can handle missing frames. This RNN architecture is a technical novelty by itself, and the whole framework is significantly different from the existing view selection methods.

Another related work is by Possas et al. [34] for human activity recognition. Their setting, however, is different from ours. In their setting, an ego-centric camera and a motion sensor can both record data about human activity. The data from the motion sensor is always analyzed, while the data from the camera might not be analyzed due to the need to reserve power. Possas et al. [34] learn a power-efficient policy to confidently recognize what already happened instead of what is happening.

Our work should not be confused with camera selection in TV broadcast [9, 38], where the selection policy sees all views *before* making the selection (no communication bandwidth problem). Meanwhile, we need to select a view without seeing the content of the views. Furthermore, the view selection policy for TV broadcast [9, 38] is mainly based on the visibility and changes of the human silhouette; it is not designed for action recognition.

3. Overview of Proposed Framework

Our framework for camera selection and early recognition is based on reinforcement learning. The core of the framework is an artificial agent that represents the system of multiple cameras. Camera selection is a sequential decision process of the agent, where the goal is to maximize the accuracy and minimize the latency in recognizing human actions. At each time step, the agent acquires an image

from one camera, analyzes the images, predicts the probabilities of the action classes. The goal of the agent is to maximize its sum of rewards, where the rewards depend on how accurate and early human actions can be detected and recognized.

Belief state. The belief state will integrate information from multiple sequences of images from multiple cameras. We will model the belief state based on RNNs [40] (including LSTMs [15] and IndRNNs [29]), but we extend them to account for missing observations and variable frame rates. How the belief state is modeled and the novelty of our architecture will be described in the next section.

Action and policy. At each time step, the agent will: 1) predict the class of the ongoing action, and 2) select a camera to acquire an image and analyze the scene. We will learn a probabilistic policy for both action recognition and view selection. For action recognition, the output at time t is the probability vector $\mathbf{p}_t$ for the action classes. For view selection, the output at time t is the probability vector π_t for camera selection: the i^{th} camera will be selected with the probability $\pi_t(i)$.

Reward. The reward is calculated based on the agreement between the recognition output and what action is occurring. Suppose there are m action classes. The *recognition output* of the agent at time t is an $m\times 1$ probability vector $\mathbf{p}_t$ where $\mathbf{p}_t(c)$ is the estimated probability for class c occurring at time t. Let $\mathbf{g}_t \in \{0,1\}^m$ be the ground truth binary indicator vector for what action classes occur at time t; $\mathbf{g}_t(c) = 1$ if class c is occurring and 0 otherwise. We will consider the reward function that is the sum of log likelihood: $r_t = \sum_{c=1}^{m} \mathbf{g}_t(c) \log \mathbf{p}_t(c)$. The goal of the agent is to maximize the average running reward: $\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} r_t$. By maximizing the average running reward, the agent will maximize the agreement between the predicted probabilities and the ground truth category vector at multiple time steps, leading to early recognition ability.

4. Belief state modeling and policy learning

Modeling the belief state and learning the policy are two most crucial steps for the success of the proposed method. The belief state must encode relevant information, and the policy must be appropriately parameterized for two tasks: 1) early recognition of human actions; and 2) camera selection. Unlike previous view selection methods [11, 46] that define the belief state based on individual frames, ours can integrate observations overtime to capture the dynamics of human actions. Our work is based on the recurrent network architecture [40], including LSTM [15] and IndRNN [29], **and we will commonly refer to them as as RNNs for brevity**. We introduce novel extensions to RNNs for handling missing observations and integrating information from multiple cameras. We use RNNs instead of other

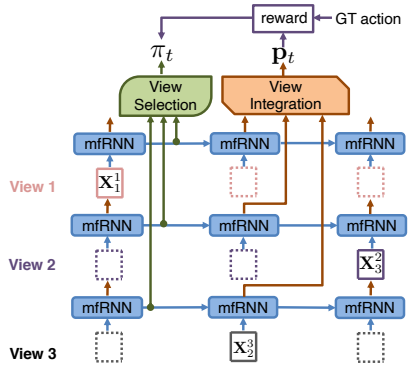

Figure 2: **Overview of our system** with three main modules: mfRNN, view integration, and view selection. $\mathbf{x}_t^j$ represents the input from view j at time t. A solid box means the view is selected while a dash box is not. View integration and selection are performed at every step. This figure shows one step only. See Sec. 4 for more details.

action classification methods, e.g., non-local network [52] due to the requirement of early recognition, i.e., to output the action probability at every time step without looking into the future.

We will model the belief state with multiple RNNs, one for each camera. Each RNN is a recurrent network that integrates observations over time, and it outputs the probability vector for the action classes in consideration. Each RNN analyzes its data stream independently of other RNNs, and each has its own ability to deal with unobserved frames. The combined recognition output is the weighted average of the outputs of the individual RNNs, where the weights are learned. For camera selection, we learn a policy that makes decision based on the concatenated hidden state vectors of the RNNs. Fig. 2 provides an overview of the system. We will describe below how to handle unobserved video frames, how to integrate recognition outputs from multiple RNNs, and how to learn the policy for camera selection.

4.1. Analyzing observations from a single camera

For each camera, we will train and use an RNN to integrate the observed video frames from the camera to obtain the predicted probability vector for multiple action classes. Using RNNs for integrating a sequence of observations is a powerful and popular approach for recognition, but most existing works often analyze and process observations at a regular interval, assuming the video frames are always observed. In our case, not all video frames can be observed and analyzed at the same time, due to limited system throughput. In this section, we briefly review RNN and then propose a novel extension to address missing observations.

RNN and its limitation. An RNN (including LSTM and IndRNN) is a recurrent network that integrates observations sequentially. Consider a camera, and let $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_t$ be

the sequence of image frames (or their feature representations) that are supposedly observed by the camera. At time t, the RNN analyzes an input $\mathbf{x}_t$, updates its internal state $\mathbf{h}_t$, and computes the output $\mathbf{p}_t$. Here $\mathbf{p}_t$ is the predicted probabilities for the action classes in consideration. The recurrent updates are:

$$\mathbf{h}_t = updateState(\mathbf{h}_{t-1}, \mathbf{x}_t); \mathbf{p}_t = computeOutput(\mathbf{h}_t).$$

Both $updateState$ and $computeOutput$ are parametric functions with learnable parameters. These functions have specific forms, such as the ones proposed in [15, 29, 40]. What is important to note here is that the RNN expects the input $\mathbf{x}_t$ at every step. Without the input, the state vector cannot be updated and the output cannot be produced. One can possibly skip the update procedure until the next observation, but this approach performs poorly in practice.

RNN for Missing Frames (mfRNN). We now describe an RNN extension that can account for missing video frames. Consider a camera, and let $\mathbf{x}_t$ be the video frame that is supposedly observed by the camera at time t, but the actual $\mathbf{x}_t$ might or might not be observed. If $\mathbf{x}_t$ is not observed, let $t' < t$ be the last time step where the video frame $\mathbf{x}_{t'}$ is observed. Let Δ_t be the elapsed time from the last observation: $\Delta_t = t - t'$. For an unobserved frame $\mathbf{x}_t$, we estimate the missing values based on the last observation and the elapsed duration Δ_t:

$$\hat{\mathbf{x}}_t = \mathbf{w}_t \odot \mathbf{x}_{t'} + (1 - \mathbf{w}_t) \odot \mathbf{x}_{null}, \quad (1)$$

$$\mathbf{w}_t = \exp(-\max(0, \Delta_t \mathbf{u} + \mathbf{v})). \quad (2)$$

We assume two nearby video frames are similar and the level of similarity depends on the time difference between the two frames. In the above, the symbol $\odot$ denotes the element-wise product between two vectors. The missing values are estimated based on the last observed values, taking into account the elapsed time from the last observation. We parameterize $\hat{\mathbf{x}}_t$ as a weighted linear combination of $\mathbf{x}_{t'}$ and a default observation vector $\mathbf{x}_{null}$. The linear combination is controlled by the weight vector $\mathbf{w}_t$, which is a function of the elapsed time Δ_t. The entries of $\mathbf{w}_t$ are decay functions of time; they are approximately one if the last observation is recent, and close to zero if the last observation is a distant past. The parameters of the decay functions are $\mathbf{u}, \mathbf{v}$, and the default vector for filling missing values is $\mathbf{x}_{null}$; all these vectors have the same dimensionality as $\mathbf{x}_t$.

The learnable parameters of an mfRNN are $\mathbf{u}, \mathbf{v}$, $\mathbf{x}_{null}$, and the normal parameters of an RNN. At time t, the input to the mfRNN is either $\mathbf{x}_t$ or $\hat{\mathbf{x}}_t$, depending on whether the frame at time t is observed or not. The output of the mfRNN is the class probability vector $\mathbf{p}_t$ of m dimensions, where m is the number of action classes. To learn the parameters of mfRNN, we minimize the negative log probability of the correct class at each time step t: $l_t = -\log \mathbf{p}_t(c)$, where c is the ground truth action class, and $\mathbf{p}_t(c)$ is the predicted probability for this class. The loss for a particular training sequence is the sum of the losses at all time steps: $\sum_t l_t$. By optimizing the total loss over multiple time steps, we force the mfRNN to make a correct prediction for partial actions, enabling early recognition ability. This loss function is consistent with the reward function defined in Sec. 3.

We can learn the parameters of an mfRNN for each camera using multiple training video sequences from the camera. From each video sequence, we can generate multiple training video sequences by randomly dropping some frames. This yields an augmented set of training data, proactively preparing the mfRNN for missing observations and also increasing its generalization ability for no missing observation cases.

The extension we proposed here is not specific to any recurrent network architecture. In our experiments, we experiment with both LSTMs [15] and IndRNNs [29], which are recurrent network architectures that achieved state-of-the-art performance on many sequence modeling tasks. For brevity, we will refer to all recurrent networks with the extension to handle missing frame estimation as mfRNNs. When it is necessary to specify the underlying architecture, we will refer to them as either **mfLSTM** or **mfIndRNN**.

4.2. Integrating information from multiple cameras

To integrate information from multiple cameras, we compute a weighted average of the mfRNNs' outputs. Recall from the previous subsection that there is one mfRNN for each camera, and the output of each mfRNN is a vector of class probabilities. Let $\mathbf{p}_t^i$ be the output at time t of the mfRNN for the i^{th} camera. We propose to aggregate multiple outputs by computing their weighted average, where the weights are determined based on the elapsed times from the last observations. The intuition is that for a certain view, if the elapsed time from last observation is large, the output of the corresponding mfRNN is unreliable, so it should contribute less to the consolidated output. Let Δ_t^i denote the elapsed time from the last observation for the i^{th} camera. We combine the outputs of the mfRNNs as follows: $\mathbf{p}_t = \sum_{i=1}^k \omega^i(\Delta_t^1, \cdots, \Delta_t^k)\mathbf{p}_t^i$. Here $\omega^i(\Delta_t^1, \cdots, \Delta_t^k)$ is the weight for the i^{th} view; and it is a function of the elapsed times $\Delta_t^1, \cdots, \Delta_t^k$. The set of weight functions can be seen as a network with k inputs and k outputs, mapping from the elapsed times $\Delta_t^1, \cdots, \Delta_t^k$ to the contribution weights. In this work, we use a simple network with two linear layers, a Leaky ReLU of 0.2 as the activation layer, and one soft-max layer. The parameters of this network are learned during the training phase.

4.3. Learning the view selection policy

To decide which camera to analyze at every step, a policy needs to be learned for camera selection based on the

history of previous observations. The input to the policy needs to contain information about what has been observed and what has been selected for observation. We therefore parameterize the policy function so that the input to the function has two parts: 1) the hidden states of mfRNNs, 2) the elapsed times from the last observations. Formally, the input of the policy function is: $\mathbf{s}_t = [\mathbf{h}_t^1, \cdots, \mathbf{h}_t^k, \Delta_t^1, \cdots, \Delta_t^k]$, where $\mathbf{h}_t^i$ is the hidden state of the i^{th} mfRNN and Δ_t^i is the elapsed time from the last observation for the i^{th} camera. This input vector $\mathbf{s}_t$ is essentially the belief state of the reinforcement learning agent for the view selection policy. We parameterize the policy function as a multi-layer perceptron that takes $\mathbf{s}_t$ as input and outputs the selection probability for each camera.

Let π denote this multi-layer perceptron policy function and $\boldsymbol{\theta}$ the parameters of the policy. Let $\pi(a|\mathbf{s}_t, \boldsymbol{\theta})$ be an output of the policy function, specifying the probability for selecting camera a. We use the advantage actor-critic [23, 31] (A2C) to learn the parameters $\boldsymbol{\theta}$. Each training instance is a set of video sequences from all cameras. Following the current policy $\pi(\cdot|\cdot, \boldsymbol{\theta})$ on the training instance, we obtain a training episode of belief states, actions, and rewards: $\mathbf{s}_1, a_1, r_1, \mathbf{s}_2, a_2, r_2, \cdots$. For each step t of the episode $t = 0, 1, \cdots$, the return G_t for time t is computed as a discounted sum of future rewards: $G_t = \sum_{j=t}^{T} \gamma^{j-t} r_j$, where γ the discounted factor. We then update the parameters of the policy function using the formula: $\boldsymbol{\theta} := \boldsymbol{\theta} + \alpha G_t \nabla_{\boldsymbol{\theta}} \log \pi(a_t|\mathbf{s}_t, \boldsymbol{\theta})$.

5. Experiments

5.1. Datasets and implementation details

We perform experiments on three multi-view or multi-modality datasets: NTU RGB-D dataset [43], IXMAS dataset [54], and nvGesture dataset [32].

NTU RGB-D dataset is the largest multi-view dataset for action recognition. This dataset is collected by three Microsoft Kinects, capturing human actions from different views simultaneously. The dataset contains 60 different action classes: 40 daily actions, 9 health-related actions, and 11 mutual actions of two people. There are 40 distinct subjects. We used cross-subject evaluation as in [43], i.e., 20 subjects for training and 20 other subjects for testing. We excluded a small portion of data that does not have three views. In total, there are 39,984 training and 16,395 testing sequences from all three views.

The dataset contains RGB images and skeleton information, but we only use the skeleton information in our experiments. Following [39], we use the distance matrix between skeleton joints to represent a skeleton. At time t, the original skeleton information is a vector of the 3D locations of 25 body joints, and we replace it with the pairwise distance matrix between the body joints. The distance matrix is symmetric, so only the upper triangular matrix is kept, vectorized, and used as the feature representation for a human performer. For a video frame and an action class with two performers, we concatenate the representation vectors of the two performers. For a video frame with only one performer, the feature representation vector of the performer is duplicated. The length of the feature vectors is 600.

IXMAS dataset contains 11 action classes and 10 actors. Each action was performed three times by each actor. The actions were recorded by four side-view cameras and one top-view camera. This dataset only contains RGB frames. We used the pre-trained I3D model [7] to densely extract features for each frame.

nvGesture dataset contains 25 different gesture classes, intended for human-computer interaction. A total of 20 subjects participated in data collection. The data was captured using two different cameras: one SoftKinetic depth camera recording depth and RGB frames in the front and another top-mounted DUO 3D camera capturing stereo IR. In addition, optical flow can be computed from the color stream and IR disparity map can be computed from IR-stereo pair, resulting in five different modalities. For each modality, there are 1,050 training and 482 test videos. As shown in [32], among all five modalities, the IR disparity performs worst and barely provides any benefit. In our experiments, we excluded the IR disparity modality. For this dataset, all sequences in the same modality have same temporal length. We uniformly split each sequence into 20 segments and used pre-trained I3D model to extract features for each segments. The I3D model was trained on three-channel RGB images and one-channel flow images. To apply them to one-channel depth or IR images, we inflated the one-channel images into three-channel images and fine-tuned the I3D model on these modalities. This is a weakly-labeled dataset, and some parts of the videos do not contain gestures. We therefore discarded the first three and last three segments. We only used the center segments, within which most gestures occur.

Overlapping views. Although NTU and IXMAS datasets contain certain overlapping camera views, they are still valid to compare different view selection policies, as also adopted by others [38, 46, 47]. Despite overlapping views, camera perspectives are drastically different and some views are more informative than others.

Implementation details. The mfRNNs for different cameras are trained separately. For the NTU dataset, we adopted IndRNN [29], which is an RNN architecture that achieved state-of-the-art performance on this dataset (*for recognition, not early recognition*). We used the same network architecture: 6-layer IndRNN with the hidden layer of 512 units and the dropout rate of 0.25. Note that our results are not directly comparable to [20, 28, 29, 39, 60] due to: 1) we

	View 1	View 2	View 3
Test Scenario 1			
IndRNN	63.84	59.83	56.34
IndRNN + data augmentation	65.62	61.08	58.66
IndRNN + input interpolation	66.03	62.33	59.75
mfIndRNN (proposed)	**70.53**	**65.38**	**60.99**
Test Scenario 2			
IndRNN	73.27	69.49	64.61
IndRNN + data augmentation	68.72	65.61	60.82
IndRNN + input interpolation	68.86	65.25	62.09
mfIndRNN (proposed)	**75.40**	**71.04**	**66.02**

Table 1: **Handling missing frames**. This shows the classification accuracies of several methods for two different test scenarios. The proposed mfIndRNN achieves the best performance on two test scenarios.

trained an mfIndRNN on each view separately, while these methods trained on all three views at once; and 2) some methods [20, 28, 39, 60] use CNN for sequence classification which *requires seeing the full video and therefore is not applicable for early recognition*. For the IXMAS dataset, we used one-layer LSTM with the hidden state size of 100 and the dropout rate of 0.5. For the nvGesture dataset, we used one-layer LSTM with the hidden state size of 512 and the dropout of 0.25.

Both actor and critic networks for view selection policy were two-layer perceptrons (hidden size of 512 and 128) and with Leaky ReLU of 0.2 as activation function. The discounted reward factor γ was 0.9. All the models were trained with Adam optimizer [22] with an initial learning rate of 10^{-4}, which was decreased by a factor of 10 when training performance plateaued. Training was stopped when the learning rate was smaller than 10^{-8}.

5.2. Handling missing frames

One contribution of this paper is the development of mfRNNs, novel recurrent network architectures for handling missing observations. In this section, we analyze the ability of mfRNNs and competing approaches for handling unobserved video frames. We analyze each camera separately, without camera selection and integration.

Methods for comparison. Our approach for handling missing frames has two notable steps: 1) train an RNN classifier with augmented data by random frame dropping; 2) extend the normal RNN architecture to include the null and time-decay parameters for filling the missing values. We consider three alternative methods for comparison: 1) use a normal RNN classifier without any modification and without using augmented training data. 2) perform data augmentation, but use a normal RNN classifier. 3) use a RNN classifier, while perform data augmentation by random frames dropping and use linear interpolation to fill the missing values (we assume future frames is available for interpolation, this is not true in practice). There are works [8, 26, 36, 58] that fill missing observations by either interpolation or imputation. Interpolation uses the temporal relation within the data steam to fill the missing value. However, this usually requires knowing the frames before and after the missing frame, but the frame after the current time is unavailable for online setting. Imputation usually results in a two-step process and missing patterns are not effectively explored [55].

Integration method	TestScenario 1	TestScenario 2
Uniform weights	75.35	79.76
Learned weights (proposed)	**76.66**	**81.28**

Table 2: **Comparison of two integration methods**. The first method is based on uniform averaging, and the second is based on weighted averaging with learned weights.

Test scenarios. We measure the classification performance of these methods for two test scenarios. In Test Scenario 1, the test sequences have variable frame rates due to the random frame dropping; the dropping rate ranges from 20% to 70%. In Test Scenario 2, the test sequences are the original test sequences; every frame is observed.

Results. Tab. 1 shows the experiment results for Test Scenario 1 on the NTU dataset. The actual RNN architecture used in this experiment is IndRNN [29], so the resulting network for handling missing frames is called mfIndRNN. Tab. 1 reports the action classification accuracy of four methods on three camera views separately. The reported numbers are averaged over 5 experiment runs. The proposed mfIndRNN achieves the best performance. Using augmented training data or input interpolation also improves the performance of IndRNN. This is expected, as the classifier is trained to anticipate missing video frames, and this scenario does occur during testing. However, the augmented training data may hurt the performance of an IndRNN classifier, as can be seen in Tab. 1, which shows the classification performance on the test sequences without missing frames. This is due to having the wrong type of augmented data: the test data has no missing frames, while the generated training data is severely different. On the other hand, the proposed mfIndRNN with learnable decay parameters has the right architecture to take advantages of the augmented training data, regardless of test scenarios.

5.3. Integrating information from multiple cameras

To integrate the outputs of multiple mfRNNs, we perform the following training process. Each training instance is a set of sequences from all camera views. We randomly select a view at every time step. At every time step, each mfRNN will output a probability vector. As described in Sec. 4.2, we learn a weighted combination of these outputs.

Policy/Method	acc@40	acc@100	$\overline{\text{acc}}$
Use All Views	59.90	81.28	61.77
Use View-invariant Features [39]	44.74	70.38	50.17
Use Keyframe Classifier [47]	16.22	43.57	26.01
Frame-based Q-Learning [46]	27.23	53.71	35.47
Always Select View 1	52.80	74.18	54.94
Always Select View 2	47.71	68.92	50.52
Always Select View 3	41.52	62.01	45.24
Select Random View	49.86	76.66	54.81
Cycle Thru All Views	51.69	78.02	56.31
The learned policy (proposed)	**54.55**	**79.62**	**58.01**

Table 3: **Comparison of different view selection policies on the NTU dataset.** This experiment assumes only one view can be analyzed at a time. acc@R is the classification accuracy when only R% of the action has occurred. $\overline{\text{acc}}$ is the average accuracy taken over all observation ratios.

During training, the parameters of the weight computation function are learned, and the parameters of the mfRNNs are finetuned.

We compare the proposed view combination method with a baseline where the outputs of multiple mfRNNs are averaged. Tab. 2 shows the performance of these methods under two test scenarios on the NTU dataset. In Test Scenario 1, only one view is available at a time, while in Test Scenario 2, all views are available at every step. In both cases, the proposed method outperforms the uniform pooling approach.

5.4. Evaluating the learned policy for view selection

Evaluation metrics. We consider three evaluation metrics: 1) recognition accuracy when only the first 40% of the action has observed (denoted as acc@40), 2) recognition accuracy when the full action has been observed (denoted as acc@100), and 3) the average early recognition performance (noted as $\overline{\text{acc}}$). The third metric is based on the average accuracy over different observational ratios. The **observational ratio** is the proportion of an action that has been observed when the recognition decision is made.

Comparison with direct baseline methods. We compare the learned policy with some baseline methods: i) keep using the same view; ii) select a random view at every step; and iii) select views in a cycle, such as 1, 2, 3, 1, 2, 3, etc. Experimental results are shown in Tab. 3, and there are some notable messages[1]. First, consider the top five policies shown in the table. *Always Select View 1* is better than the policies that keeps selecting either View 2 or View 3. This is understandable because View 1 is the frontal view. *Always Select View 1* is better than the *Select Random View* and *Cycle Thru All Views* when the observation ratio is 40% (acc@40), but not when the observation ratio is 100% (acc@100). This suggests that View 1 is much more informative than the other views at the beginning of the action, and it is important to observe and analyze more video frames from View 1. However, the additional information provided by View 1 will diminish as the observation ratio increases. Toward the end of the action sequence, it is better to observe the action from multiple cameras for better recognition accuracy. The reinforcement learning policy automatically learns which view to select at each time step, and it outperforms all other direct baseline policies.

Comparison with other view-invariant and view-selection methods. We also implemented three other methods for comparison, and the results are shown in Tab. 3. **1)** *Use View-Invariant Features*: we use the vectorized distance matrix as feature representation, and it is a view-invariant feature representation for skeleton data, as described in Sec. 5.1. Since the features are view invariant, we can use a single RNN for all views. Specifically, an IndRNN is trained with sequences which are generated by randomly selecting one view at each time step. However, only using view-invariant features is not enough, as can be seen in Tab. 3. Due to occlusion and other factors, one view might still be better than other views even though we use view-invariant features. **2)** *Use Keyframe Classifier* for view selection [47]. This method first uses iterative clustering to learn a set of keyframes that are discriminative for recognition. It then uses supervised learning to learn to select the discriminative frames at each time step. **3)** *Use frame-based Q-learning* for view selection [46]. However, this method, and also the second method, only considers the current frame for choosing the next view. The performance of these two methods are relative poor, possibly due to the lack of a recurrent network to integrate information from multiple observations. As can be seen from Tab. 3, the learned view selection policy outperforms other methods by a wide margin. This policy has the average early recognition accuracy $\overline{\text{acc}}$ of 58.01%. This is not too far below 61.77%, which is the average early recognition accuracy of the method that analyzes *all views from all cameras* at each time step. This unfair comparison is reported here for reference only.

Selection behavior of the learned policy. Due to space limit, the selection behavior of the learned policy at test time is shown in supplementary materials. This policy does not stick to any particular view, and it does not cycle through the views in any order. But the learned policy outperforms the random policy in our experiments, so it must take into account what is occurring and what has been observed to make the decisions. On average, less informative views are selected less frequently than other views. The selection frequency for View 1, 2, and 3 of the NTU dataset are 40.3%, 38.9%, and 20.9%, respectively.

[1] See the supplementary material for the entire performance curves.

Policy/Method	acc@40	acc@100	$\overline{\text{acc}}$
Always Select View 1	18.85	30.80	22.22
Always Select View 2	46.49	67.25	49.45
Always Select View 3	41.17	59.96	43.87
Select Random View	40.55	68.34	47.17
Cycle Thru All Views	40.02	71.32	48.85
The learned policy (proposed)	**49.88**	**73.74**	**53.30**

Table 4: **Performance of view selection policies when View 1 suffers from severe occlusions.**

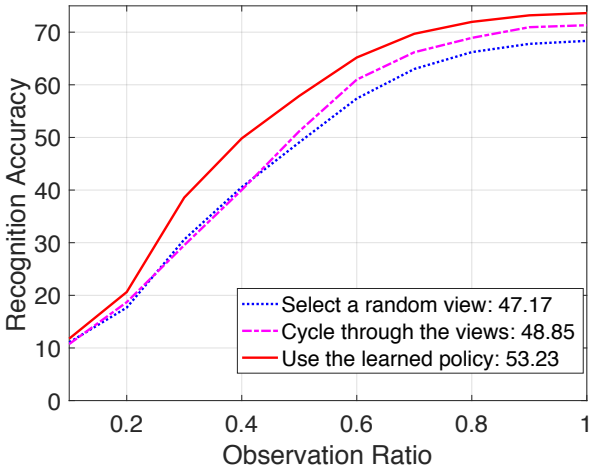

Figure 3: **Early recognition performance on the NTU dataset when View 1 suffers from severe occlusions.** This shows the recognition accuracy against the observational ratio, which is the proportion of an action that has been observed when the recognition decision is made.

Effect of severe occlusion. We further study the performance of the learned policy in the presence of severe occlusions. Assuming View 1 is blocked by a table, and only the upper body of a person is visible, we set the values of the leg joints in View 1 to 0. Tab. 4 shows the performance of different view selection policies under this condition. The learned policy learns to avoid View 1, and it outperforms other view selection policies by a wide margin. The overall selection frequency for Views 1, 2, 3 are 0.1%, 56.0%, 43.9% respectively. Fig. 3 shows the entire performance curves of the last three policies on the NTU dataset, plotting the recognition accuracy as a function of the observational ratio. To reduce clutter, we only plot the top performing policies in this figure.

Results on the IXMAS dataset. Tab. 5 shows the performance of different view selection policies on the IXMAS dataset. Following [46], we compute and report the leave-one-actor-out cross validation performance of the methods. The shown results for [38, 46, 47] are copied from the original papers. As can be seen, the learned policy achieves the best performance in all three metrics.

Method	acc@40	acc@100	$\overline{\text{acc}}$
Use All Views	98.79	99.39	97.52
Select Random View	95.94	97.15	92.21
Cycle Thru All Views	96.06	97.57	93.09
Use Keyframe Classifier [47]	n/a	84.0	n/a
Visibility-based Selection[38]	n/a	89.0	n/a
Frame-based Q-Learning [46]	85	94.24	83
The learned policy (proposed)	**97.64**	**97.87**	**94.32**

Table 5: **Performance on IXMAS dataset.**

Method	acc@40	acc@100	$\overline{\text{acc}}$
Use All 4 Modalities (our impl.)	65.35	82.37	68.20
Use All 4 Modalities (in [32])	n/a	83.4	n/a
Select Random Modality	58.22	78.59	62.29
Cycle Thru All Modalities	58.09	81.95	64.00
The learned policy (proposed)	**61.82**	**82.36**	**65.37**

Table 6: **Performance on nvGesture dataset.** The first two methods are shown for reference only; they use all four modalities at each time step, so they have unfair advantages over other methods.

Results on the nvGesture dataset. Tab. 6 shows the performance of view selection policies on the nvGesture dataset. The learned policy outperforms other view selection policies and that can only analyze one view at each time step. For acc@100, the learned policy performs as well as the method that uses all four modalities, even though the learned policy only uses one modality at a time. Note that the additional post-processing step used in [32] is not applicable to early recognition (due to the need for all frames). This explains the 1% accuracy gap between our implementation and [32] when using all four modalities.

6. Conclusions

In this paper, we study the problem of early recognition using multiple cameras. Our objective is to recognize human actions as early as possible under the constraint that only one camera can be accessed and analyzed at a time. We formulate this problem as a sequential decision process and develop our method based on reinforcement learning. Using reinforcement learning, we optimize a policy for selecting the best camera to use at each step and to integrate information from multiple cameras. We also propose mfRNN, a novel recurrent neural network architecture that can deal with unobserved video frames, improving the overall performance of our method in recognizing human actions.

Acknowledgement. This project is partially supported by Brookhaven National Lab, the US National Science Foundation Award IIS-1763981, and VinAI Research Vietnam.

References

[1] Yazan Abu Farha, Alexander Richard, and Juergen Gall. When will you do what?-anticipating temporal occurrences of activities. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[2] Saad Ali and Mubarak Shah. Human action recognition in videos using kinematic features and multiple instance learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(2):288–303, 2010.

[3] Mohammad Sadegh Aliakbarian, Fatemeh Sadat Saleh, Mathieu Salzmann, Basura Fernando, Lars Petersson, and Lars Andersson. Encouraging lstms to anticipate actions very early. In *Proceedings of the International Conference on Computer Vision*, 2017.

[4] Nikolay Atanasov, Bharath Sankaran, Jerome Le Ny, Thomas Koletschka, George J Pappas, and Kostas Daniilidis. Hypothesis testing framework for active object detection. In *Proceedings of the IEEE Conference Robotics and Automation*. IEEE, 2013.

[5] Zhuowei Cai, Limin Wang, Xiaojiang Peng, and Yu Qiao. Multi-view super vector for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[6] Yu Cao, Daniel Barrett, Andrei Barbu, Siddharth Narayanaswamy, Haonan Yu, Aaron Michaux, Yuewei Lin, Sven Dickinson, Jeffrey Mark Siskind, and Song Wang. Recognize human activities from partially observed videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[7] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[8] Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. Recurrent neural networks for multivariate time series with missing values. *Nature Scientific reports*, 8(1):6085, 2018.

[9] Jianhui Chen, Hoang M Le, Peter Carr, Yisong Yue, and James J Little. Learning online smooth predictors for realtime camera planning using recurrent decision trees. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[10] Lei Chen, Jiwen Lu, Zhanjie Song, and Jie Zhou. Part-activated deep reinforcement learning for action prediction. In *Proceedings of the European Conference on Computer Vision*, 2018.

[11] Trevor Darrell and Alex Pentland. Active gesture recognition using partially observable markov decision processes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1996.

[12] Enrique Dunn and Jan-Michael Frahm. Next best view planning for active model improvement. In *Proceedings of the British Machine Vision Conference*, 2009.

[13] Harshala Gammulle, Simon Denman, Sridha Sridharan, and Clinton Fookes. Predicting the future: A jointly learnt model for action anticipation. In *Proceedings of the International Conference on Computer Vision*, 2019.

[14] Minh Hoai and Fernando De la Torre. Max-margin early event detectors. *International Journal of Computer Vision*, 107(2):191–202, 2014.

[15] Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[16] Jian-Fang Hu, Wei-Shi Zheng, Lianyang Ma, Gang Wang, Jian-Huang Lai, and Jianguo Zhang. Early action prediction by soft regression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[17] Dong Huang, Shitong Yao, Yi Wang, and Fernando De La Torre. Sequential max-margin event detectors. In *Proceedings of the European Conference on Computer Vision*, 2014.

[18] Karim Iskakov, Egor Burkov, Victor Lempitsky, and Yury Malkov. Learnable triangulation of human pose. In *Proceedings of the International Conference on Computer Vision*, 2019.

[19] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. In *Advances in Neural Information Processing Systems*, 2017.

[20] Qiuhong Ke, Mohammed Bennamoun, Hossein Rahmani, Senjian An, Ferdous Sohel, and Farid Boussaid. Learning latent global network for skeleton-based action prediction. *IEEE Transactions on Image Processing*, 2019.

[21] Qiuhong Ke, Mario Fritz, and Bernt Schiele. Time-conditioned action anticipation in one shot. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[22] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *arXiv preprint arXiv:1412.6980*, 2014.

[23] Vijay R Konda and John N Tsitsiklis. Actor-critic algorithms. In *Advances in Neural Information Processing Systems*, 2000.

[24] Yu Kong, Dmitry Kit, and Yun Fu. A discriminative model with multiple temporal scales for action prediction. In *Proceedings of the European Conference on Computer Vision*, 2014.

[25] Yu Kong, Zhengming Ding, Jun Li, and Yun Fu. Deeply learned view-invariant features for cross-view action recognition. *IEEE Transactions on Image Processing*, 26(6): 3028–3037, 2017.

[26] David M Kreindler and Charles J Lumsden. The effects of the irregular sample and missing data in time series analysis. *Nonlinear Dynamical Systems Analysis for the Behavioral Sciences Using Real Data*, page 135, 2012.

[27] Catherine Laporte and Tal Arbel. Efficient discriminant viewpoint selection for active bayesian recognition. *International Journal of Computer Vision*, 68(3):267–287, 2006.

[28] Chao Li, Qiaoyong Zhong, Di Xie, and Shiliang Pu. Co-occurrence feature learning from skeleton data for action recognition and detection with hierarchical aggregation. *Proceedings of the International Joint Conference on Artificial Intelligence*, 2018.

[29] Shuai Li, Wanqing Li, Chris Cook, Ce Zhu, and Yanbo Gao. Independently recurrent neural network (indrnn): Building a longer and deeper rnn. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[30] Shugao Ma, Leonid Sigal, and Stan Sclaroff. Learning activ-

ity progression in lstms for activity detection and early detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[31] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2016.

[32] Pavlo Molchanov, Xiaodong Yang, Shalini Gupta, Kihwan Kim, Stephen Tyree, and Jan Kautz. Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[33] Ehsan Adeli Mosabbeb, Kaamran Raahemifar, and Mahmood Fathy. Multi-view human activity recognition in distributed camera sensor networks. *Sensors*, 13(8750–8770), 2013.

[34] Rafael Possas, Sheila Pinto Caceres, and Fabio Ramos. Egocentric activity recognition on a budget. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[35] S. Ramagiri, R. Kavi, and V. Kulathumani. Real-time multi-view human action recognition using a wireless camera network. In *Proceedings of the International Conference on Distributed Smart Cameras*, 2011.

[36] Kira Rehfeld, Norbert Marwan, Jobst Heitzig, and Jürgen Kurths. Comparison of correlation analysis techniques for irregularly sampled time series. *Nonlinear Processes in Geophysics*, 18(3):389–404, 2011.

[37] Sumantra Dutta Roy, Santanu Chaudhury, and Subhashis Banerjee. Active recognition through next view planning: a survey. *Pattern Recognition*, 37(3):429–446, 2004.

[38] Dmitry Rudoy and Lihi Zelnik-Manor. Viewpoint selection for human actions. *International Journal of Computer Vision*, 97(3):243–254, 2012.

[39] Alejandro Hernandez Ruiz, Lorenzo Porzi, Samuel Rota Bulò, and Francesc Moreno-Noguer. 3d cnns on distance matrices for human action recognition. In *ACM Multimedia*, 2017.

[40] D. Rumelhart, G. Hinton, and R. Williams. Learning internal representations by error propagation. In *Parallel Distributed Processing*, volume 1, chapter 8, pages 318–362. MIT Press, Cambridge, MA, 1986.

[41] M. S. Ryoo, Thomas J. Fuchs, Lu Xia, J. K. Aggarwal, and Larry Matthies. Robot-centric activity prediction from first-person videos: What will they do to me? In *International Conference on Human-Robot Interaction*, 2015.

[42] M.S. Ryoo. Human activity prediction: Early recognition of ongoing activities from streaming videos. In *Proceedings of the International Conference on Computer Vision*, 2011.

[43] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang. Ntu rgb+d: A large scale dataset for 3d human activity analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[44] Abhishek Sharma, Abhishek Kumar, Hal Daume, and David W Jacobs. Generalized multiview analysis: A discriminative latent space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[45] Richard Souvenir and Justin Babbs. Learning the viewpoint manifold for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[46] Scott Spurlock and Richard Souvenir. Multi-view action recognition one camera at a time. In *Proceedings of the IEEE Workshop on Applications of Computer Vision*, 2014.

[47] Scott Spurlock, Junjie Shan, and Richard Souvenir. Discriminative poses for early recognition in multi-camera networks. In *Proceedings of the 9th International Conference on Distributed Smart Cameras*. ACM, 2015.

[48] Pavan Turaga, Ashok Veeraraghavan, and Rama Chellappa. Statistical analysis on stiefel and grassmann manifolds with applications in computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[49] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Anticipating the future by watching unlabeled video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[50] Boyu Wang and Minh Hoai. Predicting body movement and recognizing actions: an integrated framework for mutual benefits. In *Proceedings of the International Conference on Automatic Face and Gesture Recognition*, 2018.

[51] Boyu Wang and Minh Hoai. Back to the beginning: Starting point detection for early recognition of ongoing human actions. *Computer Vision and Image Understanding*, 175: 24–31, 2018.

[52] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[53] Xionghui Wang, Jian-Fang Hu, Jian-Huang Lai, Jianguo Zhang, and Wei-Shi Zheng. Progressive teacher-student learning for early action prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[54] Daniel Weinland, Remi Ronfard, and Edmond Boyer. Free viewpoint action recognition using motion history volumes. *Computer Vision and Image Understanding*, 104(2):249–257, 2006.

[55] Brian J Wells, Kevin M Chagin, Amy S Nowacki, and Michael W Kattan. Strategies for handling missing data in electronic health record derived data. *eGEMs*, 1(3), 2013.

[56] Chen Wu, Amir Hossein Khalili, and Hamid Aghajan. Multiview activity recognition in smart homes with spatio-temporal features. In *Proceedings of the International Conference on Distributed Smart Cameras*, 2010.

[57] Zhen Xu, Laiyun Qing, and Jun Miao. Activity auto-completion: Predicting human activities from partial videos. In *Proceedings of the International Conference on Computer Vision*, 2015.

[58] Jinsung Yoon, William R Zame, and Mihaela van der Schaar. Deep sensing: Active sensing using multi-directional recurrent neural networks. In *ICLR*, 2018.

[59] Mabel M Zhang, Nikolay Atanasov, and Kostas Daniilidis. Active end-effector pose selection for tactile object recog-

nition through monte carlo tree search. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems*. IEEE, 2017.

[60] Pengfei Zhang, Cuiling Lan, Junliang Xing, Wenjun Zeng, Jianru Xue, and Nanning Zheng. View adaptive neural networks for high performance skeleton-based human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.