# PROCESS PROGRESS ESTIMATION AND ACTIVITY RECOGNITION

**BY XINYU LI**

**A dissertation submitted to the**

**School of Graduate Studies**

**Rutgers, The State University of New Jersey**

**In partial fulfillment of the requirements**

**For the degree of**

**Doctor of Philosophy**

**Graduate Program in Electrical And Computer Engineering**

**Written under the direction of**

**Ivan Marsic**

**and approved by**

_____

_____

_____

_____

**New Brunswick, New Jersey**

**May, 2018**

## ABSTRACT OF THE DISSERTATION

# Process Progress Estimation and Activity Recognition

**by Xinyu Li**

**Dissertation Director: Ivan Marsic**

Activity recognition is fundamentally necessary in many real-world applications, making it a valuable research topic. For example, activity tracking and decision support is crucial in medical settings, and activity recognition and prediction are critical in smart home applications. In this paper, we focus on activity recognition strategies and their applications to real-world problems. Depending on the application scenario, activities can be hierarchically categorized into high-level and low-level activities. The high-level activities may contain one or more low-level activities. For example, if cooking is a high-level activity, it may contain several low-level activities such as preparing, chopping, stiring, etc...

Although studied for decades, there are several challenges remaining for high-level activity recognition, also known as process phase detection. A high-level activity usually has a long duration and consists of several low-level activities. Treating high-level activity recognition as a per-time-instance classification problem overlooks the associations between activities over time. We thus proposed considering high-level activity recognition as a regression problem. Based on this assumption, we implemented a deep learning framework that extracts features from input data and designed a rectified tanh activation function to generate a continuous regression curve between 0 and 1. We used

the regression result to represent the overall completeness of the event process. Because the same event often follows similar high-level activity processes, we then used a Gaussian mixture model (GMM) to take the estimated overall completeness to supplement high-level activity recognition. Since the Gaussian mixture model requires that there be no duplication of high-level activities in an event (single activity has to follow Gaussian distribution), it might not fully represent real-world scenarios. To combat this limitation, we further proposed the use of LSTM layers to replace the GMM for high-level activity prediction. We applied our system to four real-world sports and medical datasets, achieving state-of-the-art performance. The system is now working in a trauma rooms at the Children's National Medical Center, estimating the overall completeness of each trauma resuscitation, the high-level activity of each trauma resuscitation, and remaining time for a trauma resuscitation to complete in real-time.

Compared to high-level activities, the low-level activities are more challenging to recognize. This is because low-level activity recognition often requires detailed, noise-free sensor data, which is often difficult to obtain in real-world scenarios. Many manually crafted features were proposed to combat the data noise, but these features were often not generalizable and feature selection was often arbitrary. We are the first to propose deep learning with passive RFID data for activity recognition. The automatic feature extraction does not require manual input, making our system transferable and generalizable. We further proposed the RSS-map representation of RFID data, which works well with ConvNet structures by including both spatial and temporal associations.

Because of the limitations of passive RFIDs, we extended our system from using a single sensor to working with a sensor network. We studied activity recognition with multiple sensory types, including RGB-D cameras, a microphone array, and the passive RFID sensor. We were able to follow previously successful activity recognition research focusing on each different sensor type. To build a system that makes final decisions based on features extracted from all sensors, we developed a modified slow fusion strategy, instead of traditional voting. We built a deep multimodal neural network that has multiple feature extraction sub-networks for different input modalities, that feed into a single activity prediction network. The multimodal structure is able to increase overall

activity recognition accuracy, but one key problem remains: the extracted features from different sensors contain both useful and misleading information. The system simply takes all the extracted features for activity recognition, because it does not know which features to rely on for activity recognition.

Addressing this issue, we proposed a network that automatically generates "masks" that highlight the important features for video-based activity recognition. Unlike many "attention" based deep learning frameworks, we used a conditional generative adversarial network for mask generating. This is because the conditional GAN gives us additional control of the generated masks, whereas we have no control of the generated attention map with regular attention networks. Our experimental results demonstrate that given manually generated activity performer masks as ground truth, the cGAN is able to generate masks that only highlight the activity performer. The activity recognition network with our proposed mask generator achieved performance comparable with other online systems on the published dataset. Though proven applicable, training the cGAN requires a large number of manually generated masks as ground truth, which is not often available in real-world applications. Building on the idea of a cGAN mask generator, we proposed a multimodal deep learning framework with attention that works with multi-sensory input. We proposed the feature attention and modality attention for feature extraction and fusion. The network can be fine-tuned by our asynchronous fine-tuning strategy using deep Q learning. Our experimental results demonstrate that our attention network with deep reinforcement learning based fine-tuning outperforms previous research. The proposed fine-tuning also prevents over-fitting when training a deep network on a small datasets.

Finally, we propose and introduce our ongoing work on concurrent activity recognition and our future work. Concurrent activity performance is common in the real-world: a person can drink while watching TV; a medical team can perform multiple tasks simultaneously through different medical personnel. However, recognizing concurrent activities remains an open research topic because it is neither a simple multi-class nor a binary classification problem. We proposed a shared feature extractor to extract

features from different input modalities. We then treated the concurrent activity recognition as a coding problem, and trained a deep auto-encoder to generate binary code denoting each activities' relevant and irrelevant features for activity recognition. However, this network was hard to train and converge because the shared features contains both . The recognition network easily over-fit to the unrelated features as opposed to the activity itself. Because the ground truth labels only provide whether the recognized activity is correct or incorrect, it disregards the associations between recognition results and the feature space. Addressing such an issue, we further proposed to modify the reinforcement learning based plugin that has been successfully used in our attention tuning to provide additional information for concurrent activity recognition. We asked human to provide feedback on whether the system made the decision based on the correct and associated features first, and then only partially tuned the network weights based on human feedback.

# Table of Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Overview

The activity recognition in constrained real-world environment, such as medical settings, remains to be a challenge. The challenges mainly from two aspects: 1. the real-world constraints (e.g., no RGB camera, no wearable devices) and 2. the complexity of activities in medical settings (e.g., crowded environment, RFID signal noise, view occlusion, etc.). Instead of solving the activity recognition problem at once, we break down the problem into high-level activity recognition (process progress estimation) and then activity recognition.

Work processes can be roughly categorized into two types: sequential and parallel. Our current work does not model parallel processes. *Linear* sequential processes can be partitioned into a set of "phases" that occur one after another in a fixed order. For example, during the trauma resuscitation process or the sports event broadcasting process, the phases are rarely skipped or duplicated. Previous process-phase detectors can be roughly categorized into three types: manual, shallow-modeled, and deep-learning-based. The first type used manually-generated event logs or medical equipment signals [20, 75]. As these systems worked with relatively noise-free datasets, they achieved good performance. However, manual detectors require manual log generation or signals from specific medical equipment, making them hard to implement and generalize. The use of sensors addressed these two issues at a cost: sensor data may be hard to obtain (e.g., wearable sensors are not preferred in medical settings as they may interfere with the work), and are subject to the hardware or environmental noise. The second type of phase detectors used shallow classifiers to predict phases from noisy sensor data [20, 10, 83]. Yet, the features and model selection were often arbitrary and difficult

to generalize, because different researchers using different features can only claim that those features worked best for their specific scenarios [55]. Due to its success in the image classification and speech recognition, deep learning and automatic feature extraction has given rise to the third type of phase detectors. All three approaches treated the process phase detection as a discrete classification problem, which overlooks the continuous nature of the processes and misses the associations between process phases and percentage completion. These limitations have caused systems to return logically impossible predictions [53].

We address these issues and estimate process completeness as a continuous variable. Our system is designed to work with sensor data, as opposed to manually-generated event logs. We designed a multimodal structure to make the system compatible with different types of input sources. Features in our model are automatically learned using a deep neural network. We introduce a deep regression model to continuously estimate process completeness, and the rectified hyperbolic tangent (*rtanh*) activation function to bound the regression output values in the context of process completeness. The system detects the current process phase based on the currently estimated process completeness using a Gaussian mixture model (GMM). Our model for completeness estimation is trained on the regression error as well as a novel conditional loss from the phase prediction. Finally, the remaining time is estimated during run-time using the calculated process execution speed.

We tested our model with two datasets: a medical dataset containing depth video and audio records of 35 actual trauma resuscitations at Children's National Medical Center (CNMC) [53], and an Olympic swimming dataset containing 60 YouTube video records of Olympic swimming competitions in different styles. For the resuscitation dataset, our process progress estimation system achieved 86% average online phase detection accuracy and 0.67 F1-score, outperforming existing systems. Specifically, our system incurred 12.65% completeness estimation error, with an average 7.5 minute remaining-time estimation error (14% of total duration). For the Olympic swimming dataset, the system achieved 88% accuracy with 0.58 F1-score and incurred 6.32% completeness estimation error with average 2.9 minute remaining-time estimation error

(18% total duration). Using additional network layers to replace the GMM, our system slightly outperformed existing process-phase detection systems on the trauma resuscitation dataset [105, 100], while simultaneously providing progress and the remaining time estimation.

The process progress estimation focuses on recognize the high level activities, but each high level activity (process phase) can be further breakdown into low level activities. For example, a trauma resuscitation consists of six phases. We started our work of low level activity recognition with RFID only in two steps. First, the use status of different objects is determined based on the RFID information, such as signal strength. Second, activities are predicted based on the use status of objects. For object-use detection, we used small, inexpensive, battery-free passive RFID tags attached to medical objects and fixed reader antennas. We placed tags on 10 object types commonly used in trauma resuscitation. Data from these tags were collected by eight RFID-reader antennas installed in the trauma room in the emergency department of a trauma center. By reviewing videos, medical experts of our team coded object-use data and a synchronized medical activity log from trauma resuscitations. We used these data to build our activity recognition model.

RFID-based activity recognition has treated activity recognition as a binary classification problem where a specialized classifier decides whether or not an activity of a particular type is occurring. These types of systems, however, may not be scalable to a large number of activities. In addition, the common approach for activity recognition involves two steps: first detect the use of objects associated with specific activities by detecting human-to-object-interaction from sensor data, and then recognize activities based on used objects [52]. The predication errors made by the system in the first step will be cascaded into second step and impair the final prediction result. Our approach for activity recognition uses passive RFID sensing. The RFID tags need to be strategically placed on objects of interest. Various features have been proposed and classifiers tested for RFID systems in different application settings [52, 79], which makes it unfeasible to compare their relative efficiency. As a result, feature and classifier selection for RFID data is often arbitrary.

We proposed and demonstrates a novel way for activity recognition from RFID data without using manufactured features [55]. To perform process-phase detection and activity recognition from RFID data, we treated the process-phase and activity recognition as a multi-class classification problem instead of extracting manufactured features and cascading object-use detection with activity prediction. We implemented a deep convolutional neural network with three convolutional layers and three fully-connected layers totaling 8.7M weights. The network was developed with a Microsoft Azure cloud computing platform [69] and locally with Google TensorFlow [1]. We trained this network with RFID data collected during 16 actual trauma resuscitations in a trauma center. Different networks were trained for process-phase detection and for activity recognition. We were the first to introduce the deep learning for passive RFID data processing and we proposed the "RSS map" which is a new RFID data representation that fits convolutional neural network. Our system recognized 10 common medical activities directly from RFID data with F-score 18% greater than an existing RFID-based system in the same application scenario [52]. To our knowledge, we are the first to apply deep learning with RFID sensing for activity recognition in complex teamwork.

A key limitation of our current system is that it relies only on relies on RFID sensing to capture activity information and making predictions. Some activities, such as palpation of the patient's body, do not involve the use of physical objects that can be tagged. In addition, RFID technology does not work very well with metal objects or liquid containers, and objects in sterile packages can be tracked only until the packaging is discarded. Our continuing research involves the use of video and audio for activity recognition. We introduce an activity recognition system that recognizes activities using video as input in two steps. First, it localizes the activity by generating an *activity mask* outlining the location where the activity is expected to occur. For mask generation, we used a conditional generative adversarial network (cGAN) [66]. Given that activities are continuous and usually represented as video clips, we introduced a Conv-LSTM-Deconv structure as a continuous mask generator with cGAN for training.

The cGAN-based image generator has shown better performance than a Conv-Deconv-based generator [34]. The ConvNet-LSTM structure has been used to model spatio-temporal associations [35, 80]. In addition to the adversarial loss, we introduced a spatio-temporal loss and implemented perceptual loss [36]. These losses penalize the neural network for pixel-wise errors in the generated mask and discontinuities between masks from consecutive frames. Second, the generated mask is appended to each color channel of its input video frame to delimit the activity region for the activity recognizer. Because the ConvNet-LSTM structure has been used successfully for modeling spatio-temporal activity associations [56, 103], we adopted a VGG-LSTM network for activity recognition. To train and test our system, we manually created activity masks for two datasets. We selected six activities from a well-known Olympic sports dataset (15 videos per activity) and for 10 frames of each video manually created binary masks outlining the activity performer's location. The proposed method works in real-time, as opposed to offline prediction based on features extracted from the entire video [107, 58].

Although we made progress on single person activity recognition there are three main aspects to the challenge: 1. Feature selection: real-time activity recognition often requires features directly associated with the activity for decision making. However, in real-world scenarios with multiple people and moving backgrounds, it is hard to extract only the associated features. This is noticeable especially in small datasets, where the lack of variance makes the system more likely to learn background features specific to the training data [35]. We proposed to generate a mask indicating the important feature subspace (an attention map). Previous research mainly uses ConvNet based attention modules with deep neural networks for image recognition [65]. We extend ConvNet attention to work with other input modalities, including video, audio, and mobile sensors. We also introduce modality attention for feature fusion, which assigns different modality branches dynamic weights based on their expected contribution to the activity predictions. 2. Network training: even with an attention mechanism, there is no way to guarantee the attention module is well trained, because the ground truth labels only provide right-or-wrong information regarding the final prediction. It is unclear whether we should penalize the entire net-work or only the attention module

during model training. It is also possible to make correct predictions based on irrelevant features (e.g. overfitting to background features). Addressing this issue, we introduce an asynchronous tuning strategy that first tunes the attention module, and then the recognition module only. This ensures the system first learns the relevant features to use when per-forming activity recognition. Because the ground truth labels do not provide information regarding attention region, we designed the system to use human feedback on samples of generated attention to improve attention generation. 3. Misalignment Between Observation and Ground Truth: we noticed that in many datasets the ground truth label is not well aligned with the data. For example, in the Olympic sports dataset, a video labeled as long-jump roughly contains 30% frames with no activity in progress. This is acceptable for some offline methods that first extract features from the entire video to make a prediction. But online systems making per-frame predictions would incur inappropriate loss when training on a labeled frame with no activity in progress. Addressing this issue, we modified the DQN model to assign rewards and penalties to each frame based on the accumulated prediction results instead of ground truth for each time instance.

We tested our system with three commonly used and challenging published datasets, the Olympic sports dataset [72], the Hollywood 2 dataset [65], and the 50-salads dataset [100] (contains depth, RGB, and 3-axis gyroscope). Our system is able to compete with state-of-the-art online activity recognition systems with 0.796 mAP on Olympic sports dataset, 0.631 mAP on Hollywood 2 dataset, and 0.449 mAP on the 50-salads dataset. The proposed asynchronous tuning with DQN improves mAP by around 6-9%. We further visualized the attention before and after tuning, demonstrating that our proposed method helps focus attention on more representative features.

## 1.2 Organization

The following sections are organized as follows: We will first introduce the deep regression model based process progress estimation strategy in chapter 3. In chapter 4, we are going to introduce our initial work using passive RFID for activity recognition with shallow classifiers and deep learning based strategies. In chapter 5, we introduce the

attention based multimodal activity recognition strategy. In Chapter 6 we introduce our under-going work on concurrent activity recognition and possible future works. Chapter 7 summarize our work and conclude the paper.

## 1.3 Contribution

Our work on multimodal real-time activity recognition can be summarized :

1 Introduced deep regression network to estimate the event process progress in real-time and deployed the system in an actual trauma room.

2 Designed RSS map representation for RFID and introduced the ConvNet for RFID based activity recognition.

3 Introduced conditional GAN based activity mask generating system and activity recognition based on it.

4 Introduced the multimodal attention network with DQN based tuning for online activity recognition.

5 Proposed multi-level fully connected network and encoder-decoder based network for concurrent activity recognition.

# Chapter 2

# Background and Related Work

## 2.1 Process Phase Detection

Process progress modeling can be done from three aspects: the completeness, the process phase, and the remaining time. Most of the previous research only focused on the process phase detection. Based on different application scenarios, previous works approached process modeling in three ways.

Systems designed to focus only on process modeling or work in very specific applications (such as certain types of surgery) used manually generated event logs [20] or the medical equipment signals [75] as the input. These types of input data are relatively noise free, and can be directly used for process modeling without preprocessing and feature extraction. Previous approaches treated process phase detection as a classification problem and directly applied shallow classifiers commonly used for sequential data analysis (e.g., HMM or decision tree [75]). Such approaches have proven accurate and easy to implement, and some association rules could be mined by analyzing the generated HMM transition matrix or decision tree topology [31]. But the drawback of these approaches is the use of manually generated event log or the data from specific equipment, making these systems hard to deploy, and the trained classifier may have difficulty working with high-dimensional and noisy sensor data.

To avoid such limitations, we designed our system to use commercially available sensors as the input source. There are previously proposed process phase detection systems using different sensor data [9, 55, 53]. For daily living and certain surgery scenarios, the RGB camera and wearable sensors were used to capture the gesture and posture for process phase estimation [9]. For more constrained scenarios, such as in medical settings, the less intrusive and privacy-preserving sensors such as passive RFID

or depth camera were used [55, 53, 6]. Because the sensor data collected in a real-world environments is noisy (may contain hardware noise, outliers, and missing data), machine learning algorithms were commonly used to establish the connection between sensor data and the process phase. On the other hand, the feature and classifier selection were often chosen arbitrarily or empirically [55, 52], which makes the shallow-modeled systems difficult to transfer. In addition, when processing multi-modal input data, the shallow-modeled systems usually make classification independently based on each input modality and combine them by voting [101]. In this way, potential correlations between different input modalities are ignored.

This limitation of shallow classifiers is not unique for process progress modeling; many fundamental computer science fields (such as image classification and speech recognition) face the same issue. In recent years, the CNN and LSTM [96, 33] have been successfully implemented in these fields and achieved significantly better performance compared with the state-of-the-art shallow-modeled solutions [48]. The CNN was widely used in image classification and image feature extraction, because the CNN with the learnable filters can automatically learn the representative spatial features from the raw training data and is proven generalizable (the pre-trained model can be used as the feature extractor) [23]. The current state-of-the-art process progress estimation systems use the pre-trained AlexNet [105] for process classification using video frames as input, but this pure CNN does not consider temporal associations between features in adjacent frames. To compensate for lost temporal associations that maintain the logical phase order, previous research attempted using a time window [99], HMM[105], or modified softmax layer [53]. However, the HMM does not scale well to complex process with a large number of phases, because the topology of HMM's transition matrix becomes less representative and requires manual tuning. The time-window approach prevents the system from working online. Based on the LSTM's ability to model temporal associations in speech recognition [25] and natural language understanding [112], we implemented the LSTM with CNN in our system to extract the spatio-temporal features. We then introduced a regression model with GMM to achieve both process completeness estimation and phase detection.

## 2.2 Passive RFID Based Activity Recognition

For low-level activity recognition different types of sensors are widely used. Due to their unique advantages (small, cheap and battery free), passive RFID tags have been used in applications where other sensors are not suitable or have failed. These applications include detection of human-object interaction [120, 120], people and object tracking [119] and more complex problems such as activity recognition. RFID was used for activity recognition in a kitchen setting [116], but only as secondary to a vision system because the received radio signal was subject to noise and interference caused by moving people and other objects. RFID was also used as the primary system for process-phase detection with wearable RFID antennas and other sensors [110]. The system was able to achieve satisfactory performance for phase recognition, but wearing the antennas requires user participation and may interfere with work in fast-paced medical settings. Recent research demonstrated that the status of object manipulation can be estimated using passive RFID tags and fixed antennas based on manufactured features extracted from received signal strength indicator (RSSI) [76]. The use of specialized objects to perform complex activities provided the basis for activity recognition [51]. Challenges remain because the recorded RFID data contain noise and variance due to environmental changes, such as people moving in the room. The noise and variance in RFID data compromise the representativeness of manufactured features and in turn impact activity recognition results. Similar challenges exist in other applications, such as image recognition and speech recognition, where large part of input data are inessential (e.g., redundant pixels, background noise), requiring the classifier to be insensitive to those variations. Earlier research tried to accomplish the complex tasks such as object recognition or activity recognition by using manufactured features, or building a hierarchical model with several layers of classifiers to extract low-level features for final decision making [5, 71]. The use of deep learning in recent years has led to great leaps in many fields, from image classification [2] to speech recognition [82]. Deep learning has revolutionized image classification and speech recognition. It is reasonable to expect similar success in pervasive computing [41]. Earlier research showed that deep learning

can be applied to data from mobile phone sensors or an accelerometer for recognition of person's simple physical activities [4, 12]. No system has yet been developed that combines deep learning and RFID for activity recognition during complex work, such as patient care, instead of simple physical activities like sitting or standing. We developed a deep learning system in a setting similar to one we previously studied [52] and achieved better performance on medical activity recognition compared with existing research.

## 2.3 Region-based Activity Recognition

Besides mobile sensors, the computer vision is also widely used for activity recognition. The computer vision based activity recognition approaches can be roughly divided into two classes: representation based and deep learning based [30]. Representation-based approaches rely on crafted features and descriptors that face generalization issues. The body skeleton and joints provided by the Kinect sensor [94] are often used for activity recognition as a representation of body posture [80]. Due to the variety of application environments, these features and skeletons may fail to generalize and lead to recognition errors.

Deep learning has been recently applied to activity recognition, initially using single images independently [121], that ignores the temporal associations of activities. Subsequent ConvNet research used approaches that learned the temporal associations by feeding in video frames stacked over a time window [38]. This approach could only model short-range temporal associations of activities in short video clips. Fusion-over-time strategies partially addressed this issue by stacking the features extracted from video frames and using slow fusion [38]. Recurrent neural networks and long-short-term memory networks (LSTMs) also were used for modeling long-range temporal associations. The ConvNet-LSTM structure was used for activity recognition with different types of input (RGB video, mobile sensor data) [56, 93, 103].

Most existing research treated the activity recognition as a classification problem performed within a black-box. It is difficult to tell whether this model learned the actual

activity related features or over-fitted to the background features. Previous research has focused on visualizing and understanding the trained ConvNet [115] and LSTM [37], and the learned feature maps demonstrated that the neural network learned the features of both activity and its surroundings. Generating the correct label on the testing set with limited testing data does not prove the model learned the activity; therefore, this system may not be generalizable. This problem is particularly noticeable for small-size datasets.

Our approach to activity recognition draws from several existing ideas. The region proposal method [121, 62] requires a pre-trained object recognition model for region generating (usually trained on ImageNet [90]). The regions containing the target (and possibly unrelated people or objects) are generated, and a secondary network takes the generated regions and their relative location for textual image description or activity recognition. Unlike this approach, we only generated the activity region for training the activity recognition model. Other research has relied on the use of trajectory information to distinguish people from background, and multiple descriptors for action detection [107, 58], or has used an LSTM autoencoder to help the supervised learning [98]. Because not all the people in the scene may be involved in the activity, we first generate masks that only outline the activity location or the performer's role in a team instead of simply segmenting the people from the background.

Our approach first mines additional information (activity location or team members' roles) and then does activity recognition based on both inputs and this additional information. We used a per-pixel mapping strategy for activity localization or team role labeling, similar to semantic segmentation that distinguishes foreground and background, or image-to-image mapping [34]. Image segmentation previously used fully-convolutional networks with convolution and deconvolution [62]. More recent state-of-the-art research used GAN-based generative structures [66, 24] for the image-to-image mapping [34] and segmentation [64]. Our work is novel in that we trained the system to find the activity location information from input data for activity recognition.

## 2.4 Attention based Activity Recognition

The research for activity recognition started with feature extraction from different sensor data; many features were proposed [13, 102]. However, most early activity recognition systems were neither scalable nor generalizable because the feature extraction and selection was often arbitrary and not transferable.

Deep learning, with the ability to automatically learn the features from raw input data and capacity to hold mil-lions of weights to generalize to large-scale datasets, has achieved state-of-the-art performance in many fields [57, 112]. Although the deep neural network is able to ex-tract many features, it is hard for system to determine which features to rely on for activity recognition. In some cases, only the activity performer provides useful information in a frame with multiple irrelevant people [65], while in other cases like cooking only hand gestures provide information [100]. To address this issue, some research proposed activity masks generated by a conditional GAN to assist ConvNets in activity recognition [35]. However, training a GAN mask generator requires a large number of ground truth masks, which are difficult to obtain. Another approach is to train an attention subnetwork that generates attention regions that highlighting the most important feature subset [112]. We modified the ConvNet attention module to make them work with both images and sensor sequential data while maintaining the ease of visualizing and tuning. The deep attention neural network can use different sensor modalities together for activity recognition [15, 87]. There are several proposed fusion strategies to make use of data from different sensors, including voting [11] and ensemble [73] methods. For deep learning, the most commonly used strategy is the multimodal network, which merges the features extracted by different network branches [26, 106]. As an extension, we propose modality attention on the fusion strategy to learn and score the importance of different branches.

Supervised training is commonly used for activity recognition tasks, but regular supervised learning tunes the weights associated with prediction and attention simultaneously. Unlike the GAN approach where the generated mask is penalized by a ground truth mask [11], there is no guide directly associated with attention map generation

in attention models [112]. As a result, the attention model might generate attentions that are not associated with activity. We propose an asynchronous tuning method that tunes the attention generating and recognition separately to ensure the model learns to recognize based on associated features. Another issue with activity recognition is temporal association modeling. As argued in [72], many activity clips contain time instances that are not directly associated with a certain activity. However, to use them for supervised frame-labeling training, these instances are often labeled with the same ground truth label as the entire video. The system then receives loss from these time instances where the activity has either not started yet or has already finished. This problem is hard to address with supervised learning if we want to make per-instance predictions, because we have to assign a label for each time instance during training. However, this is a common problem in the domain of reinforcement learning, e.g. training machine for game playing [68], where a machine is trained to make a control action for each time instance without using labels for each time in-stance. The reinforcement learning tunes the model based on accumulated rewards and penalty. We introduce the use of DQN for tuning instead of supervised learning. DQN is one type of reinforcement learning which has demonstrated proficiency in control systems, game playing, and many other problems [68, 67]. We modified the DQN for network tuning based on the accumulated prediction results so that the network does not receive additional penalty for time instances with no activity in progress.

# Chapter 3

# Process Progress Estimation

## 3.1 Overview of Chapter

A sequence of activities that achieves a certain goal represents a work process. For instance, the trauma resuscitation process consists of pre-arrival, patient-arrival, primary survey, secondary survey, post-secondary survey, and patient-leave phases; each phase consists of one or more activities. Several successful systems have been introduced for image classification [96] and activity recognition [55]. We take a further step in this analysis by developing a system for real-time process progress estimation. Online progress estimation has applications for many real-world problems, including automated human-computer interaction systems. For example, online detection of a sports process phase can be used to guide the camera on overhead drones for the video broadcasting. Online progress information for a medical process, such as the process completeness and remaining time, can help medical providers organize resources and schedule treatment timing. We designed and evaluated our sensor-based system to estimate the work progress in three ways:

1. **Process completeness,** indicating the percentage completion of the whole process.

2. **Process phase,** indicating a major stage of the process progress. A phase may contain one or more logically-grouped activities that achieve a larger goal.

3. **The remaining-time,** representing the estimated time left until the process finishes.

Work processes can be roughly categorized into two types: sequential and parallel.

Our current work does not model parallel processes. *Linear* sequential processes can be partitioned into a set of "phases" that occur one after another in a fixed order. For example, during the trauma resuscitation process or the sports event broadcasting process, the phases are rarely skipped or duplicated. *Nonlinear* sequential processes allow phases to be repeated or performed in an arbitrary order. For example, when making a salad, the mixing, vegetable chopping, and sauce preparation can be performed in any order before serving. We focused on characterizing linear processes.

Previous process-phase detectors can be roughly categorized into three types: manual, shallow-modeled, and deep-learning-based. The first type used manually-generated event logs or medical equipment signals [20, 75]. As these systems worked with relatively noise-free datasets, they achieved good performance. However, manual detectors require manual log generation or signals from specific medical equipment, making them hard to implement and generalize. The use of sensors addressed these two issues at a cost: sensor data may be hard to obtain (e.g., wearable sensors are not preferred in medical settings as they may interfere with the work), and are subject to the hardware or environmental noise. The second type of phase detectors used shallow classifiers to predict phases from noisy sensor data [20, 10, 83]. Yet, the features and model selection were often arbitrary and difficult to generalize, because different researchers using different features can only claim that those features worked best for their specific scenarios [55]. Due to its success in the image classification and speech recognition, deep learning and automatic feature extraction has given rise to the third type of phase detectors. All three approaches treated the process phase detection as a discrete classification problem, which overlooks the continuous nature of the processes and misses the associations between process phases and percentage completion. These limitations have caused systems to return logically impossible predictions [53].

We address these issues and estimate process completeness as a continuous variable. Our system is designed to work with sensor data, as opposed to manually-generated event logs. We designed a multimodal structure to make the system compatible with different types of input sources. Features in our model are automatically learned using a deep neural network. We introduce a deep regression model to continuously estimate

process completeness, and the rectified hyperbolic tangent (*rtanh*) activation function to bound the regression output values in the context of process completeness. The system detects the current process phase based on the currently estimated process completeness using a Gaussian mixture model (GMM). Our model for completeness estimation is trained on the regression error as well as a novel conditional loss from the phase prediction. Finally, the remaining time is estimated during run-time using the calculated process execution speed.

## 3.2 System Structure

Our system consists of four main parts (Fig. 3.1):

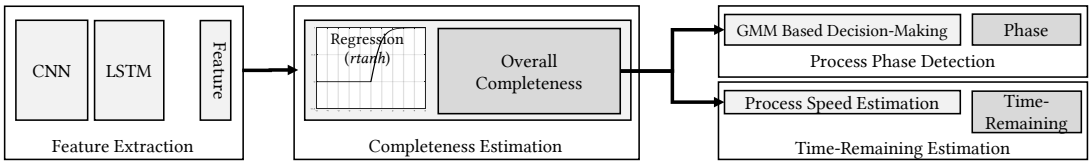


Figure 3.1: The overall system structure used for modeling process progress estimation.

**Feature extraction:** Because we used different sensors for data collection in different applications, the first step is the data representation and feature extraction. Instead of using manually crafted features, we used a CNN and LSTM based model to learn the spatio-temporal features from the input data [91]. We implemented a multimodal structure to fuse the features extracted from different sensor data.

**Completeness estimation:** The system has a deep regression model that directly produces a single regression output value. We introduced the *rtanh* activation function for bounding the neuron output to a valid range.

**Phase detection:** The system takes estimated completeness as input and uses a probabilistic GMM inference to detect the process phase. Phase detection provides a conditional loss function to help train the regression model.

**Remaining-time estimation:** Our system dynamically updates the estimation of remaining time based on the observed speed of process execution, which we defined as the rate at which one percent of the process is being accomplished.

### 3.2.1 Feature Extraction

Similar to the activity recognition [38, 55], our estimation of the process progress relies on both spatial and temporal features. The spatial features in a video frame define the activity at a time instance, while the temporal features from consecutive frames define a phase in the process. The learnable filters in CNNs are commonly used to extract the spatial features [117], and LSTMs are often used to model temporal dependencies of the sequential data [37].

Our trauma resuscitation dataset contains low-resolution depth images and audio from a Kinect [53]. We chose the Kinect depth sensor for our medical application because it is privacy-preserving (does not capture any facial details). Other types of sensors can be easily combined by introducing additional multimodal branches into our system. During every second, the video input branch was directly fed depth frames, while the audio branch was fed MFSC feature maps [3] for feature extraction. A CNN-LSTM structure then performed spatio-temporal feature extraction, where we used a pre-trained CNN structure (AlexNet [40]), followed by multiple LSTM layers. The features extracted from different sensors are subsequently combined in a fusion layer, following our previous implementation [53] (Fig. 3.2, top).

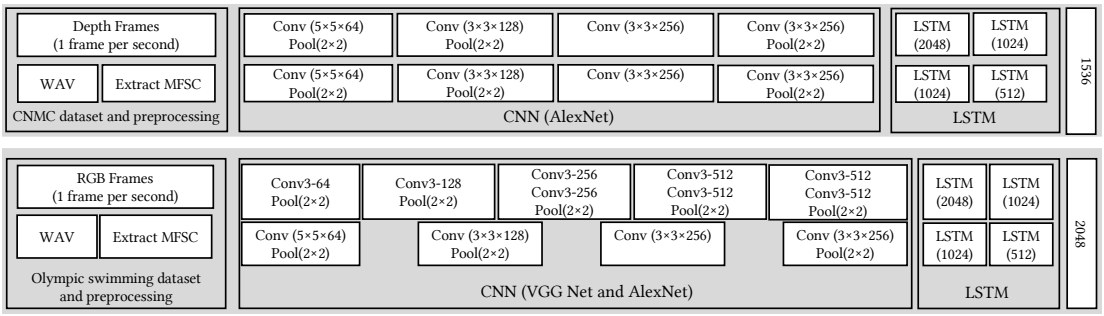


Figure 3.2: Feature extraction framework for the trauma resuscitation dataset (top) and the Olympic swimming dataset (bottom).

The Olympic swimming dataset was collected from YouTube videos recorded by different types of cameras (including cell phone cameras and professional cameras). This dataset has videos of different resolutions and audio recorded by cameras at different distances. Due to our hardware limitation (described in Section 3.5 in Section 3.5), we

downsampled the video frames from 25 fps to 10 fps and resized them to $256 \times 256px$. Instead of AlexNet, we used a deeper VGG Net (Fig. 3.2, bottom) with pre-trained weights [96] for image feature extraction (Fig. 3.2), because RGB images contain more textural details than depth images.

### 3.2.2 Process Regression and Completeness Estimation

Unlike image classification or single-image activity recognition where each sample corresponds to an individual image, process completeness is a continuous variable which cannot be estimated as a discrete classification problem. For this reason, we introduce a regression-fitting model that uses the extracted spatio-temporal features for process completeness estimation (Fig. 3.3, completeness estimation part).

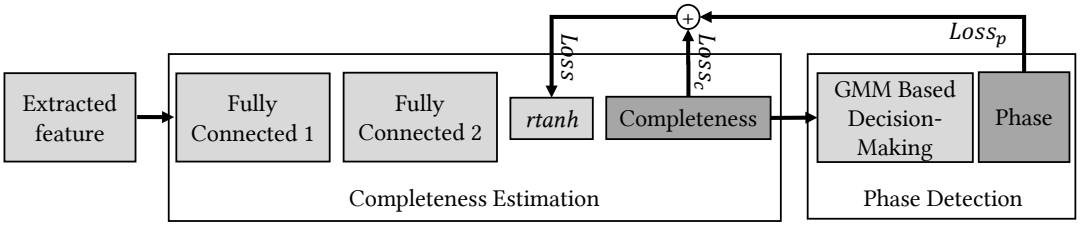


Figure 3.3: The system diagram for completeness estimation and phase prediction. The $Loss_c$ represents the loss from completeness regression error and the $Loss_p$ represents the loss from phase prediction error.

Our deep neural network extracts features from input data and performs regression using an activation function which generates the regression value $\hat{y} \in [0, 1]$. The activation of regression neuron can be expressed as:

$$\hat{y} = f(\boldsymbol{\omega}^T \boldsymbol{\phi} + b) \tag{3.1}$$

where $\boldsymbol{\omega}$ $(N \times 1)$ is the weight vector and $\boldsymbol{\phi}$ $(N \times 1)$ is the output of the last fully connected layer (Fig. 3.3, fully connected layer 2, with $N$ neurons in total), $b$ is the bias term and $f(\cdot)$ is the activation function of the output neuron. To model the process progress, the activation function must have the following properties: (1) the activation output should be zero if the process has not yet started (e.g. pre-arrival phase of the trauma resuscitation); (2) the regression value changes continuously from zero to one

as the process proceeds; and (3) the activation should be equal or close to one, with a proper strategy (e.g., thresholding) for when the process has been completed (e.g., after the patient left the trauma room).

We modified the *hyperbolic tangent function* for this purpose, and we named it the rectified hyperbolic tangent (*rtanh*) function:

$$rtanh(x) = \max(0, \tanh(x)) = \max(0, \frac{e^{2x} - 1}{e^{2x} + 1}) \tag{3.2}$$

This activation function returns zero for negative inputs and positive values up to one for positive inputs. This value range makes it suitable for returning a progress completeness percentage ranging from 0% to 100%. The *sigmoid* function has the same value range, but its range of positive values is distributed across all real numbers, making it slower to train. In addition, during the backpropagation, the gradient of *rtanh* activation can better prevent gradient vanishing than the *sigmoid* because the derivative of $tanh \leq 1$ and of $sigmoid \leq 0.25$ [32]. This property will lead to a faster training under the same setting of the learning rate. To compare the performance of the *rtanh* and *sigmoid* functions, we initialized the neural network parameters to the same value and trained the network using *rtanh* and *sigmoid* with the same training and testing split of the data. Our experiments showed that the two activations eventually led to similar accuracies, but the convergence time (the time it takes until accuracy stabilizes for 3 epochs) using *rtanh* was 30% shorter than that using *sigmoid*, because of *rtanh*'s steeper gradient. As we focused on modeling a linear process, the current completeness value should be increasing compared to the previous completeness value. We used an LSTM after the CNN to retain the estimated progress information and help the system with tracking the increasing completeness trend.

Similar to a classifier, a regression model can be trained using backpropagation. When generating the ground truth label, the completeness is labeled differently depending on the application. For example, the completeness of a trauma resuscitation equals zero before the patient entered the room and remains at one after the patient left. Given the time at which the data were collected relative to the beginning and end of a process enactment, we can label the data from a certain time instance with

the completeness range to which this time instance belonged. We divided the data into twenty 5%-segments, and labeled each segment with an associated completeness between 0% (process start) and 100% (process end), forming a stepwise function with 5% increments. The output of the *rtanh* neuron can be directly used as the regression result (Fig. 3.3, decision-making part).

To make the regression smooth as the completeness progresses for real world processes, we applied a Gaussian smooth filter after the *rtanh* neuron. The loss from completeness estimation error $Loss_c$ can be expressed as:

$$Loss_c(\theta = \{w, b\}, D) = \frac{\sum_{i=0}^{|D|} abs(R(\theta = \{w, b\}, D_i) - p_i)}{|D|} \tag{3.3}$$

where the $\theta = \{w, b\}$ denotes the model with parameter set $\theta$, and $D$ is the input dataset. $R(\theta = \{w, b\}, D_i)$ denotes the regression output from the model with parameter set $\theta$ on dataset $D$ at time instance $i$ as input. $p_i$ denotes the $i^{th}$ percentage label for regression in terms of process completion. We used the mean absolute error for the loss, which is the $abs(\cdot)$ term in the loss function. Although other error measurements such as mean square error could be used, the square error would inhibit training by making the losses even smaller (considering that $errors \in [0, 1]$).

### 3.2.3 Phase Detection and Conditional Loss

With the completeness estimated by the regression model, we need to predict the discrete phase based on the continuous completeness estimation results. A separate classification model using the same extracted features can be used for phase detection [53], but performing classification without considering the completeness ignores the temporal associations between the extracted process progress features.

Given that the processes we considered are linear, we found that the duration of each phase is similar to the Gaussian distribution. Therefore, we assumed a Gaussian distribution for the duration of each phase, which allows us to use a Gaussian mixture model (GMM) with one centroid per phase for phase prediction (Fig. 3.4, left & middle). On the other hand, this GMM framework does not generalize to nonlinear processes

with duplicate or rearrangeable phases (Fig. 3.4, right EndoVis dataset). Nonlinear processes require modifications in the GMM-based model as discussed in Section.

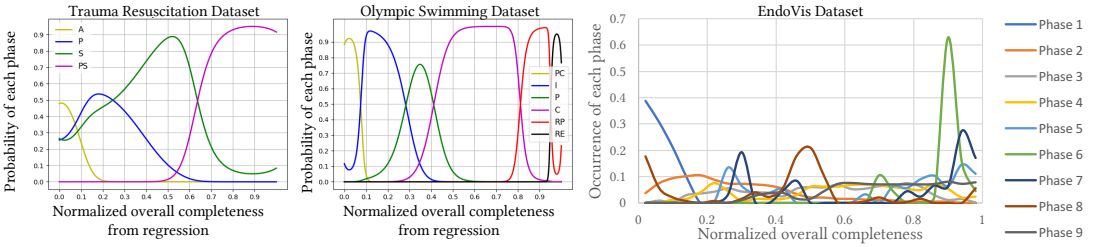


Figure 3.4: The probability for each phase plotted against the normalized process duration for the trauma resuscitation dataset (left) and the Olympic swimming dataset (middle) and EndoVis dataset (right) that contains nonlinear process.

Training the GMM [8] only requires the occurrence distribution for each phase on the normalized process duration scale, which can be obtained from the ground truth data. We can pre-train the GMM using completeness ground truth and establish the probabilistic association between the overall completeness and process phase. The phase recognition results can be calculated from:

$$\hat{p} = \text{argmax}_{1\leq k\leq K}\{\log(w_k) - \frac{1}{2}\log(\det(2\pi\boldsymbol{\Sigma}_k)) - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)\} \quad (3.4)$$

where $\hat{p}$ is the predicted phase. $\boldsymbol{x}$ is the completeness estimated by the regression model. $w_k$ is the weight for the $k^{th}$ Gaussian kernel. $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ denote the mean vector and the covariance matrix for the $k^{th}$ Gaussian kernel. $K$ is the total number of phases. $\text{argmax}_{1\leq k\leq K}\{\cdot\}$ denotes the function finding the index $k$ with the largest likelihood among all $K$ indices.

With equation 3.4 and the regression result, we can directly use the GMM for phase prediction (Fig. 3.4). We only applied the GMM for four out of the six phases in the trauma resuscitation dataset, removing the pre-arrival and patient-leave phases. This is because the duration of the pre-arrival phase depends on the transport time from the injury scene to the hospital, which was not recorded in the dataset; and the patient-leave phase is coded as an ending signal lasting only one second. Our GMM-based phase prediction approach relies only on the regression result, and the phase prediction error does not impact the backpropagation training process used to tune the regression.

The regression model can be trained based on both the completeness estimation error and the phase prediction error. Based on the assumption that the regression value is correct, the system generates no additional loss if the GMM-based phase detection result is correct. Otherwise, an additional loss is incurred, calculated as the distance from the regression result to the mean of occurrence distribution for the actual current phase. The loss from phase estimation error $Loss_p$ is conditional based on the phase detection result and can be expressed as:

$$Loss_p = \begin{cases} 0, & \hat{p} = p \\ |R(D_p) - \mu_p|, & \hat{p} \neq p \end{cases} \tag{3.5}$$

where the $\hat{p}$ denotes the GMM-predicted phase and $p$ is the actual current phase. $D_p$ is the input data for phase $p$ and $R(D_p)$ denotes the regression model output. The $\mu_p$ is the mean of the Gaussian distribution for the actual current phase $p$. By combining the loss from regression error and the classification error, the regression model can be tuned to make completeness estimations and phase predictions that obey a logical order of phases. During the training, we added weights $\alpha$ for $Loss_c$ and $\beta$ for $Loss_p$ in equation 3.6:

$$Loss = \alpha Loss_c + \beta Loss_p \tag{3.6}$$

When $\alpha = 1$ and $\beta = 0$, the model becomes a regressor trained only on the regression error. Alternatively, when $\alpha = 0$ and $\beta = 1$, the model becomes a classifier trained only on the classification error. Training the system with a larger $\alpha$ would cause the system to prioritize overall completeness regression instead of minimizing phase prediction error, and a larger $\beta$ would do the opposite. We determined $\alpha, \beta$ empirically by minimizing the completeness estimation and phase classification error based on a small subset of each dataset. We used 10 training cases from both datasets to determine the ratio of $\alpha$ vs. $\beta$. Fig. 3.5 shows the averaged completeness estimation error and process phase detection accuracy with F1-score on the Olympic swimming dataset. We selected $\alpha = 0.6$ and $\beta = 0.4$ which achieved the best balance between the completeness and phase detection.

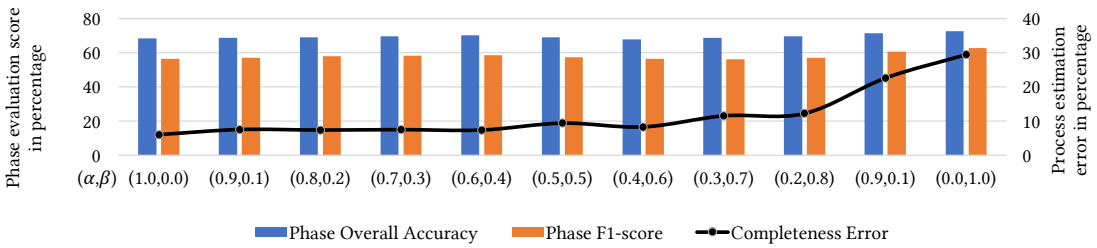


Figure 3.5: The process completeness estimation error and phase detection accuracy with F1-score using different $\alpha, \beta$ values.

### 3.2.4 The Remaining Time Estimation

The speed of process performance may change, which means that our remaining-time estimator requires a dynamically-updated estimation strategy. We estimated the remaining time by calculating the average speed of completing 1% of the process. If $\rho$ represents the current completeness (in percentage) and $\tau$ is the time elapsed since the start, the remaining time $t$ can be estimated as:

$$t = (\tau/\rho) \times (1 - \rho) \tag{3.7}$$

where $(\tau/\rho)$ denotes the time needed to make 1% overall progress and $(1-\rho)$ denotes the remaining percentage for the process to finish. During the runtime, we updated the remaining-time estimation for every second.

### 3.2.5 Implementation

We implemented our model with the *Keras* framework using *TensorFlow* backend. We implemented our *rtanh* activation and loss functions using the tensor interfaces of the *Keras*. As proposed in previous research [40], we used the rectified linear unit (ReLU) as our CNN activation function. For the trauma resuscitation dataset, we initialized and trained the model using a single GTX 1080 GPU. For the YouTube Olympic swimming dataset that has a larger input frame resolution and larger network structure, we used dual GTX 1080 GPUs for training. We used the weights trained in VGG Net [96] to initialize the VGG Net for RGB frame processing (Fig. 3.2, bottom). The Adam optimizer [39] was implemented with initial learning rate of 0.001 and a decay of $10^{-8}$.

We configured the system to stop automatically if the performance did not change for three consecutive epochs using the *Keras* callback function.

Due to the large model size, we adopted the dropout strategy [97] in the network to avoid overfitting. We also partitioned the training and testing sets by whole case instead of segmenting each case for evaluation, to minimize similarities between training and testing sets, as suggested in [53].

Unlike image classification models with a target in each training image, process phase classification may be significantly different from case to case (e.g., treating patients with different injuries or performing different swimming styles). Entering data into the classifier simultaneously causes slow convergence. For this reason, we used a training strategy: we initially fed only two process enactment cases into the classifier. When the system achieved a specified loss value (manually defined), we fed one more case into the classifier and kept feeding in new cases after each step until all cases have been used. Using this approach, the model learned the specific scenarios rapidly and later was able to discriminate between similar classes in other cases.

## 3.3 Experimental Results

### 3.3.1 Data Collection

**Trauma Resuscitation Dataset:** Our trauma resuscitation dataset was collected in a trauma room at the Children's National Medical Center in Washington D.C. Use of this data and its related research have been approved by the hospital's IRB. 150 trauma resuscitation cases were manually coded as the ground truth. The data was collected through a Kinect depth sensor mounted on the side wall of the trauma room [55, 53] (Fig. 3.6, left). Of the 150 trauma resuscitation cases, 50 cases had synchronized depth data and 35 from these 50 cases also had the synchronized audio data. We used the 150 coded cases to generate the GMM phase distributions (Fig. 3.4), and 35 cases with both depth video and audio data for model training and testing. Given the different patient conditions and times of the day, the durations of resuscitation phases varied (Fig. 3.6, right). The system recorded depth video at 1 fps to save storage space.

**Olympic Swimming Dataset:** The Olympic swimming dataset includes 60 videos from 2004 to 2016 downloaded from YouTube. The videos were recorded by different devices and at different angles and distances. We coded the ground truth for the six phases of swimming competitions manually. We manually edited some of the downloaded videos to ensure that all the video clips contained only swimming-related content. Because some videos did not record the pre-competition phase or result phase, these cases did not have all six phases.

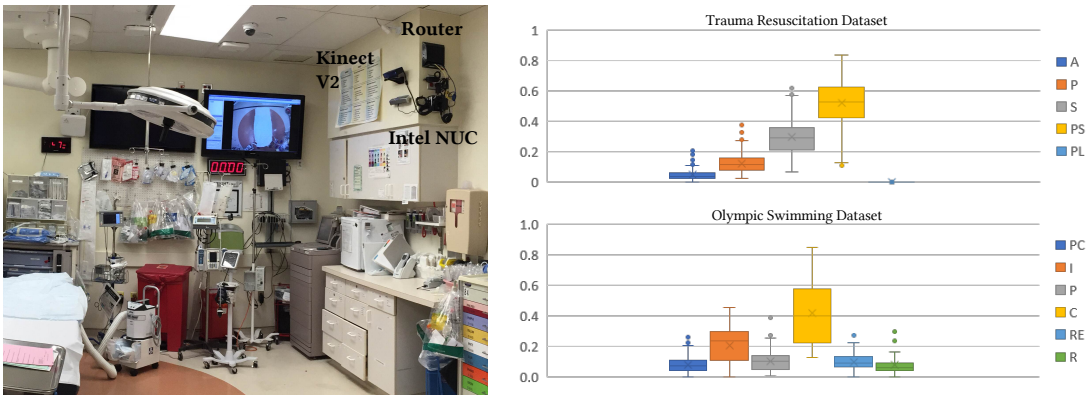


Figure 3.6: The trauma room with our Kinect installed (left) and the boxplot of phase duration in completeness percentage for trauma resuscitation dataset and Olympic swimming dataset (right).

### 3.3.2 Evaluation of Process Progress Estimation

We evaluated the proposed system with the trauma resuscitation dataset and the Olympic swimming dataset for estimating overall completeness, phase, and the remaining time. Because the dataset is imbalanced, we used the weighted average for all evaluations.

#### Completeness Estimation

We first calculated the overall completeness estimation error (Mean Absolute Error (MAE)) of the system by inputting 20% of cases into the trained network. Our system achieved an average 12.65% overall completeness error for trauma resuscitations and 6.32% error for Olympic swimming dataset. For the MAE of the completeness estimation with normalized process duration of testing cases (Fig. 3.7), a large MAE indicates

that the system had difficulty distinguishing certain scenarios and a large variance indicates that the system did not generalize well for the testing cases. We further evaluated the completeness estimation error on each process phase (Fig. 3.8), and showed that the system performed well in the starting and ending phases, which is due to the ability of our proposed *rtanh* function to hold regression output at zero (starting) and one (ending). The high error rate in the post-secondary phase may be related to the high similarity in appearance between the activities in secondary and post-secondary phases. Different types of sensors could be introduced to improve process progress estimation. The competition and replay phases of the Olympic swimming dataset had a higher error rate because replay phase is a slow-motion version of the competition phase and at low frame rates (e.g. 1 fps), the two phases are barely distinguishable. In addition, having higher-frame-rate videos containing more information can reduce overfitting [97]. To confirm this, we sampled the video at 1, 5 and 10 fps to retrain the system using the same setting of the parameters. The results showed that a higher frame rate significantly decreases completeness estimation error during the replay phase (Fig. 3.8). In the rest of this paper, we used 10 fps for the Olympic swimming dataset and 1 fps for the trauma resuscitation dataset, unless specified otherwise.

These results suggest that the performance of progress completeness estimation can be further improved by (1) Adding other sensors to provide more input features and better distinguish the phases with similar sensor data. (2) Using a higher frame rate for data preprocessing to retain more temporal associations and prevent overfitting. As a

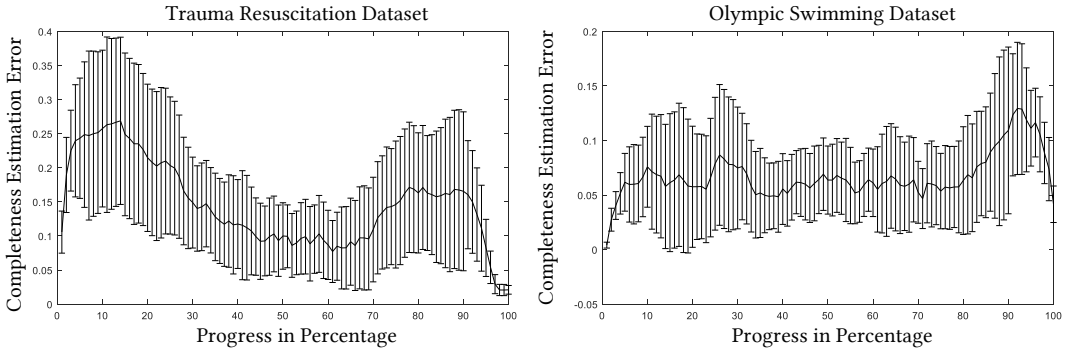


Figure 3.7: Mean (solid line) and Variance (shaded region) of completeness estimation error plotted against normalized duration of process enactment.

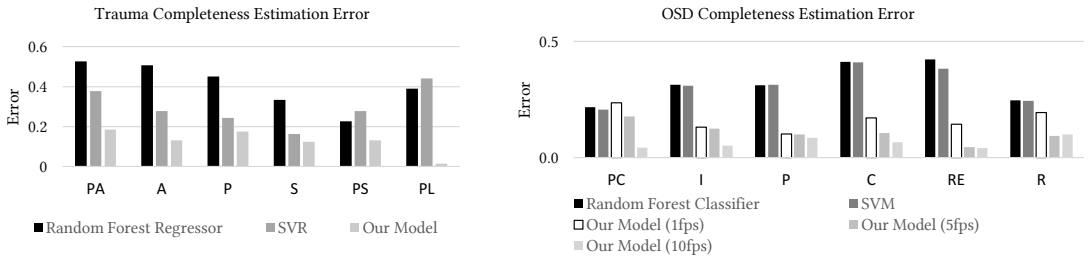


Figure 3.8: Aggregate error over different phases of resuscitation and swimming processes.

consequence, the computational resources and storage space cost would increase. Based on our experiments, a 5-minute video sampled at 10 fps with $256 \times 256px$ resolution takes more than 8GB of storage; (3) Using deeper models to extract more abstract features. As shown in computer vision research, using ResNet [29] instead of VGG may lead to better performance.

We performed additional experiments to confirm that our system structure is superior to the shallow models. We used the common shallow regressors of Support Vector Machine (SVM) and Random Forest and we trained them with the same raw video and audio data used for our deep model. Our deep model achieved significantly better accuracy than the shallow models, confirming that the CNN-LSTM extracts better multimodal features (Fig. 3.8). The shallow models faced difficulties extracting features from the raw input, and manually crafted features would be necessary to enhance their performance.

### Phase Detection

To evaluate the performance of process phase detection, we first generated the confusion matrices of the phase prediction results (Fig. 3.9). Our system achieved an average 86.06% phase prediction accuracy for trauma resuscitation data and 87.99% for the Olympic swimming data. Our analysis of the confusion matrices showed that our system is able to make logical phase predictions (Fig. 3.9). The zero values in most of the upper and lower triangles of the confusion matrices indicate that the system predictions rarely jumped between non-adjacent phases. This observation confirmed that our activation function helps maintain the estimated completeness range, and the

Confusion Matrix of Trauma Resuscitation Dataset

| | PA | A | P | S | PS | PL |
|---|---|---|---|---|---|---|
| PA | 95.8 | 4.2 | 0 | 0 | 0 | 0 |
| A | 24.1 | 43.2 | 32.6 | 0 | 0 | 0 |
| P | 0 | 12.8 | 70.9 | 16.3 | 0 | 0 |
| S | 0 | 0 | 36.4 | 63.6 | 0 | 0 |
| PS | 0 | 0 | 0 | 41.3 | 45.1 | 5.0 |
| PL | 0 | 0 | 0 | 0 | 6.9 | 93.1 |

Confusion Matrix of Olympic Swimming Dataset

| | PC | I | P | C | RE | R |
|---|---|---|---|---|---|---|
| PC | 77.8 | 20.6 | 1.6 | 0 | 0 | 0 |
| I | 9.7 | 76.2 | 14.2 | 0 | 0 | 0 |
| P | 4.7 | 19.4 | 50.2 | 25.7 | 0 | 0 |
| C | 0 | 3.9 | 14.1 | 81.2 | 0.7 | 0 |
| RE | 0 | 0 | 0 | 67.7 | 32.3 | 0 |
| R | 0 | 0 | 0 | 5.3 | 28.4 | 66.2 |

Figure 3.9: The confusion matrices for predicting phases of trauma resuscitation and swimming competition.

LSTM enforces an ascending overall completeness prediction. By further comparing the confusion matrices of both datasets, we found that: (1) The system accurately predicted the starting and ending phases, due to the *rtanh*'s ability to maintain the zeros and ones for the associated starting and ending phases. (2) The phase detection performance is associated with the regression performance. For example, the regression and phase detection systems both poorly predicted the post-secondary phase compared to other phases of trauma resuscitation (Fig. 3.8). This is because the GMM took the process completeness estimation results as the input for process phase detection, and the error made in the process completeness estimation was propagated to the process phase detection. (3) The system achieved a similar performance for both datasets, despite the differences in camera mobility. The trauma resuscitation dataset was recorded by a fixed camera, while the Olympic swimming dataset was recorded by moving cameras. The fixed camera captured the activity's scene and provided continuous activity sequence information, while the moving camera might capture some discontinuous unrelated data. Our approach achieved the similar performance under both scenarios, showing its potential for analyzing the temporal activity processes regardless of the video input's changing viewpoints, due to the ability of CNN-LSTM structure to learn and extract the representative features.

We then analyzed how our model outperformed the shallow models in learning the features automatically. Because process phases generally have different durations and

the amount of available data is imbalanced for different phases, we used three different metrics to perform a comprehensive evaluation. The F-measure (weighted average scores for precision, recall, and F-score) and 2SET metrics [109] were used to break down the wrongly classified instances into fragmentation, over-filling, and under-filling [109] (Table 3.1). Because the transition between process phases is instantaneous, the insertion, deletion, and merging metrics do not apply to our datasets [81]. To compensate for the F-measure's disregard for false positive samples, we used informedness, markdness, Matthews correlation coefficient (MCC) for a more comprehensive evaluation (Table 3.1).

Table 3.1: Phase prediction performance comparison using different modalities. The shallow classifier based results are shaded.

| **Input Data Source** | **Prec.** | **Rec.** | **F1-s.** | **Info.** | **Mark.** | **MCC** | **Frag.** | **Under** | **Over.** |
|---|---|---|---|---|---|---|---|---|---|
| Trauma audio | 0.33 | 0.37 | 0.17 | 0.09 | 0.06 | 0.07 | 0.01 | 0.13 | 0.13 |
| Trauma video | 0.63 | 0.57 | 0.51 | 0.35 | 0.43 | 0.39 | 0.01 | 0.10 | 0.15 |
| **Trauma audio & video** | **0.76** | **0.68** | **0.67** | **0.35** | **0.54** | **0.54** | **0.00** | **0.08** | **0.12** |
| Swim audio | 0.41 | 0.40 | 0.32 | 0.29 | 0.29 | 0.26 | 0.00 | 0.11 | 0.23 |
| Swim video | 0.58 | 0.55 | 0.48 | 0.37 | 0.43 | 0.39 | 0.00 | 0.09 | 0.24 |
| **Swim audio & video** | **0.67** | **0.70** | **0.58** | **0.52** | **0.53** | **0.48** | **0.00** | **0.06** | **0.12** |
| Trauma audio & video (SVM) | 0.09 | 0.37 | 0.09 | 0.07 | 0.03 | 0.07 | 0.03 | 0.01 | 0.13 |
| Trauma audio & video (SVM)* | 0.46 | 0.22 | 0.30 | 0.16 | 0.05 | 0.06 | 0.02 | 0.16 | 0.06 |
| Trauma audio & video (RF) | 0.17 | 0.25 | 0.16 | 0.01 | 0.06 | 0.01 | 0.08 | 0.17 | 0.01 |
| Trauma audio & video (RF)* | 0.44 | 0.29 | 0.34 | 0.12 | 0.08 | 0.03 | 0.02 | 0.16 | 0.02 |
| Swim audio & video (SVM) | 0.59 | 0.24 | 0.24 | 0.07 | 0.18 | 0.17 | 0.00 | 0.15 | 0.16 |
| Swim audio & video (SVM)* | 0.61 | 0.27 | 0.27 | 0.13 | 0.30 | 0.31 | 0.00 | 0.15 | 0.15 |
| Swim audio & video (RF) | 0.34 | 0.22 | 0.14 | 0.01 | 0.08 | 0.04 | 0.07 | 0.14 | 0.04 |
| Swim audio & video (RF)* | 0.46 | 0.30 | 0.21 | 0.03 | 0.14 | 0.07 | 0.00 | 0.16 | 0.19 |

*classifier takes the output of last fully connected layer as input.

As previously mentioned, shallow-modeled regressor generated significantly higher

progress completeness estimation errors (Fig. 3.8), which propagated to the phase detection. To focus on the classification comparison, we built shallow phase classifiers (SVM and Random Forest) using both raw input data and CNN-extracted features (Fig. 3.3, fully connected layer 2). The comparison of the metrics (Table 3.1) revealed that (1) Our deep model was able to achieve significantly higher precision and recall; most predictions were correct and most true instances were detected. (2) Our deep model achieved higher MCC score, indicating a positive association between the prediction and the ground truth. (3) Our deep model generated predictions with almost no fragmentation, while the predictions by the shallow classifier sometimes had fragmentations. Considering that there were gaps between the phases, the zero-fragmentation indicated that our system was able to make logically-ordered phase detection due to the use of LSTM to model temporal associations of features. Again, since we were calculating the 2SET scores using one-versus-rest method, the over-filling of one phase would result in under-filling of an adjacent phase. The over-filling and under-filling were similar to each other (not equal because we were calculating their weighted averages). Our deep model achieved lower under-filling and over-filling, demonstrating the ability to detect the phase transition points more accurately.

Since we used a multimodal structure, we further studied the impact of using single and both modalities. The results (Table 3.1) showed that our multimodal structure with all modalities included still outperformed same structure with single input modality; both video and audio positively contributed to the performance. We observed that the video-only system performed fairly well for both datasets, but the performance varied for the audio-only system. This discrepancy was caused by the difference in the audio quality. In the Olympic swimming dataset, the sound was clear and loud, while the trauma audio was often noisy and unclear. This was expected because the microphone array mounted on the side wall of the trauma room was far from the operation area (patient bed) and was close to an air vent. Noises from the hallway and air vent were sometimes louder than the medical equipment sounds and medical team's voices. In conclusion, using multiple input sources provided helpful information for process progress modeling, but only helped significantly if those inputs contained noticeable

and representative features.

Finally, our system performance was compared to that of domain experts. To give the experts the same information, we started the video clips from the beginning of the video and stopped at a random time instance before the end. Using this approach, the domain experts could not use video's progress bar to estimate the overall completeness. The comparison was made using depth data from 10 cases. Experts outperformed our system in the progress estimation when given high resolution (1080p) RGB videos (Table 3.2, Olympic swimming dataset). This outcome was expected, as the experts were the ones who generated the ground truth from RGB videos. Several other reasons contributed to the system's lower performance. The key reason is that our system did not use the state-of-the-art deep learning framework for image feature extraction; a deep ResNet might lead to better performance (we chose VGG-Net due to our limited computational power). In addition, the experts mentioned using extensive domain knowledge while making predictions. For example, in the trauma resuscitation dataset, the primary-survey phase should last less than 5 minutes. Training the system with both sensor data and domain knowledge could improve the system performance on process modeling in some domains.

In addition, our system achieved performance comparable to experts for some tasks under constrained conditions (Table 3.2, Resuscitation Dataset). For example, due to the privacy concerns, only depth cameras were allowed in the trauma room. Our results (Table 3.2) showed that experts were more accurate even on depth videos, but also significantly slower (30 times slower than the machine) at predicting process phases. Domain experts used specific indicators for process phase detection, such as certain resuscitation activities or indicative tool usage. Recognizing specific activities requires both experience and domain knowledge. This knowledge may be hard for a person to learn only from depth video and audio. It was also difficult for the domain experts to estimate overall completeness, and almost impossible for them to estimate the remaining time while our system was able to estimate both.

Table 3.2: The Comparison of Process Phase Prediction by Domain Experts and Our System.

| ***System*** | ***Avg. Accuracy*** | ***Avg. Precision*** | ***Avg. Recall*** | ***Avg. F1-Score*** | ***Avg. time/frame*** |
|---|---|---|---|---|---|
| *Resuscitation Expert* | 0.95 | 0.85 | 0.91 | 0.87 | ≈30s |
| *Resuscitation Machine* | 0.86 | 0.72 | 0.69 | 0.67 | <1s |
| *Olympic Swim Expert* | 1 | 1 | 1 | 1 | ≈20s |
| *Olympic Swim Machine* | 0.88 | 0.69 | 0.66 | 0.68 | <1s |

**The Remaining Time Estimation**

We also evaluated the average remaining-time estimation error, which was 7.5 minutes (14% of total duration) for the trauma resuscitation dataset and 2.2 minutes (18% of total duration) for the Olympic swimming dataset. Similar to the calculation of completeness error, we evaluated the remaining-time error by process phase (Fig. 3.10). We found that the remaining-time estimation error was not associated with phase detection error, but with completeness estimation error. A small remaining-time estimation error in an earlier stage may lead to a large estimation error in the subsequent phases, as the estimated process speed will be multiplied by the percentage left in the process. Comparing the remaining-time estimation error of the trauma resuscitation dataset with the Olympic swimming dataset, we found that the error was also associated with process length (Fig. 3.6, right); the longer processes had larger remaining-time estimation errors.

### 3.3.3 Comparison of Phase Prediction to Previous Work

Since we were the first to attempt the estimation of completeness and remaining-time, we only compared the performance of process phase detection with our previous research [53]. The same datasets and training-testing splits were used for this comparison (6-phase trauma resuscitation [53], 9-phase EndoTube dataset [105] and 8-phase TUM LapChole dataset [99]). Some evaluation scores were not reported in the original papers,

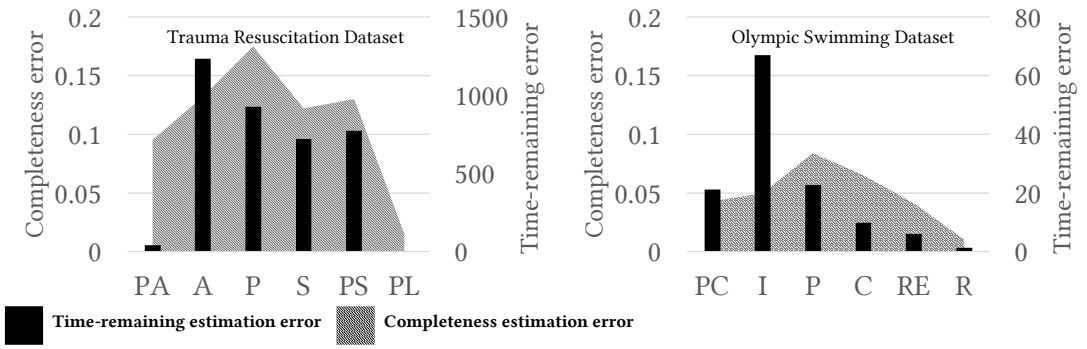


Figure 3.10: Estimated time error for different datasets.

e.g., the EndoNet handled first two phases as a single phase, and the averaging methods (average or weighted average) were not specified in the previous publications. Therefore, we reimplemented the models using the same training-testing splits [105, 99] to make the comparison fair.

Our current model achieved significantly better performance compared to our previous model [53] (Fig. 3.11). A major drawback of the previous work was its reliance on spatial information and the omission of temporal associations [53]. Our previous model [53] could make incorrect predictions that contradicted common sense, e.g., predicting the primary-survey phase before patient arrival. To address this issue, we previously proposed a constraint softmax layer. Because the constraint softmax requires information across the window centered at the current time instance, the decision making must wait for the future data. With such a limitation, our previous system could only be used for post-event analysis. Our current system instead relies on a regression curve designed to generate ascending values between $[0, 1]$, enforcing logically ordered predictions (i.e., patient arrival will happen before the primary-survey). Regression for completeness can be generated for every frame. Thus, our current system can work online. We further compared the phase prediction performance (Fig. 3.11). The results showed that our current system outperformed the existing systems (Table 3.3). Our current system maintained the similar F-score for pre-arrival and patient-leave phases, but significantly improved the scores for other phases.

We next compared our current system using the EndoTube dataset [105], and the

TUM LapChole dataset [99]. Since both datasets contain repeating workflows (repeatedly performed surgical phases), our GMM based model did not work well on such datasets and could not compete with the previous research. This confirmed that the GMM cannot model nonlinear processes, because a single Gaussian distribution does not fit well to repeated phases. We slightly modified our current system by using another LSTM model instead of GMM for process phase detection and achieved slightly better performance compared to the existing systems.

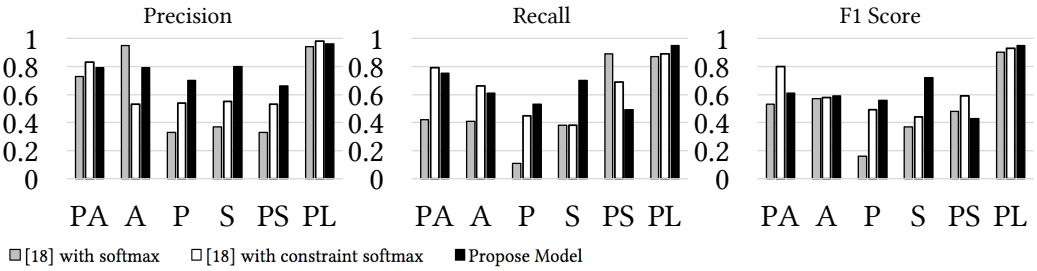


Figure 3.11: Performance comparison of our previous (classification with constraint softmax [53]) and current models.

We finally compared our system with existing phase-detection systems (Table 3.3). We were unable to reproduce their results because they didn't publish their implementation or datasets, so we used their reported results in this paper for comparison. Since these existing systems were not evaluated on the same dataset, the reported evaluation scores might not reflect the system performance in each setting but could provide a general understanding of phase detection performance using different methods. We noticed that the system with the best performance [20] could predict high-level phases only using manually generated low-level activity logs. This is a significant limitation for real-world applications: automatically generated low-level activity logs using sensor data might contain errors, and may significantly influence the system performance. Moreover, our system is general enough to work with all types of medical resuscitations. Previous systems for process phase detection were trained to handle datasets with specific injury types [75, 99, 105], but our current system is trained on cases with a variety of patient injury types, ages, and conditions. In terms of data collection, our system uses the economical, easily deployed, and commercially available depth camera with microphone array, while other early systems required human input, specific medical

equipment, or obtrusive wearable sensors.

## 3.4 Summary

We introduced a system for estimating the progress of complex processes. We provided two real-world datasets collected from different commercially available sensors and used several published datasets to evaluate our system. Our system outperformed existing systems on two datasets from linear processes and achieved the performance comparable to existing systems on datasets from nonlinear processes. Our system can be applied in many real-world applications, providing essential information for advanced human-computer interaction. We deployed this system in a hospital trauma room for online resuscitation progress estimation and discussed potential extensions of the system. The paper contributes to the community:

1. A regression-based neural network structure for process completeness estimation.
2. The GMM-based approach and conditional loss that can be used with the regression model for classification tasks.
3. The detailed system deployment instructions that can be used as a guideline to transfer the system to other fields.
4. The trauma resuscitation and Olympic swimming datasets will be published with ground truth labels, and can be used for future studies.
5. A hybrid model that uses deep regression with LSTM instead of GMM to model nonlinear process with repeating phases.

Table 3.3: Phase prediction performance comparison using different input modalities.

| **System** | **Gen*** | **Input Data** | **Model** | **Acc.** | **Prec.** | **Rec.** | **F1-s.** |
|---|---|---|---|---|---|---|---|
| Trauma Resuscitation Dataset | | | | | | | |
| Multimodal CNN [53] | Yes | Depth and audio | Multimodal CNN | 0.62 | 0.60 | 0.51 | 0.50 |
| Multimodal CNN [53] | Yes | Depth and audio | Multimodal CNN + constraint softmax | 0.80 | 0.66 | 0.64 | 0.63 |
| *Our model for linear processes* | *Yes* | *Depth and audio* | *CNN-LSTM Regression* | 0.86 | 0.72 | 0.69 | 0.67 |
| Endovis Dataset | | | | | | | |
| Endonet [105] | Yes | Endoscope video | CNN+HHMM | 0.63 | 0.59 | 0.61 | 0.59 |
| *Our model for linear processes* | *Yes* | *Endoscope video* | *CNN-LSTM Regression* | 0.66 | 0.55 | 0.61 | 0.51 |
| *Our model for nonlinear process^* | *Yes* | *Endoscope video* | *CNN-LSTM-LSTM* | 0.67 | 0.63 | 0.59 | 0.61 |
| TUM LapChole Dataset | | | | | | | |
| CNN based approach [99] | Yes | Laparoscopic video | AlexNet + time window | 0.72 | 0.61 | 0.66 | 0.63 |
| *Our model for linear processes* | *Yes* | *Laparoscopic video* | *CNN-LSTM Regression* | 0.78 | 0.52 | 0.61 | 0.56 |
| Olympic Swimming Dataset | | | | | | | |
| *Our model for linear processes* | *Yes* | *Video and audio* | *CNN-LSTM Regression* | 0.88 | 0.69 | 0.66 | 0.58 |
| Other Previous Research | | | | | | | |
| Phase detection from low-level activities [20] | No | Activity log | Decision Tree | n/a | 0.75 | 0.74 | 0.74 |
| Surgical phases detection [10] | No | Instrument signal | HMM | 0.83 | n/a | n/a | n/a |
| Phase recognition using mobile sensors [6] | No | Wearable sensor data | Decision Tree | 0.77 | n/a | n/a | n/a |
| Surgical workflow modeling [75] | No | Instrument signal | HMM | 0.84 | 0.85 | 0.84 | n/a |
| Food preparation activities recognition [100] | No | Video and accelerometer | Random Forest | n/a | 0.65 | 0.67 | n/a |

*Gen, If the system do not requires manually crafted feature as input.

# Chapter 4

# Sensor Based Activity Recognition

## 4.1 Overview of Chapter

Activity recognition in medical settings has been challenging due to potential interference by tracking devices, privacy concerns, and environmental limitations. Trauma resuscitation, the initial management of critically injured patients in the emergency department, is a complex process performed by a medical team under time pressure. Providing real-time decision support in trauma resuscitation requires strategies for tracking workflow and alerting teams to errors. Our work has focused on activity recognition as a key component in developing these strategies.

Previous work on activity recognition has used a range of techniques, including computer vision for identifying body posture, movement, and location related to different activities [48, 40]. Most previous approaches, however, are not practical in clinical settings. The use of RGB cameras can lead to privacy concerns, visual occlusion, and problems caused by variable illumination. Active wearable sensors require batteries and may hinder work. To address these limitations, we developed an activity recognition approach based on detecting object use. It relies on a common finding that an object or a combination of objects are related to specific activities (e.g., a thermometer is used only for measuring temperature). By tracking object use, our method allows activity recognition without cameras or wearable devices.

Our initial system accomplishes activity recognition purely relies on the passive RFID in two steps. First, the use status of different objects is determined based on the RFID information, such as signal strength. Second, activities are predicted based on the use status of objects. For object-use detection, we used small, inexpensive, battery-free passive RFID tags attached to medical objects and fixed reader antennas.

We placed tags on 10 object types commonly used in trauma resuscitation. Data from these tags were collected by eight RFID-reader antennas installed in the trauma room in the emergency department of a trauma center. By reviewing videos, medical experts of our team coded object-use data and a synchronized medical activity log from trauma resuscitations. We used these data to build our activity recognition model.

RFID based systems, however, have failed to achieve high accuracy of activity recognition in fast-paced and crowded environments. Two key challenges for RFID-based activity recognition are: the noise in received signal strength (RSS) that cannot be filtered out, and the absence of a direct link between the raw RSS values and human activity—an abstract concept. Similar challenges exist in computer vision for large-scale image classification and in speech recognition for voice-to-text conversion in noisy environments. Deep learning [48] introduced in those fields has achieved high levels of performance [48, 38, 2]. The main difference between deep learning and traditional machine learning algorithms is that instead of manual feature selection and defining the rules for making correct predictions, deep learning is able to learn the "right" features from large datasets and use them for this purpose. Consistent with the views of others [41], we believe that deep learning has the potential to be successful for mobile sensing. In this paper, we apply deep learning to the problem of activity recognition in a fast-paced real-world environment using only passive RFID.

We present a deep-learning architecture that uses only RFID data for detection of process phases and activities during trauma resuscitation. The resuscitation process has five consecutive phases: pre-arrival, patient arrival, primary survey, secondary survey, and post-secondary survey. Each phase consists of several activities—the specific low-level tasks performed by care providers that may or may not use medical objects. We define an activity as the interval during which one or more objects are used explicitly for patient care, which excludes the preparatory or cleanup manipulation of these objects [52]. We chose trauma resuscitation as our application domain for two reasons. First, this complex work setting is prone to errors and inefficiencies and is in need of decision support. Activity recognition is an essential building block to enable the development of this type of system. Using computer vision is not preferred for privacy concerns

and active wearable sensors are not feasible because the user must remember to wear them, they may interfere with work, and they require maintenance, such as battery charging. From a research perspective, RFID-based activity recognition has treated activity recognition as a binary classification problem where a specialized classifier decides whether or not an activity of a particular type is occurring. These types of systems, however, may not be scalable to a large number of activities. In addition, the common approach for activity recognition involves two steps: first detect the use of objects associated with specific activities by detecting human-to-object-interaction from sensor data, and then recognize activities based on used objects [52]. The predication errors made by the system in the first step will be cascaded into second step and impair the final prediction result.

Our approach for activity recognition uses passive RFID sensing. The RFID tags need to be strategically placed on objects of interest. Various features have been proposed and classifiers tested for RFID systems in different application settings [52, 79], which makes it unfeasible to compare their relative efficiency. As a result, feature and classifier selection for RFID data is often arbitrary. Our research demonstrates a novel way for activity recognition from RFID data without using manufactured features. To perform process-phase detection and activity recognition from RFID data, we treated the process-phase and activity recognition as a multi-class classification problem instead of extracting manufactured features and cascading object-use detection with activity prediction.

## 4.2 RFID-Based Medical Activity Recognition

### 4.2.1 Methodology

#### System Structure

Our work started with a two-step approach to medical activity recognition. We first detected when objects were in use based on RFID data and then made activity predictions based on the type of object manipulation for all 10 objects. To test the system performance, we semi-randomly selected 70% and 30% as training and testing data,

and repeated the experiments 10 times with different seeds to achieve random selection. After the classifier model was trained, activity-recognition predictions were made as follows (Fig. 4.1):

1. The RFID system recorded RFID data from 8 antennas installed in the trauma room (Fig. 4.1 (a)).

2. Every second the system extracted 6 features based on RFID data and generated a feature vector as described in Section III (Fig. 4.1 (b)).

3. The features were used as inputs in the classifiers for object-use prediction (Fig. 4.1 (c)). The object-use prediction results were used as input for activity-recognition classifiers (Fig. 4.1 (d)).

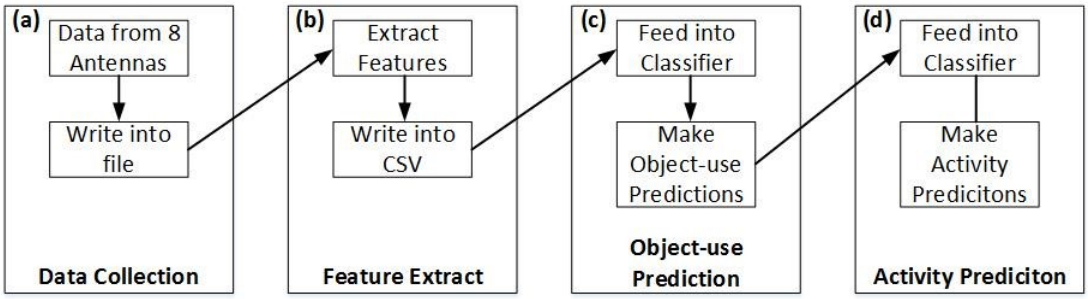


Figure 4.1: Activity recognition system diagram. (a) RFID system data collection from 8 antennas installed in the trauma room. (b) Six types of features are extracted from RFID data and feature vectors are generated. (c) Object-use detection based on extracted features. (d) Activity recognition classification based on object-use detection results.

### Tagging Strategy

We collected RFID data in a trauma room using two Alien 9900+ readers (with 4 ports) and 8 Alien ALR-8696 antennas (Fig. 4.2). The RFID readers were mounted inside the ceiling. Antennas 1 to 7 were mounted on the ceiling, facing down, and Antenna 8 was mounted on the wall, facing downwards at a 45-degree angle. This arrangement ensured that all equipment were covered by at least two antennas. The RFID readers and the computer were connected via a router mounted inside the ceiling. To avoid interference

among nearby antennas, we configured the system to activate a pair of antennas on the opposite sides of the room for 1-second each. Because continuous use of the RFID readers caused the hardware to overheat, an automatic approach was needed to start the readers only when needed. To address this issue, we installed a Honeywell AUROR passive infrared sensor (PIR) (Fig. 4.2) to monitor movement in the trauma room. If motion is detected, the PIR sensor signals the RFID system to start recording and stop recording after no motion is detected.

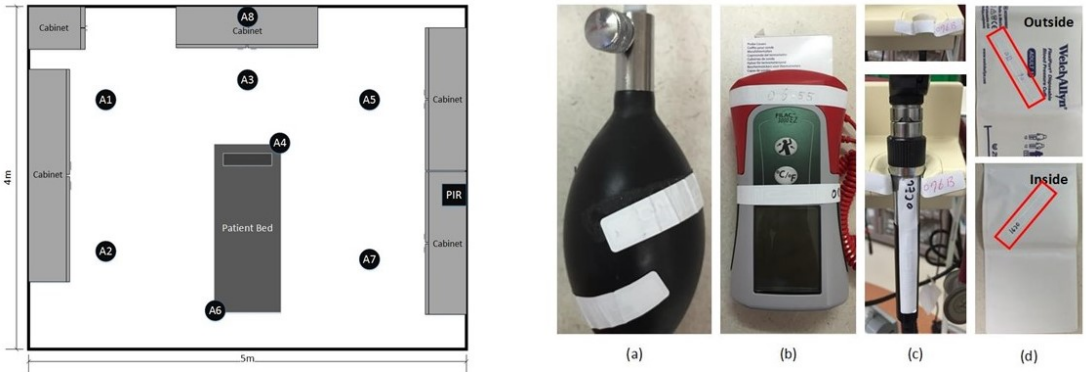


Figure 4.2: Left: The antenna configuration in trauma room we used for data collection. Antennas 1 to 7 are mounted on the ceiling and facing down; antenna 8 is mounted on the wall and facing 45 degrees to the ground. Right: (a) Direct tagging on BP Bulb. (B) Multi-tag tagging for thermometer. (c) Holder-slot tagging for otoscope. (d) Differential tagging for BP cuff.

Both RFID readers were set to collect data with maximum power and sensitivity. The collected data were written into a file using the format: [timestamp, reader IP, reader port, RSSI] Each file name was based on timestamp when the RFID reader was activated to allow synchronization between the recorded RFID data and the ground-truth data. The ground truth was manually coded by medical experts using videos of trauma resuscitations. We performed two types of ground-truth coding for each case. The first type was object-use ground truth, which contained information about the start and end time of object use, and the object name. We also noted whether the object was in actual use or manipulated (i.e., being relocated), and if manipulation was related to medical task [52]. The second type was activity ground truth, which focused on medical activities performed by trauma team members. We recorded the activity name, start time, and end time for each activity, as well as the objects used for that

activity, if any [52].

**Feature Extraction**

We used a 4-second time window ($[(n-3):n]seconds$) for feature extraction, as the period for antenna switching is 4 secs (each reader has 4 ports and each port is set to record data for 1 sec). A feature vector was generated every second based on a 4-second time window. Feature selection is critical for accurate classification. We selected 14 features to extract from RFID data [77, 78, 18, 120]. Our hardware was not able to provide other useful data, such as the phase angle and Doppler frequency shift, so the related features could not be used. Not all features worked well for all 10 objects. We ran the permutation feature importance calculation provided by the Azure platform for each object and used both precision and recall as the evaluation metrics [5]. We then chose these six features with the highest importance score:

1 **Peak RSSI:** The RSSI value is the most common feature for RFID-based systems. When an object is in use, we can expect a significant change in RSSI values due to different tagging strategies (Fig. 4.3(a)): (i) the RSSI drops for direct tagging and multi-tag tagging strategies, (ii) it increases for holder-slot tagging; and (iii) the difference between RSSI from inner and outer tags increases for the differential tagging strategy. We used "peak RSSI" in each time window as an RSSI feature, rather than the "average RSSI" value feature because, a tag is always closer to some and remoter to other antennas in a room with 8 reader antennas. A tag may be too far from some antennas and the RSSI values measured by those antennas may be outliers, thereby falsely bringing down the average. For objects with multiple tags, we collected the peak RSSI of each tag and averaged them as one of the features.

2 **Time:** Time is another important and often underused feature for making object use predictions. Medical personnel usually follow a routine sequence of procedures for a given case and the use of many objects often falls into certain time windows. We evaluated the discriminative power of the time feature by analyzing the RSSI

recordings from 10 objects during 38 trauma resuscitations. Based on manual video review, we plotted the distribution of object use over time for 10 objects (Fig. 4.3(b)). The black color at each time point denotes that the object was used at that time in more than 50% of cases, while gray color denotes the use in less than 50% of cases. The timelines in all 38 cases were synchronized with patient arrival time as time zero. Our results support the assumption that use of objects follows certain time distributions, which can help us predict their use and associated activities.

3 **Visible Antenna Combination:** The visible antenna combination isa set of antennas that can identify a tag in a particular time window. We implemented the "zoning positioning" method [18] to use the visible antenna combination as a feature roughly representing tag position. Objects at different locations in the trauma room can be represented by different visible antenna combinations. Our experiments showed that the objects used at the right side of the patient bed are more likely to be detected by antennas 4,5,7, and 8, while the objects used at the left side are more likely to be detected by antennas 1,2,6, and 8 (Fig.4.1 ). These results confirm our assumption that objects used at different positions will be detected by different antenna combinations. In our experiments, we used an eight-digit binary number to represent 8 antennas installed in the trauma room. If a tag was visible to antenna i, we placed a "1" at the i th digit of the binary number a value of "0" if not. Finally, we converted the binary numbers into decimal numbers and used them to represent the visible-antenna-combination feature.

4 **Spearman Rank Correlation Coefficient (SRCC):** The Spearman Rank correlation coefficient is defined as the linear correlation coefficient of the ranks. We divided the RSSI data recorded in each 4-second time window into a two-second left window (L) and a two-second right window (R). Because the reading rate changes rapidly, we interpolated the data in L and R to ensure that they had the same length w. Our hypothesis was that when an object is not used and

is stationary, the RSSI data in the L and R windows should be similar. When the object is in use, the tag will be occluded by a hand and the signal partially reflected or absorbed by the human body. As a result, the RSSI in the left and right windows for an object in use should not correlate well with each other. If we use Li and Ri to denote the i th RSSI value in left and right windows, the SRCC was calculated as [82]:

$$\rho = 1 - \frac{6 \cdot \sum_{i=1}^{w} (L_i - R_i)}{w \cdot (w^2 - 1)} \tag{4.1}$$

5 **Entropy:** Entropy is a measure of uncertainty and has been used as a feature for RSSI-based classification [18]. When an object is not in use, the entropy will be small due to small variance. When an object is in use, the uncertainty of the data grows. We start by dividing the RSSI range into N bins with equal bin length or BL (BL = 100 in this paper); for each object at each time window, RSSImax and RSSImin denote the maximum and minimum RSSI values, and the total number of bins, N = [(RSSImax−RSSImin)/BL]. Using the number of RSSI values in the i th bin, xi, we estimated the probability of RSSI values in the i th RSSI interval as $p_i = x_i / \sum_{i=1}^{N} x_i$. The discrete entropy was calculated as:

$$Entropy = -\sum_{i=1}^{N} (p_i \cdot log(1 - p_i)) \tag{4.2}$$

We calculated the entropy of RSSI for different objects using data from actual resuscitations. The calculation results confirmed our hypothesis that the RF signal is more randomly distributed when people are present or use objects for work. The entropy becomes higher when object is in use (Fig. 4.3).

6 **Dominant Frequency:** The dominant frequency of sensor signal has been previously used for activity classification of data from wearable devices [120]. We hypothesized that the dominant frequency of received RSSI data will be low for stationary objects as the RSSI values for motionless tags that deviate less than those of tags in use or in motion. When an object is in use, the RSSI is unstable, leading to higher dominant frequency because of tag movement and signal occlusion by the person holding the object (Fig. 4.3(c)).

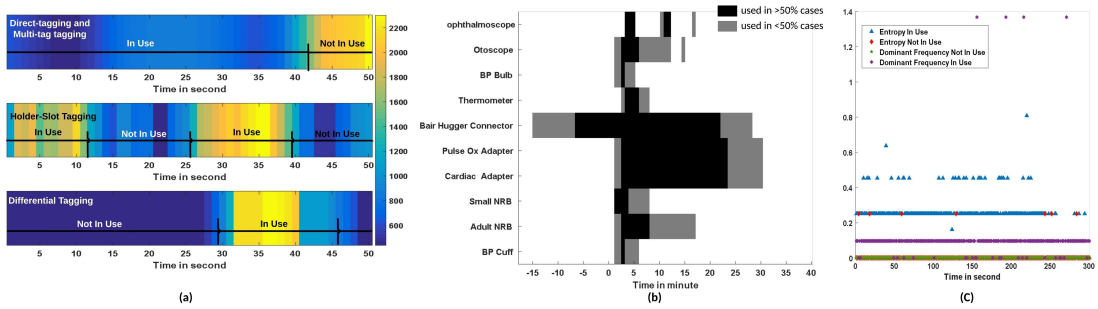


Figure 4.3: (a) Heat map of RSSI features for different tagging strategies. The RSSI decrease when the object with direct-tagging and multi-tag tagging is in use, the RSSI difference between inner and outer tag increases when the object with differential tagging strategy is in use. (b) A heat map for object-use time distribution for 10 objects used in this paper. Darker color means that the object was used in more than half of the resuscitations and lighter color means that it was used in less than half of the resuscitations. (c) The entropy and dominant frequency features for scenarios when object is in use and not in use.

For object-use detection, we constructed a feature vector for each of the 10 objects based on these two observations: 1. In the trauma room, all objects are clustered in a small area. When people use an object, their body may interfere with tag signals from other objects. The changes in signal strength of other objects may help with object-use detection. 2. Some activities are performed using more than one object, e.g., measuring blood pressure requires blood-pressure cuff and bulb. Using RFID data from several objects allows better detection of individual objects during multi-object activities. For each of the 10 objects, a feature vector consisted of the six features, which were then used for object-use detection:

RFID Feature = [Time, $Peak\_RSSI_{obj_{(1:obj10)}}$, $Visiable\_Antenna\_Combination_{obj1:obj10}$, $SRCC_{obj1:obj10}$, $Entropy_obj1 : obj10$, $Dominant\_Frequency_{obj1:obj10}$]

### Object-Use Estimation

We treated the RFID-based object-use detection as a binary classification problem. We used the Microsoft Azure Machine Learning toolbox in our experiments [7]. To train the model, we used more than 20,000 seconds of raw RSSI data recorded in 10 actual trauma resuscitations. Medical objects were used only for a fraction of time relative to the entire resuscitation process (Table 4.1). As a result, instances of object use versus

not-use are not equally represented in the RSSI data. Randomly selecting 70% of these data would not ensure sufficient number of examples for model training when objects were used. By referring to the ground-truth data, we randomly selected 70% of the data when an object was in use and 70% of the data when an object was not used. We then extracted features from these training subsets and used the remaining 30% of the data to test the model. Using this semi-random selection, we ensured balanced representation of both in-use and not-in-use classes in the training data.

Table 4.1: The object-use time, manipulation time and the breakdown of the manipulation time: In use, task-related motion, and unrelated motion.

| ***Object*** | ***In-use time (seconds)*** | ***Manipulation time, percent of total time*** | ***In-use time, percent of manipulation time*** | ***Task-related motion, % of manipulation time*** | ***Task-unrelated motion, % of manipulation time*** |
|---|---|---|---|---|---|
| *Ophthalmoscope* | 243 | 1.17% | 21.53% | 5.74% | 72.73% |
| *Otoscope* | 1294 | 6.24% | 29.33% | 8.85% | 61.82% |
| *BP Bulb* | 3771 | 18.17% | 4.90% | 2.17% | 92.93% |
| *Pulse Ox Adapter* | 11904 | 57.37% | 74.74% | 1.40% | 24.14% |
| *Cardiac Monitoring Adapter* | 12517 | 60.32% | 78.74% | 1.16% | 20.10% |
| *Small NRB* | 1315 | 6.33% | 41.89% | 2.72% | 55.39% |
| *Adult NRB* | 10 | ≈0% | 40% | 5% | 55% |
| *BP Cuff* | 1038 | 5.00% | 24.47% | 9.04% | 66.49% |
| *Bair Hugger Connector* | 11107 | 53.53% | 97.38% | 0.34% | 2.28% |
| *Thermometer* | 416 | 2.00% | 50.40% | 8.80% | 40.80% |

### Activity Recognition

Prior research has suggested that object-use status can be used for activity recognition, by assuming that an activity is performed when a related object is detected as used. This assumption, however, does not always hold true during resuscitation. When we reviewed the recorded resuscitations for ground truth data, we found that a person might manipulate an object without performing the actual task. For example, people

may relocate an object or prepare it for future work activity. People may also hold objects for extended time without meaningful interaction. For these reasons, we divided the types of human-object interaction into three categories (Table 4.1):

1 Object in use: The object is used for its intended work purpose of performing a specific task.

2 Object in task-related motion: The object is relocated from its storage place or prepared for future use.

3 Object in task-unrelated motion: The object manipulation is not related to any task, e.g., fiddling with the object.

We also found that several objects could be used together to complete a task, such as blood-pressure cuff and bulb to measure blood pressure. At the same time, other people may use other objects for different tasks or interact with objects without performing any task. It is unlikely, however, that relocation and accidental manipulation will follow any regular pattern for task-related use of multiple objects. To detect when an object is used for task performance, we use the combination of use-status of related objects. Manipulation of objects that were not related to tasks would appear as "noise" that needs to be addressed. We treated activity recognition as a classification problem in which object-use detection is indicative of activity performance. The object-use status (in-use vs. not-in-use) of all 10 object types was treated as features, and classifiers made activity recognition predictions. We generated a feature vector every second based on object manipulation type as follows:

$$Object_Feature_Vector = [obj1, obj2, \ldots, obj10] \tag{4.3}$$

where $obj_i$ denotes the type of manipulation for i th object. We used obji =1 for objects in-use and obji =0 for not-in-use. To train the activity recognition model, we used the same train-test split of data as before for training the object-use detection model. This method ensured that the testing data for object-use detection was not used as training data for activity recognition, reducing the likelihood of over-fitting. We used the object-use status of all 10 object types as a feature for activity recognition, and

chose a commonly-used Random Forest as our classifier [77]. We did not use HMM, which is also commonly used for a similar purpose, because we did not have enough data to train the transition matrix. Microsoft Azure cloud computing platform was used for classification. A classifier was trained for each activity.

### 4.2.2 Experimental Results

Object-use detection results served as the input for activity recognition. We first applied the object-use ground truth of 70% data to train the activity-recognition model. Each activity-recognition task was treated as a binary classification. We repeated the training and testing phase 10 times with different data splits and manually tuned the decision threshold for activity prediction. We used the same evaluation metrics as for the object-use detection (Fig. 4.4).

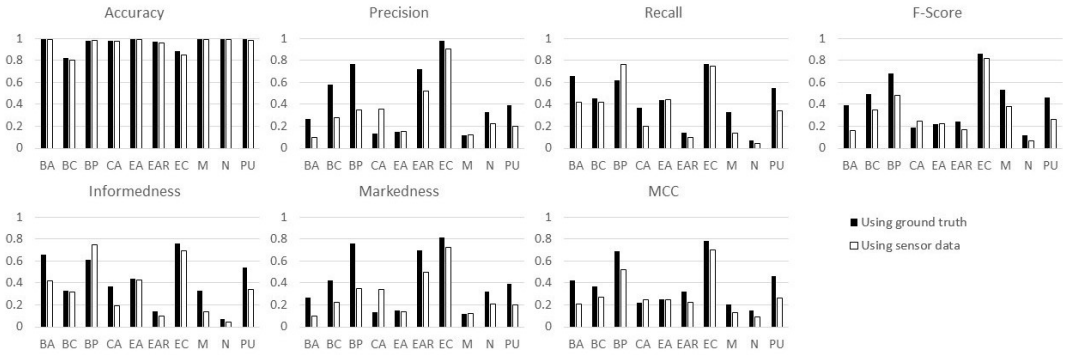


Figure 4.4: Activity recognition evaluation results using ground truth as input and using object use detection from sensor data as input.

Because the activity-recognition stage follows the object-use detection stage, any errors in object-use detection will propagate to activity recognition. To evaluate performance of activity recognition with erroneous input data, we compared our activity recognition using both ground truth ("perfect input") data about object use and predictions from the use-detection stage ("erring input"). The comparison showed a decrease of roughly 20% in F-Score, Informedness and Markedness for activity recognition when applying object-use predictions as input, instead of object-use ground truth. Only a few studies have addressed the challenge of activity recognition in medical settings, limiting comparison of our work with prior work. One study achieved an accuracy of

82.8% for process-phase detection with wearable sensors [6]. Our system achieved comparable performance with fixed antennas and required no wearable sensors. Another study achieved relatively good prediction results in recognizing four phases of the process using manually generated "low-level activity" records [20]. Unlike predictions in that study, our predictions were based only on RFID data and did not require human input. In addition, instead of predicting a few high-level phases of the process, we predicted activities within the phases, which is more challenging and can be more useful for real-world applications such as workflow tracking.

## 4.3 Deep Learning for RFID-Based Activity Recognition

### 4.3.1 Data Processing

#### Automated RFID Recording

We installed the hardware for RFID data collection and system activation control in an actual trauma room. The RFID data were collected with two Impinj R420 (8 ports) readers, set to record RFID data in maxmiller mode and dual target search mode. Because trauma events occur without warning, we could not keep the system continuously recording. We developed a fully automated system that is activated at the start of each resuscitation and keeps recording RFID data from all tags while the resuscitation is in progress. We set up a Kinect V2 sensor to monitor the number of people in the room. The RFID system will be activated to record data when more than two people are in the room and stops when no people are in the room (Fig. 4.5). To recognize 10 medical activities (Table 4.2), we tagged 11 types of medical objects following existing tagging strategies [52]. Because the blood-pressure (BP) cuff was tagged on the inside and outside, we counted it as two different object types, resulting in a total of 12 types of medical objects. The system recorded the received signal strength (RSS) from tags during 16 actual trauma resuscitations in this format: [Timestamp, Tag ID, RSS, Reader Name, Port Number]. Attributes of RFID signal other than RSS, such as Doppler shift and phase angle, have been used for human-object interaction detection or people tracking [114, 27]. Our experience and that of others

[27] has shown that Doppler shift measured by Inpinj R420 reader API is not accurate enough for our purposes. The U.S. government regulation requires that RFID readers perform frequency hopping, which affects the phase-angle measurements. Based on our experiments, the phase angle measured by the Impinj reader will have around 2.68 rad standard deviation for a stationary tag, which makes it unsuitable for classification.


Figure 4.5: Left: Antennas 1 to 7 are mounted on the ceiling and facing down; Antenna 8 is mounted on the wall and facing 45° to the ground. Middle: A photo of the room with the antennas labeled with blue rectangles and the Kinect and Mini PC labeled with a red rectangle. Right: Zoom-in of the Kinect, router, and Mini PC.

Table 4.2: Activities used in this paper and their medical code.

| ***Activity*** | ***Code*** | ***Activity*** | ***Code*** |
|---|---|---|---|
| Pulse Ox Placement | BA | Ear Exam | EAR |
| Oxygen Preparation | BC | Warm Sheet | EC |
| Blood Pressure Measurement | BP | Mouth Exam | M |
| Cardiac Lead Placement | CA | Nose Exam | N |
| Temperature Measurement | EA | Pupils Exam | PU |

### Pre-processing

Because of multiple instances of tracked objects and variable readout success rate, the recorded data needed to be preprocessed. We preprocessed the data in three steps:

1 Object name lookup: Many objects of the same type may be in the monitored area, such as four thermometers in our trauma room. We tagged all instances of an object type to ensure that all the objects used during trauma resuscitations

are tracked, which resulted in about 50 tags in the trauma room. Not all the tags were visible to antennas all the time, because some objects were kept inside a cabinet or shelf. We maintained a lookup table mapping tag IDs to the tagged objects. Before further processing the data, we replaced each tag ID with the name of its associated object type. All instances of the same object type were given the same object name, so that each RSS data entry represented one of 12 object types. The RSS data from multiple instances of the same object were combined during averaging in the following step.

2 Regularization of RSS data: Because the number of successful readings by each antenna varies over time for each tag, the recorded time series had to be regularized to a constant sampling rate. The sampling rate was determined based on the minimum achieved reading rate of the tags. For our study, we used 1 second as the sample time because the number of readings per second for tags in the trauma room were greater than one if the tags were not occluded by people or other objects. The output of regularization is an $I \times J$ matrix (we call it an "antenna-object frame"), where I rows represent I tagged object types and J columns represent J antennas installed in the room. The element (i, j) is the averaged RSS collected during one second for object type i by antenna j. We had $I = 12$ types of objects and J = 8 antennas. The regularization process for every second generated a 12×8 matrix. We put a zero if no data was received by an antenna for a given point. Note that the RSS value has a physical meaning, where "0" means the received signal strength is 1 mW. Because in our implementation the distance between tags and antennas is at least 2m, the actual received signal strength is much lower than 1 mW, so it is safe to use "0" to indicate that no data were received.

3 Stacking antenna-object frames: The final step is to stack the antenna-object frames over time (Fig. 4.6). The pre-processed RFID data forms a 3D matrix with T layers of antenna-object frames, where T is the total time (in seconds) of recorded RFID data from all executions of the process. In our case, $T = 50,000$

sec, collected during 16 resuscitations.

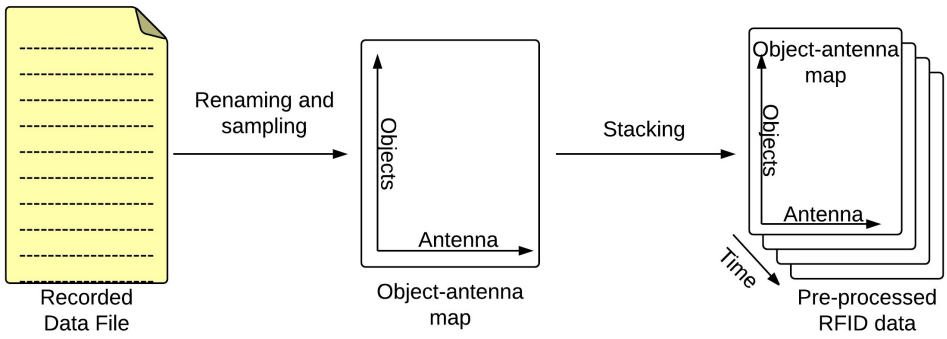


Figure 4.6: Our preprocessing procedure for RFID data.

### 4.3.2 Deep Learning Model

#### Neural Network Structure

Several types of deep learning models have been proposed for different application and sensor types. Examples include the Convolutional Neural Network (CNN), widely used for image classification [114] and recently for speech recognition [2]], the Deep Neural Network (DNN), used for speech recognition and audio sensing [42], and a multimodal structure used for audio-visual speech recognition [70]. Our choice of network structure was driven by the nature of our RFID data. The RFID signal received from a single tag by one reader antenna is a one-dimensional series, similar to a speech signal, for which both the DNN and CNN have been used. Our RFID data were collected by several antennas from multiple tags on same or different objects, which resulted in two additional dimensions: the receiving antenna and object/tag ID. We chose CNN over DNN because we wanted to process data from all tags together to capture potential concurrent object uses. CNN better handles high-dimensional input by representing it as a high-dimensional matrix. In addition, DNN is ineffective at learning features and requires as input extracted features rather than raw data. Given that it is hard to optimize manually selected features, a poor selection will lead to poor performance. CNN can generate useful features via its learnable filters, so it can directly accept RSS data as input. We implemented the CNN with three convolutional layers, followed by

three fully-connected layers and a softmax layer for output (Fig. 4.7) . We used the CNN with rectified linear units (ReLUs) because such units train several times faster than traditional tanh units [40, 42]. We recently implemented a modified DNN structure using a similar dataset, but directly using high-dimensional RSS in the input layer remains a challenge [54]. Unlike CNN structures for other applications, we designed the input and convolutional layers to reflect the structure of RFID data collected with multiple antennas. We next describe the building blocks of our CNN (Fig. 4.7).

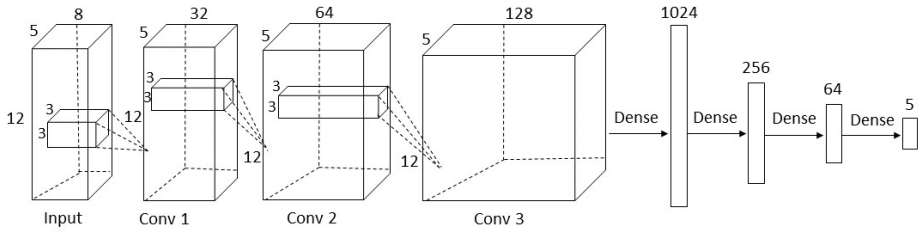


Figure 4.7: The convolutional neural network structure with 3 convolutional layers and 3 fully (dense) connected layers.

### Input Layer

The input layer prepares the input data for the convolutional network. The input data needs to be represented differently for different applications. For image classification, the input layer is often a single gray-scale image or three gray-scale images (for red, green and blue channels of color images). For speech recognition, the input layer is often constructed as a time-frequency feature map. In general, RFID data in the input layer has three dimensions that represent the objects, the antennas, and the observation time window. Unlike speech from a single microphone, RFID data are recorded by multiple antennas and have an extra dimension of space. Unlike stationary images,

RFID data are related in three dimensions: spatially across antennas, temporally over time, and semantically over tagged objects that are manipulated concurrently in one or several parallel activities. Some similarity exists with video data processing where image frames are temporally related and pixels in each frame are spatially related [95]. Our input layer is formed in two steps: first stacking t antenna-object frames collected over t seconds, and second rotating the 3-D matrix to make the object and time as the first two dimensions (Fig. 4.7), and antenna as the third dimension. The time window t is determined by the duration of the shortest activity. A window shorter than the shortest activity minimizes the chance that multiple activities will be represented in it. In our problem domain, some activities take a short time, such as evaluating the ears, which on average lasts 10 seconds and has the lowest average duration. Based on Nyquist theorem, we chose t=5 to ensure that the time window is just 50% shorter than the shortest activities. The convolution operation sums the contributions from different planes in the input layer and uses ReLU as:

$$h_j = max(0, \sum_{k=1}^{K} h_k * w_k j) \tag{4.4}$$

where $h_j$ is the jth plane of output data from each convolutional layer, hk is the kth plane of input data which has K planes in total and $w_k j$ is the kth plane of kernel j. We used the number of object types and the time value as the first two dimensions of input layer, and the number of antennas as the third dimension. The first two dimensions represent the RSS from each object over a time window, which for stationary objects should appear flat when visualized as a gray-scale image. If an object is manipulated, the RSS of its tag should be very different from stationary state in the visualization. This arrangement also ensures that each convolutional operation is performed on the data collected by all antennas, which makes our network structure applicable to scenarios with different number of antennas and antenna arrangements.

**Convolutional Layers**

The convolutional layers with sets of learnable filters are the core building blocks of convolutional neural networks, and the pooling layers implement the input data down-sampling. Several parameters need to be determined for constructing the convolutional layers [40, 49]. For each convolutional layer, the size of the convolutional kernel decides the shape and number of feature maps used in convolution operation. No analytical procedure is available to determine the optimal number of convolutional layers for a given application. The most suitable network structure is usually determined empirically. We chose to have 3 convolutional layers in our network, with odd-number kernel sizes: 3×3×32, 3×3×64 and 3×3×128, and with stride 1, which have been shown as efficient in the VGG net [96]. Because the input data had a small dimension (12 objects × 5 one-second frames) in each antenna plane, zero padding was added to perform a wide-type convolution in order to maintain the size of each output plane the same as the input plane. The number of feature maps in each kernel and the number of convolutional layers was determined empirically. We used 5,000 seconds of data from 50,000 seconds of total available RFID data as training data for classifying the five resuscitation phases. The number of feature maps in each convolutional layer was determined by a script looping through the powers of 2 from 16 to 256 and choosing the combination of kernel sizes for convolutional layers that performed best on detecting the five resuscitation phases. We reasoned that 3 convolutional layers will also provide the best tradeoff for activity recognition, because phase detection and activity recognition use the same RFID data as input data. More convolutional layers generally yield better performance, but the performance gain diminishes. We only tested the CNN with 1 to 4 convolutional layers, because the network with 5 layers has over 30M weights which was not feasible for our hardware. The results show only a small gain in precision, recall and F-Score when using four convolutional layers but a large difference in memory cost (around 2 times). We concluded that using 3 convolutional layers has the best tradeoff between the computational resources and performance gain. We did not use pooling layers, which in other applications have been used to extract low-level, shift-invariant

features, and to reduce the data dimensionality for computational efficiency. Unlike images, which normally contain redundant pixels unimportant for the classification, the raw RFID data matrix has very little redundancy. Pooling with a minimum window (2×2) would only leave one fourth of the pooled data which would distort the spatial and temporal relationships of the RFID data. A new pooling strategy with learnable weights was recently proposed [2], which will be tested in our future work.

**Fully Connected Layers**

No more than two fully-connected layers have commonly been used to avoid overfitting [40, 2]. Our experiments showed that in our domain 3 fully connected layers work better than 2 layers. This finding is due to the orders-of-magnitude dimensionality reduction between the neurons in the last convolutional layer (7680) and the output layer (5 for process phases and 10 for activities).

**Model Training**

We trained two CNNs to detect 5 process phases and 10 resuscitation activities, respectively, using preprocessed RFID data (Fig. 4.6) from 16 trauma resuscitations. The label (one of 5 process phases or 10 activities) for each second of data was manually generated by medical experts from video review of the corresponding trauma resuscitations. The 16 resuscitations provided a total of 50,000 seconds of data. Due to the great variability of the resuscitation process, the duration of each activity is unpredictable, and some of the 10 activities were not well represented, unlike the 5 resuscitation phases which were all well represented. Given the unbalanced dataset, randomly selecting the number of samples would not guarantee sufficient data for all activity classes during training and testing. As suggested [52], we selected a percentage of data from each class for training and used the remainder for testing. Over-fitting was a concern because process-phase detection is a relatively small multi-class classification problem with only 5 classes, compared with other CNN applications, such as image classification with thousands of classes [40]. We took two steps to avoid model overfitting. First, we applied the "dropout" in fully-connected layers, which is widely

used in CNNs to avoid overfitting during model training [97]. Second, we implemented the cross-validation and set the system to stop training when the cross-validation error starts to increase. We initialized the learning rate at 0.01 and adjusted it based on the ADAM optimization (Adam Optimizer in TensorFlow) [39].

We implemented our CNN using Microsoft Azure cloud service and locally with Google TensorFlow [1]. Both frameworks achieved similar performance and allow users to manually define the CNN with sharp Net or Python. The advantage of Azure is that it allows the user simultaneously run several CNN training processes with different data or network parameters and easily compare their performance. The training process is faster in Azure compared to training with computers using a Core i5 CPU. On the other hand, in Azure the trained weights are not accessible to the user. The TensorFlow runs locally, though the training speed depends on hardware and it is impractical simultaneously to train several models on a single computer. All the trained weights, however, are accessible, which makes TensorFlow suitable for model analysis. Because of these features, we used Azure for CNN model design and TensorFlow for experimental evaluation.

### 4.3.3 Experimental Results

Compared with detecting phases of a process, medical activity recognition is considered more important, because of its fundamental role in building decision-support systems or other artificial intelligence systems that help improve patient care and outcomes. Unlike systems designed for recognizing simple physical activities of individuals, such as sitting, standing, or sleeping [4, 12, 44], recognizing complex teamwork activities is significantly more challenging. We trained our convolutional neural network (Fig. 4.7) with preprocessed RFID data for 11 medical activities (10 shown in (Table 4.2) and "other" as a catch-all activity). We could not split the training and testing data as we did for process-phase detection, where we used 5000 RSS samples from each class for training data and the rest for testing, because we had very limited data for brief activities, such as evaluation of patient's ear. Using the same number of instances for training each activity could cause bias. We randomly selected 40% of data in each

class for training and the remaining 60% data for testing. Our system achieved average accuracy of 80.40% for recognition of 11 activities (Fig. 4.8).

| | OT | BA | BC | BP | CA | EA | EAR | EC | M | N | PU |
|---|---|---|---|---|---|---|---|---|---|---|---|
| OT | 72.76% | 0.66% | 4.32% | 2.56% | 4.87% | 5.64% | 1.44% | 7.06% | 0.03% | 0.05% | 0.60% |
| BA | **13.22%** | 85.90% | 0.44% | 0.00% | 0.00% | 0.00% | 0.00% | 0.44% | 0.00% | 0.00% | 0.00% |
| BC | 7.60% | 0.79% | 54.08% | 4.62% | 2.20% | 5.38% | 6.30% | **17.80%** | 0.42% | 0.08% | 0.73% |
| BP | **20.33%** | 2.40% | 2.59% | 64.14% | 9.06% | 0.00% | 0.00% | 0.18% | 0.00% | 0.00% | 1.29% |
| CA | 7.06% | 0.00% | 0.00% | 0.00% | 92.94% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% |
| EA | **12.89%** | 0.00% | 2.22% | 0.00% | 0.00% | 80.67% | 3.33% | 0.89% | 0.00% | 0.00% | 0.00% |
| EAR | 2.43% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 97.57% | 0.00% | 0.00% | 0.00% | 0.00% |
| EC | 9.95% | 0.90% | **21.24%** | 2.16% | 1.16% | 3.66% | 3.90% | 56.82% | 0.02% | 0.02% | 0.18% |
| M | 6.15% | 3.08% | **13.85%** | 3.08% | 0.00% | 7.69% | 0.00% | 3.08% | 63.08% | 0.00% | 0.00% |
| N | 3.92% | 1.96% | 0.00% | 0.00% | 5.88% | 5.88% | 1.96% | 0.00% | 0.00% | 76.47% | 3.92% |
| PU | 8.59% | 0.00% | **10.10%** | 4.04% | 3.03% | 5.56% | 3.54% | 4.55% | 0.51% | 0.51% | 59.60% |

Figure 4.8: Confusion matrix for recognition of 11 resuscitation activities. "OT" for activity other than selected 10 activities.

Unlike some other activity recognition systems that trained independent binary classifiers for different activities [52, 110], our system treats activity recognition as a multi-class classification problem and can scale up if additional activities need to be recognized. To avoid the evaluation bias caused by different training and testing sets, we fed the same training and testing sets into traditional classifiers. We compared the performance of our convolutional neural network to traditional classifiers using evaluation metrics introduced above. Our network still performed best compared with all other classifiers (Fig. 4.9). It achieved about 10% higher F-score compared with random forest, which was the second best classifier, and around 30% higher F score compared with all other classifiers. The same performance gain held for other evaluation metrics.

To demonstrate the advantage of our deep learning system in medical applications, we first compared this system with our previous system for resuscitation activity recognition from passive RFID that uses a cascade model with manufactured features such as visible antenna combination, the Spearman rank correlation coefficient, and other features [52]. The RFID data used for evaluating our deep learning and that we previously used [52] were collected in same environment with real-patients, using different RFID readers (Impinj vs. Alien). Our deep learning achieved 30% higher F-score, and MCC and double informedness scores compared with the system in [52] using sensor

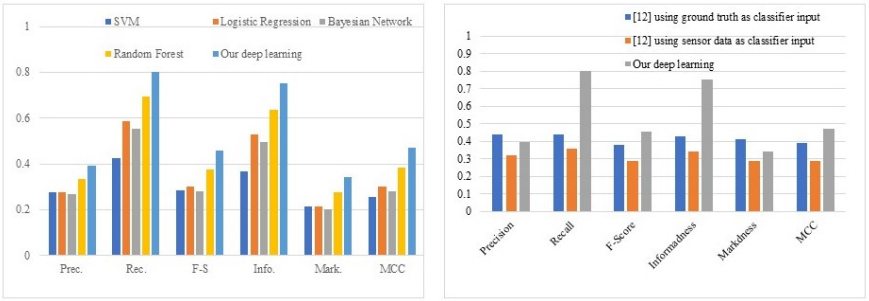


Figure 4.9: Left: Comparison of results using different classifiers for resuscitation phase prediction. Right: Comparison of results in [52] with our deep learning system in the same application environment.

data as input. Even when our previous system used ground truth of object-use as input to the classifier (instead of object-use detected from sensor data), our deep learning still achieved better performance in F-score, informedness and MCC (Fig. 4.9) . This comparison shows the power of deep learning to process the noisy RFID data and the potential of deep learning model applied to RFID-based applications. We also compared the performance of our deep-learning system using RFID data for medical activity recognition with several state-of-the-art recognition systems for real-world application scenarios [110, 17, 16] (Table 4.3). Although these systems were implemented for different environments, a comparison shows that our deep learning was able to achieve performance similar to vision-based systems for activity recognition. As was the case with other applications [2,4], our deep learning-based activity recognition system also achieved better performance compared to systems that used traditional classifiers.

### 4.3.4 Possible Extension

Unlike image and audio, mobile sensors such as RFIDs have had relatively few deep learning implementations. Previous research preprocessed the data into a 3D antenna-object-time matrix [3]. Although feasible, this data format faces two problems: (a) there is no redundancy for ConvNet pooling operations, and (b) the spatial relationships between reader antennas and tags present in the received signal strength (RSS) data are not well represented. We introduce RSS maps, a new RFID data representation suitable

Table 4.3: The Comparison of Process Phase Prediction by Domain Experts and Our System.

| ***Activity Recognition System*** | ***Input Source*** | ***Method*** | ***Accuracy of Activity Recognition*** |
|---|---|---|---|
| A Scalable Approach to Activity Recognition based on Object Use [110] | Video data and RFID | Dynamic Bayesian Network | 60.84% average accuracy for 16 single-person daily-life activities, without using domain knowledge |
| Structure Inference Machines: Recurrent Neural Networks for Analyzing Relations in Group Activity Recognition [16] | Images | Recurrent Neural Networks | 81.2% average accuracy for 5 real-life group activities |
| Deep Structured Models for Group Activity Recognition [17] | Images | ConvNet | 80.6% average accuracy for 5 real-life group activities |
| Our deep learning network | RFID | ConvNet | 80.2% average accuracy for 10 group activities during actual resuscitations |

for spatial feature extraction in ConvNets. An RSS map projects the RSSs received from each tag onto the effective field of coverage for each antenna (Fig. 5 right). In our experiments, 7 of the 8 reader antennas were hung on the ceiling facing down (black boxes in Fig. 5 left). We approximated antenna's coverage area by a circle on the room floorplan (scale: 1px $\leftrightarrow$ 1dm2). The circle radii were manually measured by moving a tag horizontally away from the antenna in several directions, always starting below the antenna moving away until the tag could not be detected by the antenna. The coverage radius was determined as the average tag-visibility-loss distance from these experiments. In our case, most tags were 0.6 meters above the ground (approximately at the height of person's hands, assuming that tagged objects are used for work), and were visible laterally up to about 1.2 meters away from an antenna. For the tilted antenna mounted on the wall facing the area of interest (triangle in Fig. 5 left), we used an ellipse approximation. The room's operational area was roughly 3.6×4.8m, so each object's RSS map was 36×48px.

Generating the combined RSS map for all (25 types of) tagged objects required two steps. First, we created maps for each object type (Fig. 5 right). Coverage areas of the

8 antennas were filled-in with that object's RSS (with zero values outside the coverage). The 8 resulting maps were then averaged, generating one RSS map per object type. Second, we created the combined RSS map by stacking the 25 2D maps for 25 object types into a 3D matrix.

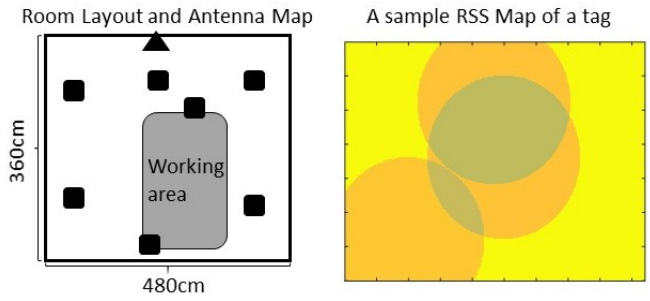


Figure 4.10: Left: our RFID reader antenna configuration. Right: example RSS map visualization for a tagged object.

## 4.4 Summary

A key limitation of our current system is that it relies only on relies on RFID sensing to capture activity information and making predictions. Some activities, such as palpation of the patient's body, do not involve the use of physical objects that can be tagged. In addition, RFID technology does not work very well with metal objects or liquid containers, and objects in sterile packages can be tracked only until the packaging is discarded. Our continuing research involves the use of multimodal sensing for activity recognition. In particular, we are using the Kinect sensor with microphone array to capture depth images and ambient sound for more reliable and complete activity recognition. Generalization is an important aspect of a classifier, and our system generalizes well for different attributes of trauma resuscitations. The cases we used for training and testing were performed by different trauma teams with patients having different injuries and health conditions. Unlike a model trained for image classification, a CNN model trained for RFID data cannot be directly used in a different environment with different antenna configuration or tagging strategy. For image analysis, a target appears similar regardless of the changing background or camera. On the other hand,

the same activity captured in RFID data by different antenna configurations and tagging strategies may be rather different because the radio signal may experience very different conditions. As a result, the model has to be retained for different antenna configurations or tagging strategies. Input data that is less influenced by the hardware configuration, such as using the standard deviation of RSS instead of RSS values would partially solve this problem because standard deviation is lacking other information. RSS values depend on the distance between tag and reader antennas and the status of the tag (covered or exposed) while the standard deviation does not contain such information. Further investigation is needed to find the sensory input both robust to hardware configuration and representative enough to support activity recognition and better model generalization, which will be our future work.

# Chapter 5

# Attention And Multi-sensory Based Activity Recognition

## 5.1 Chapter Introduction

We introduce an activity recognition system that recognizes activities in two steps. First, it localizes the activity by generating an *activity mask* outlining the location where the activity is expected to occur. For mask generation, we used a conditional generative adversarial network (cGAN) [66]. Given that activities are continuous and usually represented as video clips, we introduced a Conv-LSTM-Deconv structure as a continuous mask generator with cGAN for training. The cGAN-based image generator has shown better performance than a Conv-Deconv-based generator [34]. The ConvNet-LSTM structure has been used to model spatio-temporal associations [35, 80]. In addition to the adversarial loss, we introduced a spatio-temporal loss and implemented perceptual loss [36]. These losses penalize the neural network for pixel-wise errors in the generated mask and discontinuities between masks from consecutive frames. Second, the generated mask is appended to each color channel of its input video frame to delimit the activity region for the activity recognizer. Because the ConvNet-LSTM structure has been used successfully for modeling spatio-temporal activity associations [56, 103], we adopted a VGG-LSTM network for activity recognition.

To train and test our system, we manually created activity masks for two datasets. We selected six activities from a well-known Olympic sports dataset (15 videos per activity) and for 10 frames of each video manually created binary masks outlining the activity performer's location. We also manually created multilevel masks highlighting trauma team members and their roles for depth videos of trauma resuscitations.

Our system achieved state-of-the-art performance compared with previous activity recognition approaches [43, 72, 21, 35]. In addition, it generated activity location masks.

Our system works online, as opposed to offline prediction based on features extracted from the entire video [107, 58]. Unlike the approaches that rely on crafted features or purely rely on learned features, our system automatically generates activity masks that promote learning the most representative features. This property makes our system generalizable and scalable. Our visualization of the learned feature maps demonstrates that the generated masks enable the system to learn the features that are associated with activities, unlike a VGG net trained using only video frames.

## 5.2 Region-based Activity Recognition

### 5.2.1 Methodology

Our system has two network structures (Fig. 5.1), one for activity-mask generation that includes a generator and discriminator and one for activity recognition. For selected video frames in our datasets, we manually created a binary activity mask outlining the activity performer, or a multilevel mask for team members with different roles. We used these masks as the ground truth for system training.

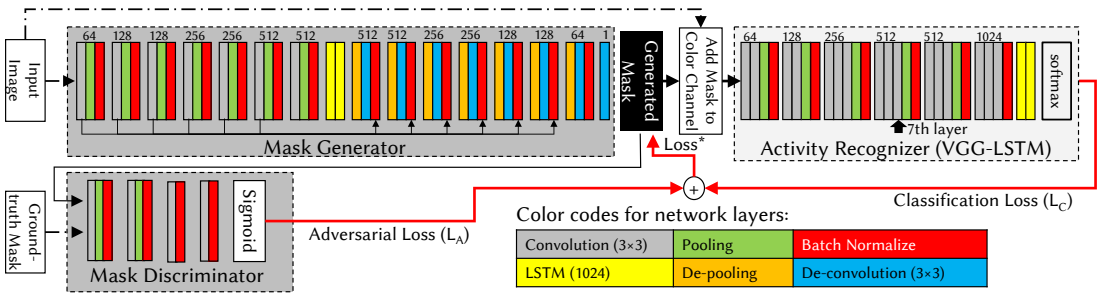


Figure 5.1: Our Conv-LSTM-Deconv cGAN structure with U-Net connection (dark shaded boxes on the left [88]). The numbers above convolution layers indicate the number of filters in each convolution kernel. (See digital version for color codes.)

#### Model Structure

We used a cGAN to generate the activity masks and a VGG-LSTM structure for activity recognition (Fig. 5.1). Generating the mask is comparable to other space-transformation or mapping tasks, such as text-to-image mapping [84], image-to-image mapping [34], or image super resolution (low to high-resolution mapping) [50]. The

current state-of-the-art approaches to these problems are GAN-based, We developed our model based on cGAN to generate the continuous activity masks.

Unlike single image segmentation which relies only on the spatial associations between inputs and outputs, generating activity masks for continuous video frames relies on both spatial and temporal associations. The generated mask should spatially match the ground truth mask. The generated mask should temporally be continuous over time (i.e., smoothly transitioning). We introduced a Conv-LSTM-Deconv structure as the generator of the conditional GAN model (Fig. 5.1, generator). The Conv-Deconv network with fully convolutional structure was used [62] to avoid the use of fully-connected layers, making the model more memory-efficient and better at preserving spatial features. The LSTM [22] has been widely used to model temporal associations and recently for acoustic modeling [92]. We inserted the LSTM layers between the convolutional and deconvolutional networks to model the temporal associations of spatial features learned by the ConvNet. This approach increased the average precision of activity recognition by 6% compared with the Conv-Deconv generator alone (Fig. 5.1).

The adversarial loss from the discriminator ($L_A$ in Fig. 5.1) can only tune the weights in the mask generator based on the feature-wise difference between the generated and ground-truth masks. To better tune the generator, we built a cascade of the mask generator followed by activity recognizer (Fig. 5.1) which produces activity recognition error to penalize the generator ($L_C$ in Fig. 5.1). This structure takes the generated mask and original image, appends the mask after each color channel, and uses it for activity recognition. Because recognizing some activities requires background features, we decided to *append* the generated mask as an additional "color channels" of the input image after each regular RGB color channel, instead of directly applying the mask on the input image. This approach allows the system to learn some features not directly associated with the activity performer. Our activity recognition module favors the area covered by the activity mask but also allows inclusion of the whole background. The activity recognition loss is backpropagated for tuning the generator ($L_C$ in Fig. 5.1).

As previously shown [30], deep Conv-Deconv structures are hard to converge. To

improve the convergence, we adopted the "U-Net" [88] to add extra connections between the associated convolutional and de-convolutional layers.

**Loss Functions**

The generative adversarial network (GAN) [24] learns the mapping of input data $z$ to output mask $y$, denoted as $G : z \rightarrow y$ [24]. The regular GAN model can be extended to a conditional model that uses additional information $x$. The conditional GAN [66] establishes the mapping function based on both the input data $z$ and conditional information $x$, denoted as $G : \{x, z\} \rightarrow y$. The additional information $x$ can be any auxiliary information. We used the manually generated masks which contained ones or categorical labels at pixels related to the activity, and zeros for the unrelated pixels as the condition $x$.

Our system was designed for activity recognition, so that loss could be measured spatially and temporally. In addition to the regular adversarial loss $L_A$, we used three other losses:

$$L^* = L_A(G, D) + \alpha L_{st}(G) + L_p(G) + \beta L_C(G, C) \tag{5.1}$$

where $\alpha,\ \beta$ are the manually defined weights. We empirically determined that $\alpha = 0.8$ and $\beta = 0.2$ showed the best performance.

**The first term** $L_A(G, D)$ is the regular adversarial loss from the mask discriminator (Fig. 5.1). The discriminator uses the ConvNet structure to measure the feature-wise similarity between the generated and activity masks, and normalizes the similarity to the range of zero to one. The adversarial loss of cGAN model is:

$$\begin{aligned} L_A(G, D) = {} & E_{x,y\sim p_{data}(x,y)}[\log D(x, y)] \\ & + E_{x\sim p_{data}(x), z\sim p_z(z)}(1 - \log(1 - D(x, G(x, z)))) \end{aligned} \tag{5.2}$$

where $G$ denotes the mask generator and $D$ denotes the discriminator. The training adjusts the parameters of $G$ to minimize the objective while the discriminator $D$ maximizes the objective.

**The second term** $L_{st} = L_{\ell 1} + L_t$ is the proposed spatio-temporal loss calculated by the generator itself. $L_{\ell 1}$ denotes the pixel-wise loss between the generated activity

mask and the ground truth, which is commonly used in GAN image generation tasks as:

$$L_{\ell 1} = \frac{1}{CWH} \sum_{c=1}^{C} \sum_{w=1}^{W} \sum_{h=1}^{H} ||[\phi_G(x)]_{c,w,h} - y_{c,w,h}||_1 \tag{5.3}$$

where image dimensions are the number of color channels $C$, width $W$ and height $H$. In our case of activity masks, $C = 1$ because of single mask per frame, whether the mask is binary or multilevel. The $\phi_G(\cdot)$ denotes the generated mask, and y denotes the ground-truth mask. $||\cdot||_1$ denotes the pixel-wise L1 norm which favors similarity between the generated activity mask and ground truth for every frame. The $L_t$ is the temporal loss. To make the system generate masks for continuous activities, the L1 distance of the consecutive masks should have minimal variation. We thus introduced $L_t$ to model the temporal loss between frames as:

$$L_t = std([L_{\ell 1}]) \tag{5.4}$$

where the $std(\cdot)$ denotes the standard deviation of the list $[L_{\ell 1}]$ of the pixel-wise L1 distances across frames.

**The third term**, $L_p(G)$, denotes the perceptual loss for the generator [36]. We define $\hat{y} = \phi_G(x)$ as the output of the generator. We followed the definition:

$$L^i_{feat}(\hat{y}, y) = \frac{1}{C_i W_i H_i} ||\phi^i_{vgg}(\hat{y}) - \phi^i_{vgg}(y)||_2^2 \tag{5.5}$$

$$L^i_{style}(\hat{y}, y) = ||G^i_{vgg}(\hat{y}) - G^i_{vgg}(y)||_F^2 \tag{5.6}$$

where $C_i \times W_i \times H_i$ defines the shape of the feature map, $L^i_{feat}$ denotes the feature loss and $L^i_{style}$ denotes the style loss at corresponding $i^{th}$ convolution layer of the VGG net [96]. The $||\cdot||_2^2$ denotes the squared L2 norm, and $||\cdot||_F^2$ denotes the squared Frobenius norm. The $\phi^i_{vgg}(\cdot)$ term denotes the channel-wise Gram matrix at $i^{th}$ convolution layer of VGG net. The element in position $1 \leq m, n \leq C_i$ of Gram matrix for any feature map $f$ is defined as:

$$[G^i_{vgg}(f)]_{m,n} = \frac{1}{C_i W_i H_i} \sum_{w=1}^{W_i} \sum_{h=1}^{H_i} (f_{m,w,h} \times f_{n,w,h}) \tag{5.7}$$

In addition, we applied a total variant regularization $L_{TL}(\hat{y})$ to generator output $\hat{y}$ for smoothing [36]. The perceptual loss consists of feature-reconstruction loss, style loss

and total variation regularization, which have been successfully used for GAN-based image generation [36, 50]. We used this loss for generator tuning.

**The last term** $L_C(G, C)$ is our proposed classification loss. We designed the generator to take the additional loss from the error of the activity recognition network $C$. The activity recognition loss is defined as the categorical cross-entropy from the activity recognizer. The cross-entropy is calculated using the stacked generated masks and input images as input and activity ground truth as output. This classification loss can be expressed as:

$$L_C = -\sum_N y'_n \log y_n \tag{5.8}$$

where $N$ is the total number of classes, $y'_n$ is the predicted probability distribution of class $n$, and $y_n$ is the ground truth for class $n$. For each iteration, we first trained the activity recognizer using the stacked ground truth mask and the input image as input, and used the activity label to tune the weights of the recognizer. To generate the activity recognition loss, we used the generated mask instead of the ground truth for activity recognition and calculated the categorical cross-entropy of activities.

To study the impact of different losses, we ran the comparison experiments with versions of our system trained on different losses. The results show that the proposed spatio-temporal loss has a positive contribution to activity recognition and segmentation (Table 5.1). We used the per-pixel accuracy [14], as well as per-frame activity recognition accuracy for this comparison.

Table 5.1: Activity recognition accuracy and generated mask per-pixel accuracy for different loss combinations.

| **Loss functions** | **Activity rec acc'y** | **Per-pixel acc'y** |
|---|---|---|
| $L_A$ | 62.21% | 69.29% |
| $L_A + L_C$ | 64.68% | 71.03% |
| $L_A + L_{st} + L_C$ | 78.16% | 81.37% |
| $L_A + L_p + L_C$ | 70.92% | 83.89% |
| $L_A + L_{st} + L_p + L_C$ | **81.75%** | **86.71%** |

### Activity Recognition and Localization

Once the system is trained, we append the activity recognizer after the generator and run feed-forward for activity recognition. It is possible that a dataset contains a large number of video clips, but only a few of them have manually-generated masks for training. A system trained with a limited amount of generator-training data is more likely to make mistakes when generating masks. To avoid the mask generation errors propagating to the activity recognition results and help the system tolerate these errors, we chose to append the generated mask to the original image after each channel instead of directly applying the mask to the original image by performing their multiplication. In this way, information from the original input image is preserved even when the generated mask contains errors.

The activity recognition network was not well trained during mask generator training due to the limited number of frames with ground truth masks. We then used the data with only activity class labels and no activity masks to further fine-tune the activity recognition network (VGG). The system first generated the mask for each input frame and then appends the generated mask to this input frame. This stacked image is then fed into the activity recognizer, which is trained using ground truth activity labels. We set the weights of the mask generator as untrainable during this fine tuning so that we can only tune the weights of the activity recognizer.

Activity localization is an end product of our method that can be simply accomplished by calculating the bounding box of the fully-connected regions in the generated masks and applying the bounding box to the input image. During activity prediction, we performed an image morphology close operation to the generated masks to eliminate the noise before calculating the bounding box.

### Model Implementation

We implemented our system with Keras using the Theano backend. We used the fully convolutional structure [62] with He initialization [29]. *Leaky ReLU* and *tanh* were used as activation functions. The batch normalization and dropout strategy [97] was used

to avoid overfitting. The generator was tuned three times in each training iteration: first using the adversarial loss from the discriminator, then using the spatio-temporal and perceptual losses [36], and finally using the classification loss from the activity recognizer.

We trained the network on the single GTX1080 GPU with 8GB memory. The loss functions were manually defined using Keras backend programming. Training with $128 \times 128$ frames took two days to converge. Because we used *Leaky ReLU* and *tanh* as activation functions, we first normalized the input data in $[-1, 1]$.

### 5.2.2 Experimental Results

#### Dataset

**Olympic Sports Dataset with Masks:** We selected six activities from a Olympic sports dataset for labeling (Table 5.2). The dataset contains videos recorded with moving backgrounds and moving athletes. Because the original videos had different frame rates, and most of the frames within each second were visually similar before the activity started, we down sampled the original color video to three frames per second and selected only the most representative frames (covering different stages of activity performance) for manual labeling. The video frame selection can be automatically accomplished based on the pixel difference between adjacent frames by selecting the frames most different than previous frames. The frame difference can be estimated by calculating the optical flow. We manually selected masks for 15 videos for each of the six activities, resulting in a total of 900 frames. We kept generating more masks for all 16 activities in the Olympic sports dataset.

#### Activity Recognition in Sports Videos

We evaluated three aspects: the overall performance, the performance breakdown for each activity and the system generalizability.

**We first evaluated the overall activity recognition performance** using accuracy and the commonly used F-measure with 30% videos of each activity (Table

Table 5.2: Activities and background labels in our datasets.

| Olympic Sports | Label # | Trauma team roles | Label # |
|---|---|---|---|
| Background | 1 | Background | 1 |
| Lay Up | 2 | Patient Bed | 2 |
| Bowling | 3 | Nurse Left | 3 |
| Weight Lifting | 4 | Respiratory Therapist | 4 |
| High Jump | 5 | Examine Provider | 5 |
| Long Jump | 6 | Team Leader | 6 |
| Lay Up | 7 | | |

5.3). We compared our system with baseline activity recognition models (ConvNet, CNN-LSTM) using the same dataset (Table 5.3). We also compared the system performance with a different configuration for activity recognition (Table 5.3) to analyze the contribution of different losses to our proposed system. The results shown that: **1** Our baseline VGG net and VGG-LSTM structure achieved performance similar to representation-based approaches [43, 72] (Table 5.3, left side). While the regular deep-learning models have the ability to learn representative features, this finding showed that, they are also limited by their simple structure when generalizing to complex tasks such as recognizing activities where unrelated people are present. The VGG-LSTM achieved an increase in accuracy increase of about 5% compared with ConvNet only, due to its ability to model the temporal associations of extracted features. **2** We also observed that different network models generated masks of different quality (Table 5.1). The cGAN with our proposed losses generated the most accurate masks for most video frames (Fig. 5.2). We also quantitatively evaluated the generated masks. Because we do not have the ground truth masks for all videos, we adopted the 80% - 20% ratio to split the training and testing sets for the frames that had ground truth masks (Table 5.1). **3** The quality of generated masks affected the activity recognition results (Table 5.1, 5.3). Our proposed spatio-temporal loss increased the activity recognition accuracy by ≈3% compared to a regular cGAN trained only using the adversarial loss (Table 5.3). Including the classification loss into our structure increased the activity recognition accuracy by ≈2% (Table 5.3). **4** The state-of-the-art activity recognition strategies [107, 58] ] used representation based on multiple features and moving trajectories to describe the entire video. These methods require computationally expensive

calculations for feature representation from the entire video. Our proposed approach can make per-frame activity predictions at 20fps by running feedforward networks, and simultaneously provide the activity location information. We believe runtime operation offers significant advantage though with slightly lower performance.

Table 5.3: Averaged accuracy and F-scores of different activity recognition methods for 6 activities in Olympic sports dataset.

| System | Acc. | Prec. | Rec. | F1 |
|---|---|---|---|---|
| Ref. [43] | na | 65.35 | na | na |
| Ref. [72] | na | 74.93 | na | na |
| Ref. [21],[35] | na | 74.5/80.0 | na | na |
| Ref. [58],[107] | na | 91.2/91.4 | na | na |
| ConvNet | 57.38 | 69.34 | 57.38 | 62.80 |
| Conv-LSTM | 59.15 | 75.62 | 59.15 | 66.38 |
| Conv-Deconv | 40.10 | 64.01 | 40.10 | 38.77 |
| Conv-LSTM-Deconv | 61.92 | 72.89 | 61.92 | 61.45 |
| Our cGAN ($L_A, L_{st}\&L_C$) | 78.16 | 84.88 | 78.16 | 79.52 |
| Our cGAN ($L_A, L_p\&L_C$) | 70.92 | 83.63 | 70.92 | 73.78 |
| Our cGAN ($L_A, L_{st}, L_p\&L_C$,6 activities) | **81.75** | **87.79** | **81.75** | **84.66** |
| Our cGAN ($L_A, L_{st}, L_p\&L_C$,16 activities) | **73.62** | **80.01** | **73.62** | **76.68** |

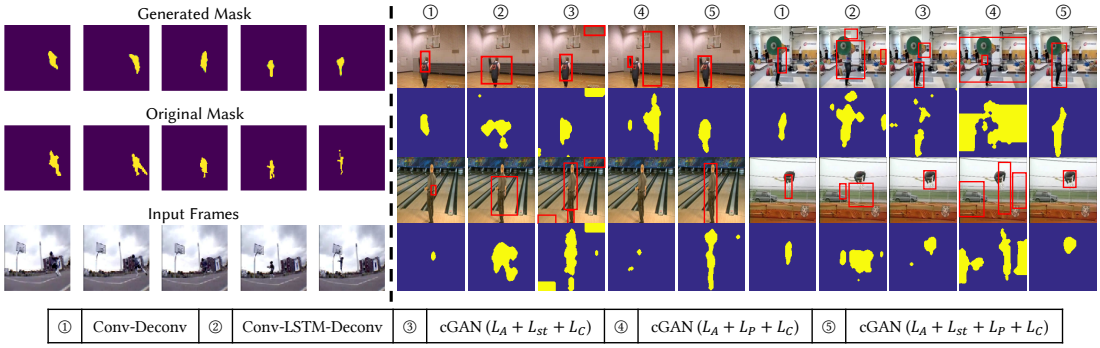


Figure 5.2: Left: The generated masks and ground truth for a sequence of input frames. Right: The input video frames with activity bounding box generated using different generator structures. (See digital version for colors.)

**Second, we evaluated the recognition performance for each of six sports activities** and compared that with previous research on the same dataset (Table 5.4). Our structure outperformed existing methods [43, 72, 38, 62, 92], particularly for the activities during which unrelated people were present (shaded in Table 5.4), such as judges and observers. For these activities, activity location was critical for recognition. The gym vault shares a similar starting posture (running) with other activates, and

the running step lasts longer than other steps. This fact makes the gym vault appear similar to some other activities, causing low predictive performance.

**We finally tested generalization in our system.** We used the generator trained for the six sports activities to generate masks for 16 activities that included the original six. We then tuned the activity recognizer to recognize all 16 activities. For all 16 activities, our system achieved 7% lower performance than for the six it trained on (Table 5.3), which was caused by errors in the generated masks. Our system was still able to achieve performance comparable to existing systems [43, 72, 35, 21], showing that it generalizes and to some extent tolerates errors in the generated masks.

Table 5.4: Activities and background labels in our datasets.

| *Activity* | *Ref. [43]* | *Ref. [72]* | *Ref. [38]* | *Ref. [92]* | *Ref. [107]* | *Ref. [62]* | *Our cGAN* $(L_A, L_{st}, L_C)$ | *Our cGAN* $L_A, L_p, L_C)$ | *Our cGAN* $(L_A, L_{st}, L_p, L_C)$ |
|---|---|---|---|---|---|---|---|---|---|
| Lay Up | 75.8 | 77.9 | 85.7 | 35.5 | 67.2 | 64.0 | 89.9 | 87.8 | 96.7 |
| Bowling | 66.7 | 72.7 | 87.6 | 70.5 | 68.6 | 66.6 | 92.2 | 91.4 | 93.6 |
| Snatch | 41.8 | 69.2 | 81.3 | 65.6 | 76.2 | 93.8 | 87.6 | 80.2 | 93.3 |
| High Jump | 52.4 | 68.9 | 40.7 | 58.5 | 53.9 | 69.8 | 85.7 | 82.5 | 96.5 |
| Long Jump | 66.8 | 74.7 | 35.7 | 37.6 | 48.1 | 48.9 | 66.7 | 68.7 | 74.8 |
| Lay Up | 88.6 | 86.1 | 76.5 | 66.0 | 42.4 | 51.9 | 43.2 | 45.2 | 48.8 |

## 5.3 Activity Recognition with Attention

We introduce a multimodal deep neural network with our proposed feature attention and modality attention for activity recognition. There are three main steps for making online activity prediction: data pre-processing, feature extraction and fusion using attention, and finally decision making (Fig. 5.3).

### Data Pre-processing

The data pre-processing serves two purposes: formatting the different sensor data into a unified representation and preparing the data for attention generation. We chose to represent data into 3D matrices so that they can be processed using ConvNets and

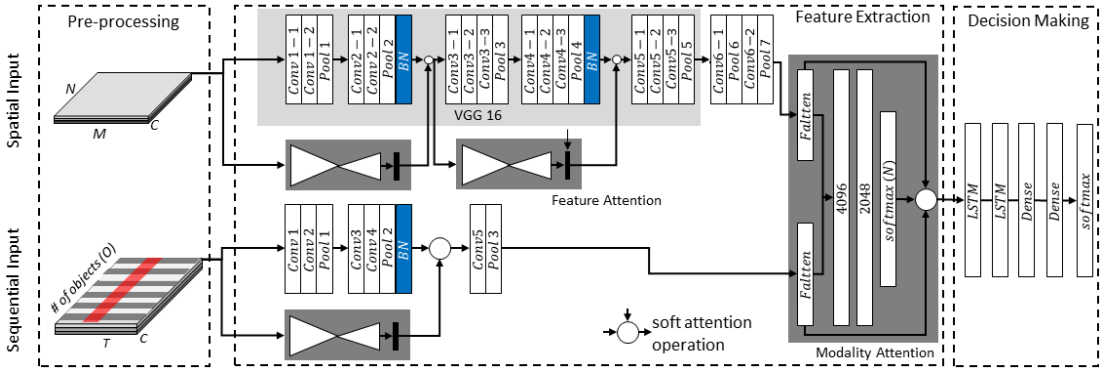


Figure 5.3: An overview of our multimodal attention network for activity recognition.

easily visualized. The visualizations of the attention are easy to interact with and will help us further tune the network.

We categorize raw data from different sensors in two ways: spatial and sequential. The spatial data ($M \times N \times C$) includes RGB images, depth images and optical flow, where there is a spatial association between data points. The $M \times N$ denotes the dimension of the input data and $C$ denotes the channel of the input data; for gray scale image $C = 1$ (Fig. 5.3, pre-processing top). The sequential data represents data with temporal associations between data points. Most mobile sensor data, such as 3-axis gyroscope, passive RFID data or audio data rec-orded by the microphone array, can be categorized as sequential data ($O \times T \times C$) [112]. Where $O$ denotes the number of sensors deployed(Fig. 5.3, pre-processing bottom) and $C$ denotes the sensor channel. For most sensors $C = 1$, gyroscopes have three channels, and microphone arrays may denotes the sensor channel (Fig. 5.3, pre-processing bottom).

### Feature Attention

We modified the residual attention module (Figure 1, Gray shaded block) [106] to accommodate the preprocessed spatial and sequential data structures.

The attention module consists of two parts: a conv-deconv network used to generate a set of attention maps in the convolutional layers, and a sigmoid layer that generates a single attention map from a set of activation maps in the last convolutional layer (Fig. 5.4). The soft-attention operation [106] was applied to use the learned attention map

as an indicator of important features while preserving the in-formation of the original features.

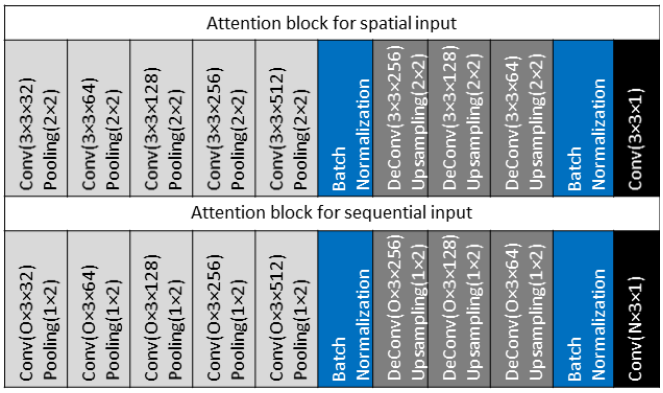


Figure 5.4: Feature attention to spatial and sequential data. $O$ denotes the number of objects.

Our proposed sequential attention slightly differs from existing spatial attention [106]. Sequential attention can be intuitively understood as a temporal window that indicates the importance of time instances within the window for activity recognition. The generated sequential attention mask should emphasize temporal associations and ignore spatial associations, so we fixed the size of the sequential attention mask to cover all objects ($O$) at any highlighted time (Fig. 5.3, pre-processing, red region). Note that there are other temporal attention frameworks for sequential data. Encoder-decoder structures [111] or RNNs with modified LSTM neurons [60] are commonly used for sequential data modeling. We didn't use those methods because they are difficult to visualize, and our system requires humans to determine the attention quality for later network tuning. With the introduced ConvNet attention module, we can easily tell if the model is putting attention on the correct spot by visualizing the generated attention map on the input im-age and then provide feedback for further network tuning.

### Modality Attention

Feature level fusion is a commonly used strategy to use features extracted from different input modalities together for activity recognition [70]]. But simply concatenating all

the extracted features together is not good enough, since the same sensor may have different contributions to recognizing different activities. For example, cameras might provide useful in-formation for sports activity recognition, but microphones are better at verbal activities when people remain in similar postures while talking. We proposed an intuitive inter-branch attention to learn modality dependency (Fig. 5.3, modality attention part). The modality attention module contains a fusion layer (fully connected layer), a score layer (softmax layer), and a linear combination layer between the original modality feature vectors and modality scores. The proposed modality attention is scalable to a large number of input modalities, and can be expressed as:

$$p = softmax(\sum_{i=1}^{K} \sigma(W_i \cdot z_i + b_i)) \tag{5.9}$$

where $p$ is a vector that denotes the importance score vector for all modalities, generated by the softmax regression. $K$ denotes the number of input branches, $\sigma(*)$ is the activation function for the fusion layer. $W_i, b_i$ indicates the weight matrix and bias vector for to the fusion layer respectively, and $z_i$ is the feature vector from the $i^th$ modality. Similarly, the output after applying modality attention is:

$$y = \sigma(\sum_{i=1}^{K} (1 + p_i) \cdot (W_i \cdot z_i + b_i)) \tag{5.10}$$

where p is a vector that denotes the importance score vector for all modalities. $\sigma(*)$ is the activation function. The $(1 + p_i) \cdot z_i$ term denotes the soft-attention applied to the input feature vector. The modality attention can be intuitively understood as the focusing on one or few types of sensors over others. Our experimental results on the 50-salads dataset [100] show that the modality attention brought around 3% mAP increase to the final activity recognition result.

### 5.3.1 Network Tuning

#### Tuning Motivation

The network can be trained in a supervised way, by tuning all of the weights in the network and using the loss from each instance. This leads to two problems: first, the supervised learning uses the activity recognition loss (categorical cross entropy) to tune

the weights associated with attention and recognition. But the activity label does not provide any information about the attention, so there is no guarantee that the attention module generates a reasonable attention map even though the final prediction results may be correct. Second, the observation might be not well-aligned with the activity ground truth for each time instance. The supervised learning may assign loss to each instance even if the instance contains no activity in progress. Addressing this issue, we introduce an asynchronous tuning strategy based on DQN.

**Asynchronous Tuning**

The weights in the proposed network can be roughly categorized into two types: associated with decision making (Fig. 5.3, unshaded part) and associated with attention generating (Figure 1 shaded part). We propose tuning these two types of weights separately. We started tuning the attention associated weights by setting the other weights untrainable. After tuning the attention module and obtaining preliminary attention generation, we stop attention training and start tuning the recognition parameters. The asynchronous tuning makes the activity recognition based on associated features by first tuning the system generate reasonable attention map. The tuning process can be repeated, and our experiments show that asynchronous tuning leads to around 3% accuracy gain.

As we argued, tuning attention module with activity ground truth labels is neither efficient nor accurate be-cause the activity labels contain no information about the attention region. We design a simple but efficient way to let humans provide usable feedback on generated attention maps (Fig. 5.5). The system will randomly sample the input data and generate attention maps. Because we used a ConvNet based attention module, it is easy to visualize and overlay the generated attention onto the original image and determine if the generated attention matches human experience. Humans can reject generated attention maps and provide their feedback by clicking on the original image. For each click, the system will generate a circle with an adjustable radius indicating the attention region provided by a human. If multiple clicks were made to a single image, the overlapped regions would not be summed up (Fig. 5.5 (a)).

Note that a single image can have multiple attention regions. For sequential data, we provided synchronized video frames so that one can mark important segments on the sequential attention map. The generated attention map for sequential data will cover all the object channels (Fig. 5.5 (b)).

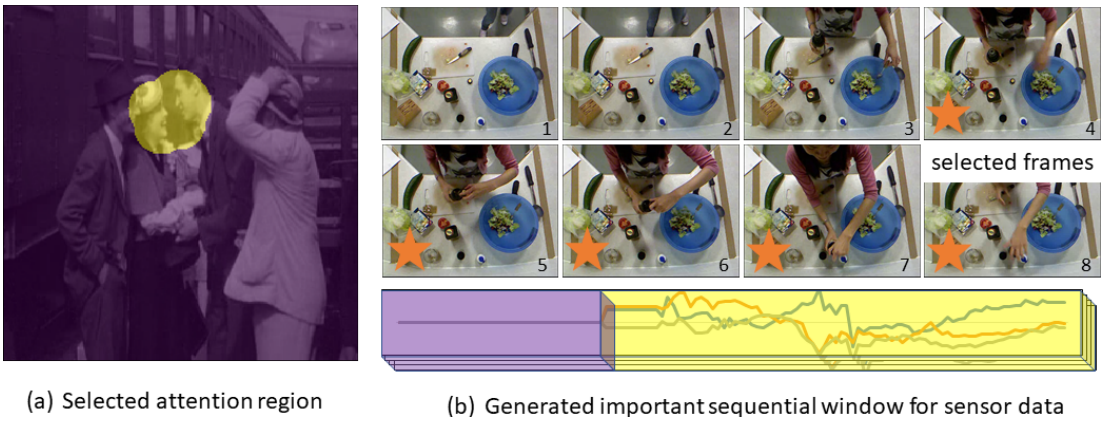


Figure 5.5: (a). the attention map for spatial data can be generated by clicking on the important region (b). the attention map for sequential data can be generated by selecting important frames.

### Tuning with DQN

One remaining problem is the misalignment between observations and ground truth labels. Supervised learning usually needs a label for each instance for loss calculation, which is often not available in published datasets. It is common to use the entire video's label as the label for each instance but doing this assumes each frame contains an activity (which is not often true). Addressing this issue, we still build the system to make online prediction but we only take the prediction from the last instance as the final activity recognition result for entire video. Instead of calculating the loss per-instance, we build the system to update the rewards for previous instances based on final prediction result strategically.

Our tuning can be then formulated as a DQN problem (Fig. 5.6). The activity recognition network with attention (Fig. 5.3) is the agent of our DQN problem that tries to make activity predictions (actions of our DQN). The quality of each action is

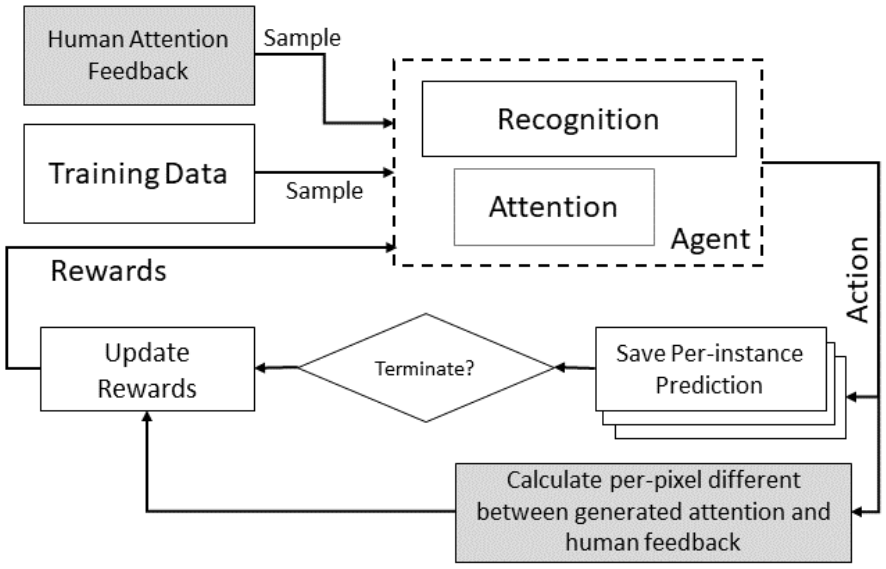


Figure 5.6: (a). the attention map for spatial data can be generated by clicking on the important region (b). the attention map for sequential data can be generated by selecting important frames.

determined on the final activity recognition result. The proposed DQN framework is different from the regular DQN framework in following aspects:

**The memory replay queue:** To avoid over-fitting and make training more efficient, the memory replay queue was introduced in [68, 67]. In our application, we directly used the training set for sampling. This is because unlike games that have unlimited number of states, the dataset with limited instances has a limited number of states.

**The rewards:** To avoid the influence from assigning labels to instances that have no activity in progress, we use rewards and penalties to represent the impact of the decision on each time instance. We use a queue to keep saving the Q values calculated from each time instance and update the rewards when the prediction of the last instance is made. The loss (categorical cross-entropy) is:

$$Loss = -\sum S(r + r' + \gamma \hat{Q_{sa}}) log^{Q_{sa}} \tag{5.11}$$

where $r$ denotes the rewards and $r'$ denotes the additional rewards we defined. $\hat{Q_{sa}}$ is the predicted action-value vector and $Q_sa$ is the actual action-value vector (softmax

output). Because we used cross entropy as loss function which requires $(r+r'+\gamma\hat{Q_{sa}}) \in [0,1]$, we defined a rectification function:

$$S(x) = min(max(x, 0), 1) \tag{5.12}$$

If the final prediction is correct, we don't penalize the previous incorrectly predicted instances, since we don't know if activity actually happens in those time instances. Similarly, we only penalize the incorrectly predicted instances if the final prediction result is incorrect. Given a sample with N instances, we assign rewards as:

$$r_i = \begin{cases} 0.1, & \text{if } p_i = landp_N = landj = argmax(Q_{sa}). \\ 0, & \text{if } p_i \neq landp_N = landj = argmax(Q_{sa}). \\ 0, & \text{if } p_i = landp_N \neq landj = argmax(Q_{sa}). \\ -0.1, & \text{if } p_i \neq landp_N \neq landj = argmax(Q_{sa}). \\ 0, & \text{otherwise} \end{cases} \tag{5.13}$$

where $r_i$ denotes the rewards for the $i^th$ instance at $j^th$ label, $p_i$ is the prediction for $i^th$ instance. The $l$ is the ground truth label and $p_N$ is the prediction of last instance.

We defined an additional reward to take the quality of generated attention maps into consideration. For each DQN training step, the network will randomly sample a batch of manually generated attention maps and compare them with machine-generated attention maps. We used the intersection over union (IoU) to measure the similarity of these attention maps and set 0.5 as the threshold. The $r'$ is defined as:

$$r' = \begin{cases} 0.5 - IoU, & j = argmax(Q_{sa}). \\ 0, & \text{otherwise} \end{cases} \tag{5.14}$$

**The termination:** The current DQN training loop can be terminated if the prediction from the last time instance do not match the ground truth. We will assign a large penalty (-1 in our experiments) to the network and fine-tune the weights.

### Implementation

We implemented the network with Keras and TensorFlow backend. Because we are using the video as input, and we have to maintain a reasonably large batch size to

have enough diversity within each batch, we down-sampled the video to 5fps. The network was trained on dual GTX 1080TI GPU and 32GB RAM. We used ReLU activation for convolutional layers and used the Adam optimizer. To avoid overfitting and boost training, we used batch normalization after each of the convolutional layers, and also adopted dropout after fully connected layers. We initialized spatial branch with pre-trained VGG weights, and randomly initialized the sequential branch weights. The model is first trained with supervised manner. When the training accuracy stops growing, we started tuning with proposed method..

### 5.3.2 Experimental Results

#### Dataset

We used three commonly used datasets for experiments. The Olympic sports dataset [72] contains 16 activities. We used the official training and testing split to make comparison fair. Hollywood 2 datasets [70] contain 12 classes of human actions over 3669 video clips and approximately 20.1 hours of video in total, we used the official training and testing split. We also tested our system with the 50-salads dataset [70] which contains multi-sensory data from RGB-D camera and 3-axis gyroscope. We ran our experiments on the most challenging 17-label setting.

#### Activity Recognition Performance

We achieved the 0.796 mAP on Olympic sports dataset, outperforming the state-of-the-art online recognition system [113] on the same dataset. We noticed that some offline systems [21, 58] achieved better performance on the same dataset, but they require global feature extraction before recognition, which makes their application limited. On the Hollywood2 dataset, our system achieved 0.582 mAP without using proposed tuning and 0.631 mAP with proposed tuning, which is comparable to state-of-the-art online systems [65, 104]. Some research demonstrated that combining the global feature achieved the better performance than our system [104], but the global feature also makes the activity recognition offline. Our system can use stacked dilated convolution

layers over time or use global feature as input for our system as other applications for performance boosting [59, 19]. We further tested our system on the 50 salads dataset with both RGB-D input and sensor data, demonstrating our system with similar input modalities and network structure outperformed other systems [47, 45] on 17 activities (Table 5.5). The system is also able to beat other systems [47, 45] on the 7-activity setting.

We made several observations worth mentioning: 1. For each dataset, we compared the performance using CNN-LSTM (Fig. 5.3, unshaded part), the proposed net-work with attention only, and the proposed network with proposed tuning. The experimental results show that the attention brings about 11% mAP gain to the CNN-LSTM network. The tuning will further bring about 8% accuracy gain, exemplifying the advantages of our proposed attention framework and tuning. 2. For the 50 salads dataset, we tried to implement the fusion layer with and without modality attention. Our experiments show that the system using the proposed modality attention achieves around 3% higher mAP. 3. we also studied the impact of the number of human attention feedbacks on the final activity recognition results. Our experiments show that labeling 10% of the dataset is sufficient enough for tuning. More manually labeled data does not lead to significant performance boost.

Table 5.5: Activity recognition system performance comparison on Olympic sport, 50 salads and Hollywood 2 dataset. The shaded systems require offline feature processing.

| Olympic Sports | | 50 Salads | | | Hollywood 2 | | |
|---|---|---|---|---|---|---|---|
| *system* | *mAP* | *System* | *mAP* | *Acc.* | *System* | *mAP* | *Acc.* |
| CNN-LSTM | 0.613 | CNN-LSTM (all sensory) | 0.271 | na | CNN-LSTM | 0.441 | 0.473 |
| Ref. [43] | 0.620 | Ref. [86] | 0.379 | 0.542 | Ref. [59] | na | 0.785 |
| Ref. [113] | 0.764 | Ref. [46] | 0.579 | 0.597 | Ref. [104] | 0.563 | na |
| Ref. [21] | 0.855 | Ref. [85] | na | 0.575 | Ref. [19] | na | 0.767 |
| Ref. [58] | 0.966 | Ours (video only)* | 0.382 | 0.413 | Ref. [65] | na | 0.654 |
| Ours* | 0.686 | Ours (all sensory)* | 0.411 | 0.476 | Ours* | 0.582 | 0.625 |
| Ours | 0.796 | Ours (all sensory) | 0.449 | 0.501 | Ours | 0.631 | 0.689 |

We further tested the generalizability of our system using sports video clips we

collected ourselves on campus, where sports activities similar to those in the training set occur on very different backgrounds. We compared the performance of our model with and without our proposed tuning. The network without tuning suffered around a 19% performance drop, while the network with tuning only suffered 6%. This is because our proposed asynchronous DQN tuning encourages the system to focus on the associated features. We further visualized the generated attention maps outputted from the sigmoid layer (Fig. 5.3, pointed by black arrow), and the results clearly show that our proposed tuning helps put attention on the representative features (Fig. 5.7). Because we resized the generated attention map from $16 \times 16$ to $256 \times 256$ for the image overlay visualization, some of the attention does not perfectly cover the associated features. The generated attention map can be further used to define a bounding box indicating the activity location.

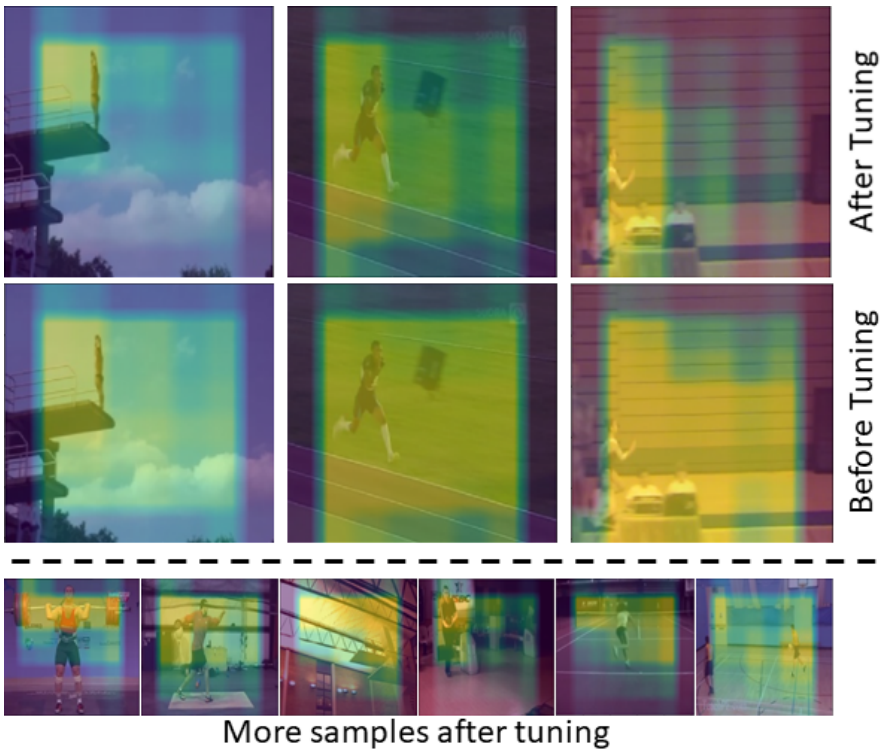


Figure 5.7: The generated attention map for Olympic sports activities after tuning.

As we argued, the DQN-based training does not assign additional loss to the instances without an activity in progress to avoid overfitting to background features.

The system should have high softmax scores when representative features appear, and have low softmax scores for instances with no activity in progress. Based on this, we can predict if there is an activity in progress by thresholding the softmax score. The experimental results (using 0.5 as threshold) demonstrate that when representative features are present, the system can make predictions with high confidence. However, when there is no activity being observed, the system remains uncertain about the prediction results (Fig. 5.8).


Figure 5.8: The system can predict activities with high confidence when representative features are present and maintains uncertainty when there is no activity (see digital version for better resolution).

## 5.4 Summary

We presented two applications of our proposed system: activity recognition. Our framework, however, is generalizable to a wider range of activity recognition problems. The idea of generating a mask can be compared to region proposal, which was used for target recognition and tracking. Instead of generating the activity-location or team-role masks, we could generate masks for any people or objects. One could also generate the masks to distinguish different activity performers and achieve concurrent activity

recognition together with their locations. We further introduced the asynchronous tuning and DQN based tuning to boost the training process and lead to better recognition performance.

# Chapter 6

# Concurrent Activity Recognition (Future Work)

## 6.1 Motivation

Concurrent activities are common, our analysis of the 42 trauma resuscitations dataset showed that more than 50% of time instances had at least two concurrent activities. Even for daily living scenarios (Charades dataset) there were more than 70% time instances with at least two concurrent activities. The challenge is: With $N$ activities, there are $N$ labels for individual-activity prediction but $2^N$ potential combinations of concurrent activities. An efficient classifier structure is needed for concurrent activity prediction.

## 6.2 Encoder-Decoder for Concurrent Activity Recognition

### 6.2.1 System Overview

Our system consists of two main component: the encoder and decoder. Because the activity is a continuous concept, and the observation of an activity at single time instance can be misleading, we used a small sliding time-window (10 seconds in this paper) to sample the collected data for activity recognition and make predicntions based on the observation over the time windows. The set the step of the time window as one-second so that the system is still able to make prediction online. The encoder consists of two modules, the feature extraction and temporal association encoder (Fig. 6.1). The feature extraction module extract features from different input modality using ConvNet and attention designed for different types of data. The LSTM layer is used to further combine the extracted feature over the time-window into a single feature vector. The generated vector is considered contains all the important spatial-temporal feature for

activity recognition.

The decoder is a set of stacked LSTM layers (Fig. 6.1) that make a prediction for the activities one at a time. In this way, the association of activities is considered for activity prediction.

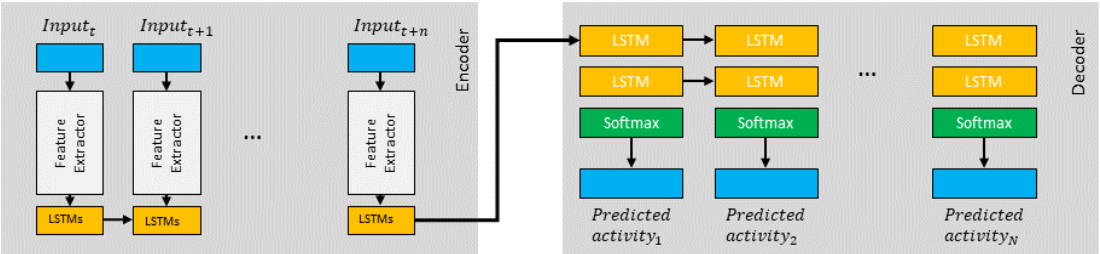


Figure 6.1: The encoder-decoder framework.

### 6.2.2 Encoder

#### Input Pre-processing

Before feature extraction, we have to pre-process the data collected from different sensory. The data pre-processing serves two purposes: formatting the different sensor data into a unified representation and preparing the data for attention generation. We chose to represent data into 3D matrices so that they can be processed using ConvNets and easily visualized.

We represent the spatial data includes RGB images, depth images, and optical flow, where there is a spatial association between data points as a 3D matrix ($M \times N \times C$). The $M \times N$ denotes the dimension of the input data and C denotes the channel of the input data; for gray scale image $C = 1$ (Fig. 6.2, pre-processing top). For the sensor data such as passive RFID used in our research, we converted the received RFID raw data as RSS map. In this way, the sensor data is represented as a 3D matrix and can be processed with the ConvNet with attention as well.

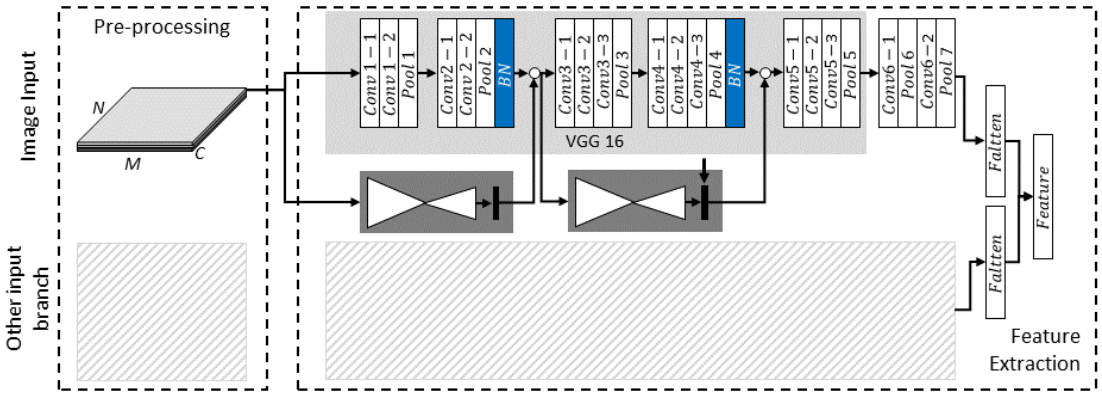


Figure 6.2: Multimodal structure with introduced feature attention.

### Feature Attention

We modified the residual attention module [19] to accommodate the preprocessed spatial and sequential data structures (Fig. 6.2, Gray shaded block).

The attention module consists of two parts: a conv-deconv network used to generate a set of attention maps in the convolutional layers, and a sigmoid layer that generates a single attention map from a set of activation maps in the last convolutional layer (Fig. 6.2). The soft-attention operation was applied to use the learned attention map as an indicator of important features while preserving the information of the original features. Because recognizing each of the simultaneous happened activities require the attention located at the different spot of the input space, it is critical to tuning the attention module and ensures the generated attention matches the human experience. With the introduced ConvNet attention module, we can easily tell if the model is putting attention on the correct spot by visualizing the generated attention map on the input image and then provide feedback for further network tuning. The feature level fusion is implemented to combine the extracted feature from different input modality.

### Temporal Encoding

As we argued that activity recognition requires temporal association between observations over time, and several strategies were proposed to build the temporal association

between observations. The slow fusion strategy stacks the convolutional layers overtime, and 3D convolutional based strategy suggest that it is possible to directly perform convolution in 3D to extract spatial and temporal feature simultaneously. Though proved to be applicable, the slow fusion and 3D convolution have a very redundant representation fo temporal association, which is hard to generalize and not scale well to large time-window. Most recent research on video to text translation and language translation, a LSTM based temporal encoder is widely used, such framework is light-weight, and LSTM is able to handle the temporal association more efficiently than ConvNet due to its unique gates system. We adopted the same structure to encode the extracted feature vector over time windows.

### 6.2.3 Decoder

Different from the single activity recognition that each activity decision is only based on observation, we noticed that the activity combination also provide additional information for concurrent activity recognition. For example, if a person is walking, he cannot be running at the same time. Therefore, we need a strategy to make the prediction not only based on extracted features but also based on the logic and associations between different activities.

The video to text translation or language faced the similar challenge since the words in the generated sentence have to deliver the correct meaning while maintaining the correct order. We used the bi-directional LSTM to make the concurrent activity recognition based on both extracted feature and association between activities. Because we are doing activity recognition, the softmax layer is used for activity prediction.

## 6.3 Network Training and Tuning

### 6.3.1 Tuning Motivation

Our proposed model can be trained in supervised manner. The supervised learning use cross entropy as loss for network tuning. One problem is that the supervised learning is sensitive to the data imbalance, the system tend to make positive predictions for

labels with higher occurrence and negative prediction for labels with lower occurrence to minimize the loss. Our statistics show that the duration of different concurrent activities are different, the long activity's (play) duration is 100 times compared with short activity (goal) on hokey dataset. The accuracy is not a good metric to measure the quality of system prediction if the dataset is imbalance and the cross entropy based loss can not well represent the overall performance of the system.

The precision and recall is much better measurement, but the overall precision and recall cannot be measured before the prediction of a video is finished. we propose a method that first train the network with supervised method, once the prediction for a full video is generated, we further tune the weights based on overall performance. During tuning, we want to penalize the network (by assign a big loss) based on both short term loss (the per-instance prediction results) and long term loss (the F score for entire video).

### 6.3.2 Network Tuning with DQN

Our tuning can be formulated as a DQN problem (Fig. 6.3) . The activity recognition network with encoder and decoder (Fig. 6.1) is the agent of our DQN problem that tries to make activity predictions (actions of our DQN). Instead of trying to minimize the loss in a supervised manner, we defined the rewards and tuned the network in reinforcement learning way. The rewards are defined based on both short-term per-frame prediction results and long-term F-score for each case. (Fig. 6.3). Our DQN for tuning follows the general rules of regular DQN, but it is Unique in following ways:

**The Loss:** Because the ultimate goal of our system is to make activity recognition, instead of using the linear layer as the last layer for Q-value, we kept the softmax layer and used the softmax score as Q value. This design enables us to directly "attach" the DQN module to the per-trained network for tuning without add and dropping any layers. To better work with the softmax layer, we used the categorical cross entropy as loss to tuning. The loss (categorical cross-entropy) is:

$$Loss = -\sum S(r + r' + \gamma \hat{Q_{sa}}) log^{Q_{sa}} \tag{6.1}$$

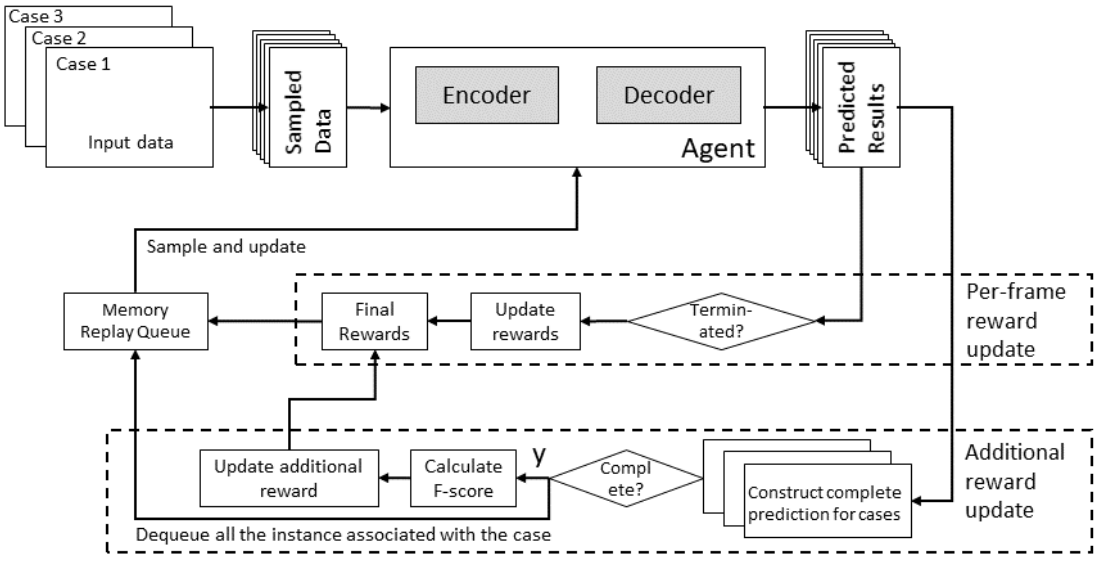


Figure 6.3: The overall diagram of our DQN based tuning.

where $r$ denotes the rewards vector defined by the per-frame prediction results and $r'$ is the additional rewards vector denotes the rewards based on long-term prediction results. $\hat{Q_{sa}}$ is the predicted action-value vector and $Q_{sa}$ is the actual action-value vector (softmax output). Because we used cross entropy as loss function which requires $(r + r' + \gamma\hat{Q_{sa}}) \in [0, 1]$, we defined a rectification function:

$$S(x) = min(max(x, 0), 1) \tag{6.2}$$

**The additional rewards:** As we argued that use the per-frame prediction results for loss calculation is bias especially when the data is imbalanced. We introduced the additional rewards that is estimated based on the F-score estimated based on the complete prediction on each case:

$$r'_{ij} = F1_{ij} - E_j \tag{6.3}$$

where $r_{ij}$ denotes the rewards for the $i^th$ case at $j^th$ label, $F1_{ij}$ is the F1 score calculated based on the prediction for $i^th$ case for activity $j$. The $E_i$ denotes the expected F1 score for at $j^{th}$ label which can be manually determined based on different dataset. The additional rewards has positive value if the system is able to make good prediction

that lead to better F1 score than the expected F1 score. Otherwise, the additional rewards turns to negative value which penalize the system.

The additional rewards is considered as long-term rewards denotes the overall performance of the system on entire case instead of at each time instance.

**The F1 score calculation:** Calculate the F1 score requires the complete prediction results for each case, but when tuning the network, we used the clips randomly selected from different cases to provide enough diversity for each batch. We build a buffer array to keep buffer the predicted data from each batch belongs to different cases . When a buffer is filled with data, we take all the predicted data out of buffer and calculate the F-score for each of the activity in this case and clear the buffer. The F1 score was calculate the for each of the activities.

**The memory replay queue:** To avoid over-fitting and make training more efficient, the memory replay queue was introduced in [68, 67]. We adopted the idea of memory replay queue with a unique dequeue strategy. When the F1 score is updated for a case, all the instances that are associated with the case will be dequeued and later replaced with new data.

**The termination:** The current DQN training loop can be terminated if the accuracy is low than a certain threshold, in our case the threshold is set to 0.8 due to the dataset contains very imbalanced data.

### 6.3.3 Implementation

The network was implemented with Keras using TensorFlow as back-end. We down-sampled the video to 5fps to reduce the calculation without compromise the performance. To deliver enough diversity to each mini-batch, we cut each video into 30 frame clips and randomly sample the clips over the videos. The batch normalization was used after each of the convolutional layers, and also adopted dropout after fully connected layers. The network was trained on dual GTX 1080TI GPU and 32GB RAM. We used ReLU activation for convolutional layers and used the Adam optimizer.

## 6.4 Preliminary Results

We applied our model on CelebA dataset. CelebA [61] is a large-scale face attributes dataset which includes more than 200K images of human faces. There are 40 attribute notes in total and each face will contain one or more than one attributes. We used the official training-testing split to compare with previous research. Although it is not an activity dataset, it is a well-known multi-label dataset. We implemented our system on it to demonstrate our proposed encoder-decoder can be generalized to multi-label classification problems.

Our system achieved the state-of-the-art performance compared with most recent published papers use the same dataset (Table 6.1).

Table 6.1: Experimental results and comparison on CelebA dataset.

| System | Accuracy | Precision (Top 10) | Recall (Top 10) |
|---|---|---|---|
| ref. [108] | 0.88 | na | na |
| ref. [89] | 0.91 | na | na |
| ref. [28] | 0.91 | na | na |
| VGG (without attention) | 0.91 | na | 0.74 |
| VGG (with attention) | 0.87 | 0.87 | 0.87 |
| ref. [63] | 0.91 | na | 0.71 |
| Our model | 0.91 | 0.93 | 0.93 |

## 6.5 Future Work

We demonstrate the proposed encoder-decoder framework with tuning works for single image based multi-label classification problems. Our future work will be:

1. **More testing:** implementing our system on more real-world concurrent recognition datasets to further test our system performance and further improve it.

2. **Better tuning:** better tuning the attention module with human feedback on generated attention map.

3. **better tuning strategy:** although the introduced DQN based tuning works for concurrent activity recognition, the DQN based tuning is slow. This is partly

because we used the categorical cross-entropy as the loss. Design the new tuning strategy or training strategy for better training performance and efficiency will be our future work.

# Chapter 7

# Conclusion

Due to its high application value, the activity recognition is still one of the most heated discussed research topic. We start from high-level activity recognition, process progress estimation and further introduced the low-level activity recognition.

For high-level (process phase) activity recognition, We introduce a deep regression based linear process progress estimation strategy and associated application in an actual trauma room. The proposed system is able to continuously estimate the overall completeness of a event, the remaining-time for process to complete. Our contribution on high-level activity recognition can be summarized as:

1. **A novel deep regression-based approach** for process progress estimation and phase detection using commercially available sensors, as well as a new rectified hyperbolic tangent (*rtanh*) activation function that bounds the regression value to a meaningful range and accelerates the neural network training.

2. **A GMM-based phase detection approach** based on completeness regression results, as well as a conditional loss function for model tuning using regression and classification error.

3. **A deep learning structure** that models the progress of *nonlinear* processes, which is derived from our structure for *linear* process modeling.

For low level (fine-level) activity recognition, we started with single sensor based low level activity recognition systems. We studied the passive RFID based activity recognition using both shallow and deep models. We are the first to use deep learning for RFID data based activity recognition. We further studied the activity recognition using video and audio as input with deep learning based approaches. To make system

aware the important features and ignore irrelevant features, we proposed the cGAN and attention based framework. Our future work will focus on concurrent activity recognition. Our contribution can be summarized as:

1. **Object tagging strategies and features combinations for object-use classification.**

2. **A deep learning model for activity recognition in complex teamwork based on passive RFID:** We developed a system for complex activity recognition from RFID data using a deep convolutional neural network. Unlike existing systems that rely on manufactured features and a cascade structure with object-use detection followed by activity recognition, our system works directly with RFID data and performs multiclass classification of activities or process phases.

3. **A novel approach for activity recognition** that first generates activity masks to provide additional information for subsequent activity recognition. The system uses the generated masks as an additional color channel of the original input image to perform per-frame activity recognition and find the activity's bounding box.

4. **The Conv-LSTM-Deconv generator for cGAN** used to generate an output sequence of video masks based on the spatio-temporal associations in a sequence of input video frames. Our **spatio-temporal loss function** can tune the mask generator based on pixel-wise errors of the generated masks and discontinuities in the mask sequence.

5. **A cascade-structure mask generator** that generates the activity mask before the activity is recognized using a VGG-LSTM network. We introduced **activity recognition loss** to tune the weights of the conditional GAN generator based

6. **A reinforcement learning framework for attention fine tuning** that can be easily attached to ConvNet-based attention models.

# References


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

[2] Abdel-Hamid, O., Deng, L., and Yu, D. Exploring convolutional neural network structures and optimization techniques for speech recognition. In *Interspeech* (2013), pp. 3366–3370.

[3] Abdel-Hamid, O., Mohamed, A.-r., Jiang, H., Deng, L., Penn, G., and Yu, D. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on audio, speech, and language processing 22*, 10 (2014), 1533–1545.

[4] Alsheikh, M. A., Niyato, D., Lin, S., Tan, H.-P., and Han, Z. Mobile big data analytics using deep learning and apache spark. *IEEE network 30*, 3 (2016), 22–29.

[5] Altmann, A., Toloşi, L., Sander, O., and Lengauer, T. Permutation importance: a corrected feature importance measure. *Bioinformatics 26*, 10 (2010), 1340–1347.

[6] Bardram, J. E., Doryab, A., Jensen, R. M., Lange, P. M., Nielsen, K. L., and Petersen, S. T. Phase recognition during surgical procedures using embedded and body-worn sensors. In *Pervasive Computing and Communications (PerCom), 2011 IEEE International Conference on* (2011), IEEE, pp. 45–53.

[7] Barga, R., Fontama, V., Tok, W. H., and Cabrera-Cordon, L. *Predictive analytics with Microsoft Azure machine learning*. Springer, 2015.

[8] Bilmes, J. A., et al. A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. *International Computer Science Institute 4*, 510 (1998), 126.

[9] Blum, T., Feussner, H., and Navab, N. Modeling and segmentation of surgical workflow from laparoscopic video. *Medical image computing and computer-assisted intervention–MICCAI 2010* (2010), 400–407.

[10] Blum, T., Padoy, N., Feussner, H., and Navab, N. Modeling and online recognition of surgical phases using hidden markov models. In *International Conference on Medical Image Computing and Computer-Assisted Intervention* (2008), Springer, pp. 627–635.

[11] Catal, C., Tufekci, S., Pirmit, E., and Kocabag, G. On the use of ensemble of classifiers for accelerometer-based activity recognition. *Applied Soft Computing 37* (2015), 1018–1022.

[12] Chen, Y., and Xue, Y. A deep learning approach to human activity recognition based on single accelerometer. In *Systems, Man, and Cybernetics (SMC), 2015 IEEE International Conference on* (2015), IEEE, pp. 1488–1492.

[13] Choudhury, T., Consolvo, S., Harrison, B., Hightower, J., LaMarca, A., LeGrand, L., Rahimi, A., Rea, A., Bordello, G., Hemingway, B., et al. The mobile sensing platform: An embedded activity recognition system. *IEEE Pervasive Computing 7*, 2 (2008).

[14] Csurka, G., Larlus, D., Perronnin, F., and Meylan, F. What is a good evaluation measure for semantic segmentation. In *BMVC* (2013), vol. 27, Citeseer, p. 2013.

[15] De, D., Bharti, P., Das, S. K., and Chellappan, S. Multimodal wearable sensing for fine-grained activity recognition in healthcare. *IEEE Internet Computing 19*, 5 (2015), 26–35.

[16] Deng, Z., Vahdat, A., Hu, H., and Mori, G. Structure inference machines: Recurrent neural networks for analyzing relations in group activity recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 4772–4781.

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

[18] Ding, H., Han, J., Liu, A. X., Zhao, J., Yang, P., Xi, W., and Jiang, Z. Human object estimation via backscattered radio frequency signal. In *Computer Communications (INFOCOM), 2015 IEEE Conference on* (2015), IEEE, pp. 1652–1660.

[19] Fernando, B., Anderson, P., Hutter, M., and Gould, S. Discriminative hierarchical rank pooling for activity recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 1924–1932.

[20] Forestier, G., Riffaud, L., and Jannin, P. Automatic phase prediction from low-level surgical activities. *International journal of computer assisted radiology and surgery 10*, 6 (2015), 833–841.

[21] Gaidon, A., Harchaoui, Z., and Schmid, C. Activity representation with motion hierarchies. *International journal of computer vision 107*, 3 (2014), 219–238.

[22] Gers, F. A., Schmidhuber, J., and Cummins, F. Learning to forget: Continual prediction with lstm.

[23] Girshick, R. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision* (2015), pp. 1440–1448.

[24] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in neural information processing systems* (2014), pp. 2672–2680.

[25] Graves, A., Mohamed, A.-r., and Hinton, G. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on* (2013), IEEE, pp. 6645–6649.

[26] Hammoud, R. I., Sahin, C. S., Blasch, E. P., and Rhodes, B. J. Multi-source multi-modal activity recognition in aerial video surveillance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (2014), pp. 237–244.

[27] Han, J., Ding, H., Qian, C., Xi, W., Wang, Z., Jiang, Z., Shangguan, L., and Zhao, J. Cbid: A customer behavior identification system using passive tags. *IEEE/ACM Transactions on Networking 24*, 5 (2016), 2885–2898.

[28] Hand, E. M., and Chellappa, R. Attributes for improved attributes: A multi-task network utilizing implicit and explicit relationships for facial attribute classification. In *AAAI* (2017), pp. 4068–4074.

[29] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 770–778.

[30] Herath, S., Harandi, M., and Porikli, F. Going deeper into action recognition: A survey. *Image and Vision Computing 60* (2017), 4–21.

[31] Herbst, J., and Karagiannis, D. Integrating machine learning and workflow management to support acquisition and adaptation of workflow models. In *Database and Expert Systems Applications, 1998. Proceedings. Ninth International Workshop on* (1998), IEEE, pp. 745–752.

[32] Hochreiter, S. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 6*, 02 (1998), 107–116.

[33] Hochreiter, S., and Schmidhuber, J. Long short-term memory. *Neural computation 9*, 8 (1997), 1735–1780.

[34] Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. Image-to-image translation with conditional adversarial networks. *arXiv preprint arXiv:1611.07004* (2016).

[35] Jiang, Y.-G., Dai, Q., Liu, W., Xue, X., and Ngo, C.-W. Human action recognition in unconstrained videos by explicit motion modeling. *IEEE Transactions on Image Processing 24*, 11 (2015), 3781–3795.

[36] Johnson, J., Alahi, A., and Fei-Fei, L. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision* (2016), Springer, pp. 694–711.

[37] Karpathy, A., Johnson, J., and Fei-Fei, L. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078* (2015).

[38] Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., and Fei-Fei, L. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition* (2014), pp. 1725–1732.

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

[40] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (2012), pp. 1097–1105.

[41] Lane, N. D., and Georgiev, P. Can deep learning revolutionize mobile sensing? In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications* (2015), ACM, pp. 117–122.

[42] Lane, N. D., Georgiev, P., and Qendro, L. Deepear: robust smartphone audio sensing in unconstrained acoustic environments using deep learning. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (2015), ACM, pp. 283–294.

[43] Laptev, I., Marszalek, M., Schmid, C., and Rozenfeld, B. Learning realistic human actions from movies. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on* (2008), IEEE, pp. 1–8.

[44] Lara, O. D., and Labrador, M. A. A survey on human activity recognition using wearable sensors. *IEEE Communications Surveys and Tutorials 15*, 3 (2013), 1192–1209.

[45] Lea, C., Flynn, M. D., Vidal, R., Reiter, A., and Hager, G. D. Temporal convolutional networks for action segmentation and detection. *arXiv preprint arXiv:1611.05267* (2016).

[46] Lea, C., Flynn, M. D., Vidal, R., Reiter, A., and Hager, G. D. Temporal convolutional networks for action segmentation and detection. *arXiv preprint arXiv:1611.05267* (2016).

[47] Lea, C., Vidal, R., Reiter, A., and Hager, G. D. Temporal convolutional networks: A unified approach to action segmentation. In *Computer Vision–ECCV 2016 Workshops* (2016), Springer, pp. 47–54.

[48] LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *Nature 521*, 7553 (2015), 436–444.

[49] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE 86*, 11 (1998), 2278–2324.

[50] Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al. Photo-realistic single image super-resolution using a generative adversarial network. *arXiv preprint arXiv:1609.04802* (2016).

[51] Li, H., Ye, C., and Sample, A. P. Idsense: A human object interaction detection system based on passive uhf rfid. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (2015), ACM, pp. 2555–2564.

[52] Li, X., Yao, D., Pan, X., Johannaman, J., Yang, J., Webman, R., Sarcevic, A., Marsic, I., and Burd, R. S. Activity recognition for medical teamwork based on passive rfid. In *RFID (RFID), 2016 IEEE International Conference on* (2016), IEEE, pp. 1–9.

[53] Li, X., Zhang, Y., Li, M., Chen, S., Austin, F. R., Marsic, I., and Burd, R. S. Online process phase detection using multimodal deep learning. In *Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), IEEE Annual* (2016), IEEE, pp. 1–7.

[54] Li, X., Zhang, Y., Li, M., Marsic, I., Yang, J., and Burd, R. S. Deep neural network for rfid-based activity recognition. In *Proceedings of the Eighth Wireless of the Students, by the Students, and for the Students Workshop* (2016), ACM, pp. 24–26.

[55] Li, X., Zhang, Y., Marsic, I., Sarcevic, A., and Burd, R. S. Deep learning for rfid-based activity recognition. In *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM* (2016), ACM, pp. 164–175.

[56] Li, X., Zhang, Y., Zhang, J., Chen, S., Marsic, I., Farneth, R. A., and Burd, R. S. Concurrent activity recognition with multimodal cnn-lstm structure. *arXiv preprint arXiv:1702.01638* (2017).

[57] Li, X., Zhang, Y., Zhang, J., Chen, Y., Li, H., Marsic, I., and Burd, R. S. Region-based activity recognition using conditional gan. In *Proceedings of the 2017 ACM on Multimedia Conference* (2017), ACM, pp. 1059–1067.

[58] Li, Y., Li, W., Mahadevan, V., and Vasconcelos, N. Vlad3: Encoding dynamics of deep features for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 1951–1960.

[59] Liu, A.-A., Su, Y.-T., Nie, W.-Z., and Kankanhalli, M. Hierarchical clustering multi-task learning for joint human action grouping and recognition. *IEEE transactions on pattern analysis and machine intelligence 39*, 1 (2017), 102–114.

[60] Liu, J., Wang, G., Hu, P., Duan, L.-Y., and Kot, A. C. Global context-aware attention lstm networks for 3d action recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), vol. 7.

[61] Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In *Proceedings of the IEEE International Conference on Computer Vision* (2015), pp. 3730–3738.

[62] Long, J., Shelhamer, E., and Darrell, T. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 3431–3440.

[63] Lu, Y., Kumar, A., Zhai, S., Cheng, Y., Javidi, T., and Feris, R. Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. *arXiv preprint arXiv:1611.05377* (2016).

[64] Luc, P., Couprie, C., Chintala, S., and Verbeek, J. Semantic segmentation using adversarial networks. *arXiv preprint arXiv:1611.08408* (2016).

[65] Marszalek, M., Laptev, I., and Schmid, C. Actions in context. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on* (2009), IEEE, pp. 2929–2936.

[66] Mirza, M., and Osindero, S. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* (2014).

[67] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

[68] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature 518*, 7540 (2015), 529–533.

[69] Mund, S. *Microsoft azure machine learning*. Packt Publishing Ltd, 2015.

[70] Ngiam, J., Khosla, A., Kim, M., Nam, J., Lee, H., and Ng, A. Y. Multimodal deep learning. In *Proceedings of the 28th international conference on machine learning (ICML-11)* (2011), pp. 689–696.

[71] Ni, R., Ye, Z., Feng, Q., and Zhang, J. Zoning positioning model based on minimize rfid reader. In *Instrumentation and Measurement, Sensor Network and Automation (IMSNA), 2013 2nd International Symposium on* (2013), IEEE, pp. 117–121.

[72] Niebles, J. C., Chen, C.-W., and Fei-Fei, L. Modeling temporal structure of decomposable motion segments for activity classification. In *European conference on computer vision* (2010), Springer, pp. 392–405.

[73] Ordóñez, F. J., and Roggen, D. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors 16*, 1 (2016), 115.

[74] Oreifej, O., and Liu, Z. Hon4d: Histogram of oriented 4d normals for activity recognition from depth sequences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2013), pp. 716–723.

[75] Padoy, N., Blum, T., Ahmadi, S.-A., Feussner, H., Berger, M.-O., and Navab, N. Statistical modeling and recognition of surgical workflow. *Medical image analysis 16*, 3 (2012), 632–641.

[76] Palmes, P., Pung, H. K., Gu, T., Xue, W., and Chen, S. Object relevance weight pattern mining for activity recognition and segmentation. *Pervasive and Mobile Computing 6*, 1 (2010), 43–57.

[77] Parlak, S., Marsic, I., and Burd, R. S. Activity recognition for emergency care using rfid. In *Proceedings of the 6th International Conference on Body Area Networks* (2011), ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), pp. 40–46.

[78] Parlak, S., Marsic, I., Sarcevic, A., Bajwa, W. U., Waterhouse, L. J., and Burd, R. S. Passive rfid for object and use detection during trauma resuscitation. *IEEE Transactions on Mobile Computing 15*, 4 (2016), 924–937.

[79] Philipose, M., Fishkin, K. P., Perkowitz, M., Patterson, D. J., Fox, D., Kautz, H., and Hahnel, D. Inferring activities from interactions with objects. *IEEE pervasive computing 3*, 4 (2004), 50–57.

[80] Piyathilaka, L., and Kodagoda, S. Gaussian mixture based hmm for human daily activity recognition using 3d skeleton features. In *Industrial Electronics and Applications (ICIEA), 2013 8th IEEE Conference on* (2013), IEEE, pp. 567–572.

[81] Powers, D. M. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation.

[82] Press, W. H. *The art of scientific computing*. Cambridge university press, 1992.

[83] Primus, M. J., Schoeffmann, K., and Böszörmenyi, L. Temporal segmentation of laparoscopic videos into surgical phases. In *Content-Based Multimedia Indexing (CBMI), 2016 14th International Workshop on* (2016), IEEE, pp. 1–6.

[84] Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., and Lee, H. Generative adversarial text to image synthesis. In *Proceedings of The 33rd International Conference on Machine Learning* (2016), vol. 3.

[85] Rezazadegan, F., Shirazi, S., and Davis, L. S. A real-time action prediction framework by encoding temporal evolution.

[86] Richard, A., and Gall, J. Temporal action detection using a statistical language model. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 3131–3140.

[87] Romera-Paredes, B., Aung, M. S., and Bianchi-Berthouze, N. A one-vs-one classifier ensemble with majority voting for activity recognition. In *ESANN 2013 proceedings, 21st European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning* (2013), pp. 443–448.

[88] Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention* (2015), Springer, pp. 234–241.

[89] Rudd, E. M., Günther, M., and Boult, T. E. Moon: A mixed objective optimization network for the recognition of facial attributes. In *European Conference on Computer Vision* (2016), Springer, pp. 19–35.

[90] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision 115*, 3 (2015), 211–252.

[91] Sainath, T. N., Vinyals, O., Senior, A., and Sak, H. Convolutional, long short-term memory, fully connected deep neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on* (2015), IEEE, pp. 4580–4584.

[92] Sak, H., Senior, A. W., and Beaufays, F. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Interspeech* (2014), pp. 338–342.

[93] Shahroudy, A., Ng, T.-T., Gong, Y., and Wang, G. Deep multimodal feature analysis for action recognition in rgb+ d videos. *arXiv preprint arXiv:1603.07120* (2016).

[94] Shotton, J., Sharp, T., Kipman, A., Fitzgibbon, A., Finocchio, M., Blake, A., Cook, M., and Moore, R. Real-time human pose recognition in parts from single depth images. *Communications of the ACM 56*, 1 (2013), 116–124.

[95] Simonyan, K., and Zisserman, A. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems* (2014), pp. 568–576.

[96] Simonyan, K., and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[97] Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research 15*, 1 (2014), 1929–1958.

[98] Srivastava, N., Mansimov, E., and Salakhudinov, R. Unsupervised learning of video representations using lstms. In *International Conference on Machine Learning* (2015), pp. 843–852.

[99] Stauder, R., Ostler, D., Kranzfelder, M., Koller, S., Feussner, H., and Navab, N. The tum lapchole dataset for the m2cai 2016 workflow challenge. *arXiv preprint arXiv:1610.09278* (2016).

[100] Stein, S., and McKenna, S. J. Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing* (2013), ACM, pp. 729–738.

[101] Stikic, M., Huynh, T., Van Laerhoven, K., and Schiele, B. Adl recognition based on the combination of rfid and accelerometer sensing. In *Pervasive Computing Technologies for Healthcare, 2008. PervasiveHealth 2008. Second International Conference on* (2008), IEEE, pp. 258–263.

[102] Stork, J. A., Spinello, L., Silva, J., and Arras, K. O. Audio-based human activity recognition using non-markovian ensemble voting. In *RO-MAN, 2012 IEEE* (2012), IEEE, pp. 509–514.

[103] Tian, Y., Ruan, Q., An, G., and Fu, Y. Action recognition using local consistent group sparse coding with spatio-temporal structure. In *Proceedings of the 2016 ACM on Multimedia Conference* (2016), ACM, pp. 317–321.

[104] Tran, A., and Cheong, L.-F. Two-stream flow-guided convolutional attention networks for action recognition. *arXiv preprint arXiv:1708.09268* (2017).

[105] Twinanda, A. P., Shehata, S., Mutter, D., Marescaux, J., de Mathelin, M., and Padoy, N. Endonet: A deep architecture for recognition tasks on laparoscopic videos. *IEEE Transactions on Medical Imaging 36*, 1 (2017), 86–97.

[106] Wang, F., Jiang, M., Qian, C., Yang, S., Li, C., Zhang, H., Wang, X., and Tang, X. Residual attention network for image classification. *arXiv preprint arXiv:1704.06904* (2017).

[107] Wang, H., and Schmid, C. Action recognition with improved trajectories. In *Proceedings of the IEEE International Conference on Computer Vision* (2013), pp. 3551–3558.

[108] Wang, J., Cheng, Y., and Schmidt Feris, R. Walk and learn: Facial attribute representation learning from egocentric video and contextual data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 2295–2304.

[109] Ward, J. A., Lukowicz, P., and Gellersen, H. W. Performance metrics for activity recognition. *ACM Transactions on Intelligent Systems and Technology (TIST) 2*, 1 (2011), 6.

[110] Wu, J., Osuntogun, A., Choudhury, T., Philipose, M., and Rehg, J. M. A scalable approach to activity recognition based on object use. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on* (2007), IEEE, pp. 1–8.

[111] Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* (2016).

[112] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., and Bengio, Y. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning* (2015), pp. 2048–2057.

[113] Yan, S., Smith, J. S., Lu, W., and Zhang, B. Cham: action recognition using convolutional hierarchical attention model. *arXiv preprint arXiv:1705.03146* (2017).

[114] Yang, L., Chen, Y., Li, X.-Y., Xiao, C., Li, M., and Liu, Y. Tagoram: Real-time tracking of mobile rfid tags to high precision using cots devices. In *Proceedings of the 20th annual international conference on Mobile computing and networking* (2014), ACM, pp. 237–248.

[115] Yosinski, J., Clune, J., Nguyen, A., Fuchs, T., and Lipson, H. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579* (2015).

[116] Yuan, B., and Herbert, J. A cloud-based mobile data analytics framework: Case study of activity recognition using smartphone. In *Mobile Cloud Computing, Services, and Engineering (MobileCloud), 2014 2nd IEEE International Conference on* (2014), IEEE, pp. 220–227.

[117] Zeiler, M. D., and Fergus, R. Visualizing and understanding convolutional networks. In *European conference on computer vision* (2014), Springer, pp. 818–833.

[118] Zhang, C., and Tian, Y. Rgb-d camera-based daily living activity recognition. *Journal of Computer Vision and Image Processing 2*, 4 (2012), 12.

[119] Zhang, M., and Sawchuk, A. A. Motion primitive-based human activity recognition using a bag-of-features approach. In *Proceedings of the 2nd ACM SIGHIT International Health Informatics Symposium* (2012), ACM, pp. 631–640.

[120] Zhang, S., Rowlands, A. V., Murray, P., Hurst, T. L., et al. *Physical activity classification using the GENEA wrist-worn accelerometer*. PhD thesis, Lippincott Williams and Wilkins, 2012.

[121] Zhu, Y., Groth, O., Bernstein, M., and Fei-Fei, L. Visual7w: Grounded question answering in images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 4995–5004.