**Abstract**

Human action recognition is still attracting the computer vision research community due to its various applications. Provided dataset contains recorded videos and annotations from two different hospitals namely HCC(Howard Community College) & CMC which are renowned for providing students with innovative, top-quality, and affordable learning opportunities to enter and succeed in the nursing field. One of the key focus areas of both the hospitals is to provide nursing education and training programs which involves conducting and evaluating future nurses under guidance and supervision of highly skilled instructors. Dataset from HCC and CMC comprises of videos shot from different static cameras installed in a simulation environment where health care providers are seen applying their learnings on manikins.

With this data a timeline over the activities such as washing hands is needed to be extracted. However, there are issues like improper annotation, video quality, occlusion, non-standard cameras, multiple actors in long untrimmed video etc. which require further pre-processing, data curation and generation of labelled data to produce a reliable source.

The aim of this thesis is to lay down a strategy for event detection in untrimmed videos using advanced techniques and methods from the field of deep neural networks by primarily focusing on detecting and localizing a particular action i.e., washing hands in long term continuous sequence of frames. Earlier work has used object detection networks to locate and classify objects in the videos and post-process the videos to a format that the activity recognition network could predict on. Historically many related works have shown some improvement in model accuracies when optical flow and object detection is used as a preliminary step. However, in this work, we decided to only concentrate on extracting, processing, and using frame data to establish a baseline and implement other architectures for comparative study.

Table of Contents

1. **Introduction**
   1. Motivation
   2. Related Work
   3. Objectives & Contributions
   4. Thesis Outline
2. **Background**
   1. Need of human activity recognition
   2. Activities

1. **Technical Background**
   1. Artificial Neural Networks
      1. Convolution Neural Network (CNN)
   2. Architectures
      1. C3D
      2. I3D
   3. Performance Metrics
2. **Data Material & Curation**
   1. Expain about input dataset
   2. Classes of Interest
   3. About Dataset, Explain need
   4. Extract data for AWS
   5. Data Curation Tool
   6. Generate test and training data
3. **Proposed Methods**

5.1. Related work & projects

* 1. Pipeline
  2. Pre-processing
  3. Proposed model for Activity recognition
  4. Train and Test strategy

1. **Experiments & Results**
   1. Classification
   2. Localization
   3. Other experiments
2. **Discussion**
   1. Discuss the results
   2. Proposed Improvements
3. **Conclusion**
4. **Bibliography**

Chapter 1

Introduction

People are arguably most important objects in images and video. Indeed, about 35% of pixels in movies and YouTube videos as well as about 25% of pixels in photographs belong to people [Laptev, 2013]. Not surprisingly, human-centric tasks usually serve as core components in many computer vision pipelines. Analyzing people in images and videos has many practical applications in security, entertainment, education and other domains. Airport security can benefit from face recognition to prevent attacks and violent behavior. Video surveillance systems require reliable detection of malicious human activities like robbery, burglary or violence. Aside from security purposes, person detection/tracking and action analysis can be used to assist sport coaches in planning strategies. Human gaze detectors in smart TVs can capture human attention and help generating targeted advertisements. Analyzing student behaviors and interactions gives more insight on how students teach each other which helps to optimize peer instruction [Jermann et al., 2011, Chong et al., 2017, Hayashi, 2018]. At a larger scale, crowd counting and crowd motion analysis are useful in scenarios such as demonstrations or mass evacuations.

Person Detection is another human-centric computer vision task aiming to localize people in images and video. It often serves as a backbone for many other human analysis tasks, e.g. human verification, action recognition, behavior understanding, crowd counting and others. Given the needs of time-critical applications, the performance of person detectors is important both in terms of speed and accuracy [Viola and Jones, 2001, Dalal and Triggs, 2005, Girshick, 2015]. Similar to action prediction, the task can be addressed in the context of still images [Girshick, 2015, Vu et al., 2015, Ren et al., 2015] and videos [Zhu et al., 2017b].

This thesis addresses the demand of classification and action localization in visual data collected in form of long untrimmed videos. In the scope of these two problems, the study also touches various deep learning concepts ranging from data capturing, data understanding, data pre-processing in the spatio-temporal domain. Below we identify the goals of the thesis in Section 1.1 and outline main contributions of the thesis in Section 1.2.

* 1. Goals

Our first goal is to perform action classification given a set of long untrimmed video from two different hospitals. The goal of action classification is to determine which action appears in a particular video.

Second goal is to perform action localization where the objective is to detect when and where an action of interest occurs. The expected output of such an action localization system is typically a sub-volume surrounding the action of interest. Since a localized action only covers a fraction of the spatio-temporal volume in a video, the task is considerably more challenging than action classification and temporal detection.

* 1. Related Work

In this chapter we review previous works that is closely related to this thesis. The chapter is comprised of two sections: Section 2.1 discusses some works on action recognition and action localization; Section 2.2 gives a brief overview of person and object detection methods.

**1.2.1 Action Recognition**

There had been a lot of research and development in the field of human action recognition in videos. Early methods mainly involved human body parts tracking [Rohr, 1994] and human motion analysis [Bobick and Davis, 2001, Efros et al., 2003]. Follow up methods focused on statistical representations for action recognition. Laptev [2005] represented motion patterns with space-time local features. The idea is to localize spatio-temporal interest points corresponding to characteristic events. Using such interest points, Bagof-Words approach has been used to represent actions in the video. Schuldt et al.[2004], Laptev et al. [2008] classified actions by applying Support Vector Machines (SVMs) on the occurrence histograms. Wang et al. [2011] proposed an action recognition framework with dense trajectory descriptors. Feature points are first localized and then tracked with optical flow to densely produce point trajectories. Each trajectory is represented by descriptors, e.g. HOG, HOF and MOH, within its neighborhood space-time volume. Action recognition is performed with the standard bag-of-features approach.

Deep convolutional neural networks have been applied for action recognition. Simonyan and Zisserman [2014] designed a two-stream architecture separately processing RGB images and optical flows. Late fusion is applied on the L2-normalized softmax outputs of the two streams. The network achieved comparable performance with state-of-the-art methods using “hand-crafted" features. Despite relatively small improvements, this work showed promising potential of CNNs for action recognition. More recently, Tran et al. [2015] introduces C3D, a 3D convolutional neural networks for action recognition. C3D architecture extends 2D CNNs to videos. The learned C3D features computed from RGB input have been used for video representation, followed by SVM for action classification. Varol et al. [2016] extended C3D to learn long-term video representation and confirmed the advantage of using optical flows for human action recognition. Like how CNN models for recognition tasks on images benefit from the pretraining phase on the ImageNet dataset, CNN models for videos considerably benefit in pretraining on big datasets such as Sport-1M [Karpathy et al., 2014] and Kinetics [Kay et al., 2017]. The “Two-Stream Inflated 3D ConvNets” (I3D) extends state-of-the-art architectures on image classification to handle spatiotemporal 3D information in videos. I3D models pretrained on the Kinetics dataset and finetuned on HMDB-51 [Kuehne et al., 2011] and UCF-101 [Soomro et al., 2012] datasets achieve state-of-the-art performance on the both action recognition benchmarks. This thesis contains experiments with both C3D and I3D networks which is explained the following chapters. It has been seen that training CNNs for videos is a challenging task due to the difficulty to collect data annotation and high memory consumption of the deep networks.

Action recognition in stills images received less attention compared to videos. The work of Ikizler et al. [2008] was one of the first attempts to recognize actions in static images using human poses. The authors argued that poses often characterize actions, so one can extract and classify poses to derive action labels of images.

**1.2.2 Action Localization**

Many researchers have concentrated on temporal action localization in long untrimmed videos.

Jain et al. [18] introduced a sampling strategy to produce tubelets with motion information from super-voxels 1520-9210 (c) 2018 IEEE. Ma et al. [25] described hierarchical space-time segments as new representations for action recognition and localization. Heilbron et al. [5] introduced a sparse learning dictionary method to recover the temporal segments containing interested actions. These methods utilize hand-crafted features with encoding methods to determine the temporal boundaries of action instances. As deep convolutional neural networks (CNN) [10] have demonstrated breakthrough performance for image feature extraction, more and more studies of temporal action localization focus on deep learning. Shou et al. [32] utilized 3D ConvNets [35] to design a multi-stage framework for temporal action localization, which explicitly took the temporal overlap into account. They also presented convolutional networks to predict actions at the frame level granularity later [31]. Yeung et al. [45] formulated the localization model as a recurrent neural network (RNN) based agent and used reinforcement learning [42] to learn the agent’s decision policy. The fully end-to-end network takes a long video as input and outputs the temporal bounds of all action instances. Dai et al. [8] presented a temporal context network to produce action proposals, rank and classify the proposals for temporal localization of human activities. Zhao et al. [52] modeled each action instance with a temporal pyramid, and all the context information of action instances are leveraged sufficiently. Hou et al. [17] proposed a tube convolutional neural network to localize actions based on 3D convolution features. Zhu et al. [55] also combined the 3D ConvNets with multitask learning. In contrast with these complicated networks, we utilize deep networks to both learn the spatio-temporal information and the high-level semantic features to effectively recognize segments in videos. More importantly, we introduce action pattern trees to model the temporal relationship between segments and infer precise temporal boundaries of action instances.

* 1. Thesis Outline

**Chapter 2: Background**

This chapter describes about different types of activities seen in the dataset. Need for detecting human activities, source of dataset and annotations.

**Chapter 3: Technical background**

This chapter describes all technical theory behind the methods implemented.

**Chapter 4: Data Curation**

This chapter explains the proposed approach adapted for creation of dataset.

**Chapter 5: Data Material**

A chapter introducing where diﬀerent data is coming from, what they contain and names

for datasets used in this thesis.

**Chapter 6: Methods**

This chapter explains how the methods and shows how they are implemented to be used in

experiments.

**Chapter 7: Experiments**

An introduction to each experiment conducted, what settings were used and which dataset

were used.

**Chapter 8: Results**

The results from the experiments introduced in the previous chapter.

**Chapter 9: Improvements & Future work**

This chapter discusses the results from the previous chapter and explain.

**Chapter 10: Conclusion**

The conclusion of the thesis is presented here and further work.

Chapter 2

Background

* 1. **Need for detecting human actions/Health care providers.**

Major area of expertise for HCC and CMC is to facilitate supervision and training of health care workers. To ascertain whether a particular type of activity was performed during a session, observers/instructors sitting in a different room need to carefully watch a camera feeds from a session. To depict an overall view, cameras are installed in such a way that all actions performed by nurses can be easily seen. In many cases, observer is seen sitting next to the patient in the patient’s room with or without a device where he/she can annotate different actions. Watching and qualifying every session by observing long videos for particular type of activity can take a lot of time and human intervention, so having a method which can automatically do the classification and can localize the activity in a video clip is beneficial. This thesis mainly focused on a particular type of activity called – “Washing hands” i.e., we need to devise a method which can classify whether or not a “washing hands” activity was performed in the video clip.

**2.2. Types of activities**

Following are some of the most common activities which can be seen in the video dataset after filtering the videos of monitoring screen and observers.

* Washing hands using sanitizer bottle/dispenser/water
* Wearing Gloves
* Standing, Talking
* Taking notes

|  |  |  |
| --- | --- | --- |
| **A picture containing indoor, wall, person  Description automatically generated** | **A picture containing indoor, person, bed, room  Description automatically generated** | **A person lying in a bed  Description automatically generated with low confidence** |
|  |  |  |
| **A picture containing indoor, wall, worktable  Description automatically generated** | **A person standing in a room  Description automatically generated with low confidence** | **A picture containing indoor, hospital room, scene, room  Description automatically generated** |
| **Fig. 1 Typical activity examples taken from the dataset** | | |
|  | | |

Chapter 3

Technical Background

**3.1 Artificial Neural Networks**

Artificial neural networks (ANNs), usually simply called neural networks (NNs), are systems which are vaguely inspired by the biological neural networks.

An ANN is based on a collection of connected units or nodes called artificial neurons which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal to other neurons. An artificial neuron that receives a signal then processes it and can signal neurons connected to it. The "signal" at a connection is a real number and the output of each neuron is computed by some non-linear function of the sum of its inputs. The connections are called *edges*. Neurons and edges typically have a weight that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Neurons may have a threshold such that a signal is sent only if the aggregate signal crosses that threshold. Typically, neurons are aggregated into layers. Different layers may perform different transformations on their inputs. Signals travel from the first layer (the input layer), to the last layer (the output layer), possibly after traversing the layers multiple times.

Neural networks learn (or are trained) by processing examples, each of which contains a known "input" and "result," forming probability-weighted associations between the two, which are stored within the data structure of the net itself. The training of a neural network from a given example is usually conducted by determining the difference between the processed output of the network (often a prediction) and a target output. This is the error. The network then adjusts its weighted associations according to a learning rule and using this error value. Successive adjustments will cause the neural network to produce output which is increasingly similar to the target output. After a sufficient number of these adjustments the training can be terminated based upon certain criteria which is called Supervised learning.

Such systems "learn" to perform tasks by considering examples, generally without being programmed with task-specific rules. For example, in [image recognition](https://en.wikipedia.org/wiki/Image_recognition), they might learn to identify images that contain cats by analyzing example images that have been manually labelled as "cat" or "no cat" and using the results to identify cats in other images. They do this without any prior knowledge of cats, for example, that they have fur, tails, whiskers, and cat-like faces. Instead, they automatically generate identifying characteristics from the examples that they process.

### Components of ANNs

#### **Neurons**

ANNs are composed of artificial neurons which are conceptually derived from biological neurons. Each artificial neuron has inputs and produces a single output which can be sent to multiple other neurons. The inputs can be the feature values of a sample of external data, such as images or documents, or they can be the outputs of other neurons. The outputs of the final *output neurons* of the neural net accomplish the task, such as recognizing an object in an image.

To find the output of the neuron, first we take the weighted sum of all the inputs, weighted by the *weights* of the *connections* from the inputs to the neuron. We add a *bias* term to this sum. This weighted sum is sometimes called the *activation*. This weighted sum is then passed through a (usually nonlinear) activation function to produce the output. The initial inputs are external data, such as images and documents. The ultimate outputs accomplish the task, such as recognizing an object in an image.

#### **Connections and weights**

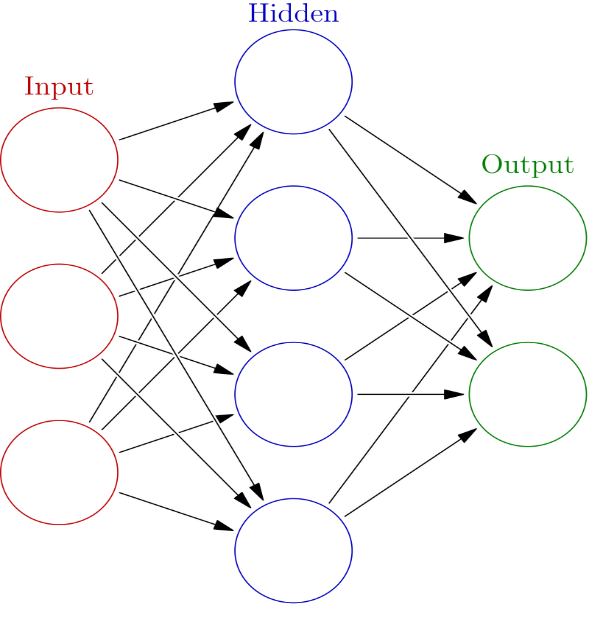
The network consists of connections, each connection providing the output of one neuron as an input to another neuron. Each connection is assigned a weight that represents its relative importance. A given neuron can have multiple input and output connections.

#### **Propagation function**

The *propagation function* computes the input to a neuron from the outputs of its predecessor neurons and their connections as a weighted sum. A *bias* term can be added to the result of the propagation.

### Organization

The neurons are typically organized into multiple layers, especially in deep learning. Neurons of one layer connect only to neurons of the immediately preceding and immediately following layers. The layer that receives external data is the *input layer*. The layer that produces the ultimate result is the *output layer*. In between them are zero or more *hidden layers*. Single layer and unlayered networks are also used. Between two layers, multiple connection patterns are possible. They can be *fully connected*, with every neuron in one layer connecting to every neuron in the next layer. They can be *pooling*, where a group of neurons in one layer connect to a single neuron in the next layer, thereby reducing the number of neurons in that layer. Neurons with only such connections form a directed acyclic graph and are known as feed forward networks. Alternatively, networks that allow connections between neurons in the same or previous layers are known as recurrent networks.



### Hyperparameter

A hyperparameter is a constant parameter whose value is set before the learning process begins. The values of parameters are derived via learning. Examples of hyperparameters include learning rate, the number of hidden layers and batch size. The values of some hyperparameters can be dependent on those of other hyperparameters. For example, the size of some layers can depend on the overall number of layers.

### Learning

Learning is the adaptation of the network to better handle a task by considering sample observations. Learning involves adjusting the weights (and optional thresholds) of the network to improve the accuracy of the result. This is done by minimizing the observed errors. Learning is complete when examining additional observations does not usefully reduce the error rate. Even after learning, the error rate typically does not reach 0. If after learning, the error rate is too high, the network typically must be redesigned. Practically this is done by defining a cost function that is evaluated periodically during learning. As long as its output continues to decline, learning continues. The cost is frequently defined as a statistic whose value can only be approximated. The outputs are actual numbers, so when the error is low, the difference between the output (almost certainly a cat) and the correct answer (cat) is small. Learning attempts to reduce the total of the differences across the observations. Most learning models can be viewed as a straightforward application of optimization theory and statistical estimation.

#### Learning rate

The learning rate defines the size of the corrective steps that the model takes to adjust for errors in each observation. A high learning rate shortens the training time, but with lower ultimate accuracy, while a lower learning rate takes longer, but with the potential for greater accuracy. Optimizations such as Quickprop are primarily aimed at speeding up error minimization, while other improvements mainly try to increase reliability. In order to avoid oscillation inside the network such as alternating connection weights, and to improve the rate of convergence, refinements use an adaptive learning rate that increases or decreases as appropriate.[[47]](https://en.wikipedia.org/wiki/Artificial_neural_network#cite_note-47) The concept of momentum allows the balance between the gradient and the previous change to be weighted such that the weight adjustment depends to some degree on the previous change. A momentum close to 0 emphasizes the gradient, while a value close to 1 emphasizes the last change.

#### Cost function

While it is possible to define a cost function in an adhoc manner, frequently the choice is determined by the function's desirable properties (such as convexity) or because it arises from the model (e.g. in a probabilistic model the model's posterior probability can be used as an inverse cost).

#### Backpropagation

Backpropagation is a method used to adjust the connection weights to compensate for each error found during learning. The error amount is effectively divided among the connections. Technically, backprop calculates the gradient (the derivative) of the cost function associated with a given state with respect to the weights. The weight updates can be done via stochastic gradient descent or other methods, such as Extreme Learning Machines, "No-prop" networks training without backtracking, weightless networks, and non-connectionist neural networks.

### Learning paradigm

#### Supervised learning

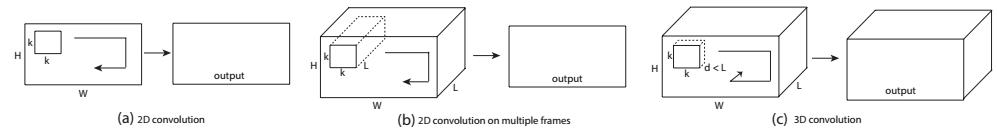
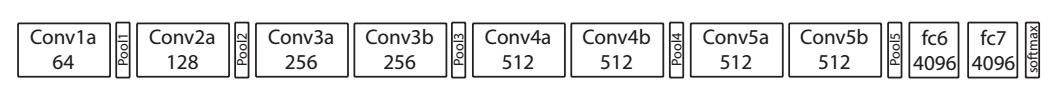
Supervised learning uses a set of paired inputs and desired outputs. The learning task is to produce the desired output for each input. In this case the cost function is related to eliminating incorrect deductions. A commonly used cost is the mean-squared error, which tries to minimize the average squared error between the network's output and the desired output. Tasks suited for supervised learning are pattern recognition (also known as classification) and regression (also known as function approximation). Supervised learning is also applicable to sequential data (e.g., for handwriting, speech and gesture recognition). This can be thought of as learning with a "teacher", in the form of a function that provides continuous feedback on the quality of solutions obtained thus far.

#### Unsupervised learning

Unsupervised learning methods do not need so many labels of data, so this reduces some requirements of the collected data. Unsupervised learning relies on its unique training system, and classifies original data with little or no already known label data. So unsupervised learning could find some potential quality that we humans could not such as when researchers observe and get some galaxies that have never been seen before, and they can’t archive these galaxies because there is no knowledge of them.

Unsupervised learning algorithms are used to group cases based on similar attributes, or naturally occurring trends, patterns, or relationships in the data. These models also are referred to as self-organizing maps. Unsupervised models include [clustering techniques](https://www.sciencedirect.com/topics/computer-science/clustering-technique) and self-organizing maps. Different algorithms use different strategies for dividing data into groups. Some methods are relatively straightforward, quickly dividing the cases into groups based on common attributes or some other similarity. The two-step [clustering method](https://www.sciencedirect.com/topics/computer-science/clustering-method) differs somewhat in that an optimal number of clusters is determined in an initial pass through the data, based on certain statistical criteria. Group assignment is then made on a second pass through the data; hence the name “two-step.” [Neural networks](https://www.sciencedirect.com/topics/social-sciences/neural-network) are more complicated than some of the other unsupervised learning algorithms and can yield results that are relatively opaque and difficult to interpret. An example of an unsupervised learning algorithm includes the analysis of the conference call data

**C3D**

  
2D and 3D convolution operations. a) Apply two-dimensional convolution to the image to obtain the image. b) Applying two-dimensional convolution to a video volume (multiple frames as multiple channels) will also result in an image. c) Perform three-dimensional convolution on one video volume to obtain another volume, and save the time information of the input signal.  
  
.C3D architecture. C3D net has 8 convolutions, 5 max pools, 2 fully connected layers, and then two fully connected layers. All three-dimensional convolution kernels are 3×3×3 and stride 1 in space and time dimensions. Each box represents the number of filters. The 3D pool layer is represented as pool1 to pool5. All pool cores are 2×2×2, except for pool1 which is 1×2×2. Each fully connected layer has 4096 output units.

After five times of convolution and four times of pooling, the size of the feature map in Conv5 will be reduced to 16×16 times of the original image, and the time dimension will be reduced to 8 times of the original video.