Universitetet
i Stavanger

FACULTY OF SCIENCE AND TECHNOLOGY

MASTER'S THESIS

Study programme/specialisation: Computer Science - Master's Degree Specialization : Data Science	Spring semester, 2021 Open/~~Confidential~~
Author: Anil Kumar Dhiman	[illegible] (signature of author)
Supervisor(s): Øyvind Meinich-Bache Professor Kjersti Engan	
Title of master's thesis: Automatic Event Detection in videos using Deep Neural Networks	
Credits: 30	
Keywords: Deep Learning, Convolutional Neural Network, Data Curation	Number of pages: 66 + supplemental material/other Stavanger, 15th of June 2021.

University of Stavanger

Automatic Event Detection in Videos using Deep Neural Networks

MASTER'S THESIS

Anil Kumar Dhiman

June, 2021

Supervisors

Øyvind Meinich-Bache

Professor Kjersti Engan

Faculty of Science and Technology
Department of Electrical Engineering and Computer Science
University of Stavanger

Abstract

Within a large range of applications in computer vision, Human Action Recognition has become one of the most attractive research fields.This thesis investigates possibilities of applying automatic event detection on large dataset of simulation videos captured during medical training sessions. In a typical training session different scenario-based event can occur and the students undergoing the training must take actions accordingly. These events and actions are manually annotated by an observer using an app or by watching the video after a session. Such hand-crafted annotations are later used for evaluating the recorded sessions which requires human intervention and can quickly become tedious, time consuming and difficult task (especially when there are a lot of things going on in a particular training or simulation setting). Hence, this thesis aims to solve the challenges by :

- Providing a baseline approach for automatically detecting events occurring in long untrimmed videos
- Activity localization

This thesis is focused mainly on detecting **Washing Hands** activity performed by health care providers and medical students under different settings. The proposed system approach consists of activity recognition and generation of activity timelines using 3D CNNs.

The dataset used in this thesis originally contained more than 4000 untrimmed videos with associated annotations, of which only 60% of the data was found to be relevant but required reliable annotations before it could be fed into the deep neural network. Hence, as an initial step into this thesis a reusable Data Curation tool was developed and used extensively for generation of ground truth annotations.

This thesis proposes a generalized methods for data curation and activity recognition. An overall classification accuracy of 68% was achieved in this work using the proposed method.

Acknowledgements

This master thesis has been conducted at the Department of Electrical Engineering and Computer Science at the University of Stavanger, during the spring semester of 2021. I would like to give my sincere thanks to my supervisors, Professor Kjersti Engan and Øyvind Meinich-Bache for always being available for discussion, constructive feedback and suggestions on how to improve the work. I would like to thank Laerdal Medical for giving me an opportunity to participate in this work and providing me with the use-case and the dataset. I would like to thank Rune Wetteland Theodor Ivesdal for assisting me time to time with GPU access and unix server requirements. I would also like to direct gratitude to my family for supporting and believing in me throughout this period.

Contents

Abstract i

Acknowledgements ii

List of Figures vi

List of Tables viii

Abbreviations ix

1 Introduction 1
- 1.1 Motivation . . . 2
- 1.2 Objectives & Contributions . . . 2
  - 1.2.1 Contributions . . . 3
- 1.3 Thesis outline . . . 5

2 Theory and Background 6
- 2.1 Deep Learning . . . 6
  - 2.1.1 Historical Background . . . 6
  - 2.1.2 Dataset splits . . . 8
  - 2.1.3 Neural Networks . . . 9
    - 2.1.3.1 Loss function . . . 13
    - 2.1.3.2 Optimization . . . 14
    - 2.1.3.3 Training Process . . . 14
    - 2.1.3.4 Data Preprocessing . . . 15
    - 2.1.3.5 Regularization . . . 16
  - 2.1.4 Convolutional Neural Networks . . . 17
    - 2.1.4.1 Convolutional Layers . . . 17
    - 2.1.4.2 Pooling . . . 18
  - 2.1.5 Recurrent Neural Networks . . . 19
    - 2.1.5.1 Long Short Term Memory(LSTM) . . . 20
  - 2.1.6 3D-Convolutions . . . 21
- 2.2 Transfer Learning . . . 21
- 2.3 Performance Metrics . . . 22
  - 2.3.1 Confusion Matrix . . . 22
  - 2.3.2 Binary Classification Measures . . . 23

2.4 Human activity recognition 25
2.5 Related work 26
2.5.1 Action Recognition 26
2.5.2 C3D Network 27
2.5.3 Inception 3D Network 28
2.5.4 Action Localization 30
2.6 Video Classification Overview 31
2.6.1 Data Curation 32
2.6.2 Video classification architecture 32
2.7 Deep Learning Development Platforms 32
2.7.1 TensorFlow™ 33
2.7.2 Keras 34

3 Data Material **35**
3.1 Dataset Overview 35
3.2 Dataset Challenges 37
3.3 Data Source & Extraction 38
3.4 Dataset splits 39

4 Proposed Method - Data Curation **41**
4.1 Data curation pipeline 41
4.1.1 Data discovery 42
4.1.2 Entity resolution 42
4.1.3 Data cleaning 44

5 Proposed Method - Activity Recognition **45**
5.1 Method Overview 45
5.1.1 Training Setup 46
5.1.1.1 Data preparation 46
5.1.1.2 Spatial stream 47
5.1.1.3 Final Classification Layer 47
5.1.1.4 Hyper-Parameter Tuning 48
5.1.2 Testing Setup 48
5.1.2.1 Data Preparation 48
5.1.2.2 Model Evaluation 49
5.2 Implementation Details 49
5.2.1 Computational platform 49
5.2.2 Source References 50
5.3 Network Models 50
5.3.1 I3D Network Implementation 50
5.3.2 C3D Network Implementation 51

6 Experiments & Results **53**
6.1 Preliminary Experiments 53
6.2 Experiment 1 54
6.3 Experiment 2 54
6.4 Experiment 3 55
6.5 Experiment 4 56

6.6 A closer look at results 57

7 Discussions **63**
7.1 Dataset Challenges 63
7.2 Network configuration & Pipeline 64
7.3 Results 65

8 Conclusion **67**
8.1 Future Work 68

Bibliography **69**

List of Figures

1.1 Examples from the dataset 4

2.1 The figure shows a fully connected Neural Network with an input layer, one hidden layers and an output layer. This network is of the typical feed forward architecture, where all connections go forward through the network. Each of the connections between the neurons also have a weight W 10
2.2 Neural Network formulation 10
2.3 Sigmoid activation function 11
2.4 Tanh activation function 12
2.5 ReLU activation function 13
2.6 A figure showing an overview of the Convolution Operation. The activation map is computed by sliding(or convolving) the filter F over the input image I and computing the dot product between the filter and its current location on the input image. Figure adapted from [1]. 17
2.7 An illustration showing a 2×2 MaxPooling with stride 2. Each max is taken over a 2×2 square. The filter is then moved two squares for the next computation. Figure adapted from [2]. 18
2.8 Fundamentals of Recurrent Neural Network(RNN) 19
2.9 Long short term memory network 20
2.10 Confusion Matrix 23
2.11 C3D Architecture[3] 28
2.12 Inflated 3D (I3D) network architecture [4] 29
2.13 Inception module network architecture [4] 30
2.14 Basic steps in video classification 33

3.1 Initial data allocation under different categories. 35
3.2 Shows different types of video clips in the dataset. Starting at top-left - No activity, others(camera angle), observers, patient monitoring screen respectively 36
3.3 Trimmed video frames showing washing hands activity. 37
3.4 Shows different simulation/training environments 39

4.1 Data curation pipeline 41
4.2 Graphical User Interface - Data Curation Tool 42
4.3 Flowchart for Data Curation Tool 43

5.1 Pipeline for proposed method 45
5.2 A visualization of training and testing pipeline 46

5.3 Learning rate 51
5.4 Model Loss 51
5.5 Fig representing learning rate and model loss plots during I3D network training 51
5.6 C3D model training 52

6.1 Shows training and validation accuracies which suffers from overfitting due to class imbalance 55
6.2 Shows training and validation accuracies with C3D approach 57
6.3 Visualization of ROC curve using I3D approach 58
6.4 Visualization of Confusion Matrix 58
6.5 Shows an instance where model wrongly classifies an action as washing hands 59
6.6 Shows an instance where model correctly classifies it as a normal event 59
6.7 Shows that model correctly classifies the video clips as positive class i.e. Washing hands 60
6.8 Shows that model correctly detects washing hands activity 60
6.9 Shows that model correctly detects washing hands activity 61
6.10 Shows Activity Localization on different videos and compares ground-truth with C3D and I3D models. 62

8.1 Frame 1 69
8.2 Frame 2 70

List of Tables

3.1 Shows break-up of final dataset 40

5.1 Annotation format used for I3D 47
5.2 Annotation format used for C3D 47
5.3 List of common hyper-parameters for Activity detection network . . . 48
5.4 Annotation format used for Testing set 49

6.1 Table showing results of preliminary approaches 54
6.2 Shows hyper-parameters and model accuracies 55
6.3 Shows hyper-parameters and model accuracies 56
6.4 Shows hyper-parameters used and model accuracy 56

Abbreviations

HCP	**H**ealth **C**are **P**rovider
HCC	**H**oward **C**ommunity **C**ollege
DNN	**D**eep **N**eural **N**etworks
ANN	**A**rtificial **N**eural **N**etwork
PMS	**P**atient **M**onitoring **S**ystem
RNN	**R**eccurrent **N**eural **N**etworks
LSTM	**L**ong **S**hort **T**erm **M**emory
C3D	**C**onvolutional 3D
I3D	**I**nception 3D

Chapter 1

Introduction

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.

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 as depicted in [5]. Similar to action prediction, the task can be addressed in the context of still images and videos [6].

This thesis addresses the demand of human action classification and action localization in visual data collected in long untrimmed videos. In the scope of these two problems, this study also touches various deep learning concepts ranging from data capturing, data understanding, data pre-processing in the spatio-temporal domain.

This chapter presents the motivation behind this master's thesis. The findings and conclusions from the the field of human activity recognition are used as guidelines and motivation for the hypotheses we present in this thesis. Finally, we give a brief overview of the thesis structure.

1.1 Motivation

Due to manifold increase in the growth of computational power, scientists and researchers have looked at Deep Neural Network and its application in a much greater depth. This improvement and in-depth research has triggered the need for automation and decision support systems over recent years. Health care is one such sector where adaptation of innovative technologies to reduce the time and cost required for solving a problem is of utmost importance.One avenue being explored lies in the technological advancements that can make hospital working environments much more efficient. Automating certain processes can save time and cost. In this thesis, we aim to deduce a baseline approach that can ascertain whether or not a particular type of activity has been performed during a training or evaluation session, which otherwise is done manually by observers and health care professionals.

This Master thesis focuses on detecting and localizing **"Washing hands"** activity by applying known practices for human activity recognition using Deep Neural network architectures.

1.2 Objectives & Contributions

Two main objectives of this thesis are Activity classification and Activity localization. As discussed in the previous sections, both areas have been explored significantly with the advent of deep neural networks. Hence the preliminary idea for this work is to start with existing state of the art neural network models and baseline our video understanding. However, observing the quality of data and associated annotations posed a a set of initial challenges which are listed below :

- We collected around 4800 video clips of varying length, of which 40% clips needed to be discarded due to bad quality, camera angle, audio only etc. Filtering these videos nearly halved the number of video clips available for training and testing.

- After filtering the noisy videos, it was seen that more than 15% of the provided annotations were either off by few seconds or there were instances where no activity could be seen yet annotation were made available to us. Hence, the ground truth labels were found to be not completely reliable. This triggered a need for developing and using a tool which can easily produce ground truth labels for the filtered videos.

1.2.1 Contributions

In order to overcome the initial challenges with the dataset, a Data Curation Tool (Chapter 4) was developed and extensively used to prepare correct labels. Data Curation tool is a multipurpose and easy to use graphical user interface which enables users to rapidly produce annotations for any video dataset. Following is the brief summary of self contributions made in this thesis:

- Developed a utility to download and extract relevant video clips by using ground truth annotations. Due to unreliability of the annotations a wider window of $\pm 5\,sec$ was initially used to extract the desired but wider portion of video which significantly increases the probability of finding 'Washing hands' activity in the clip.Portion of the video which did not contain the desired activity were classified as Not Washing hands. See Section 3.3 for more details.

- Developed and used Data Curation tool to produce shorter video clips with labels.Chapter 4)

- Prepared dataset splits and class labels for various methods as explained in Chapter 3

Fig 1.1 shows instances of 6 different video clips in which health care providers can be seen performing different type of activities. Given the different settings, our main goal is to identify and recognize the washing hands activity in a video clip.

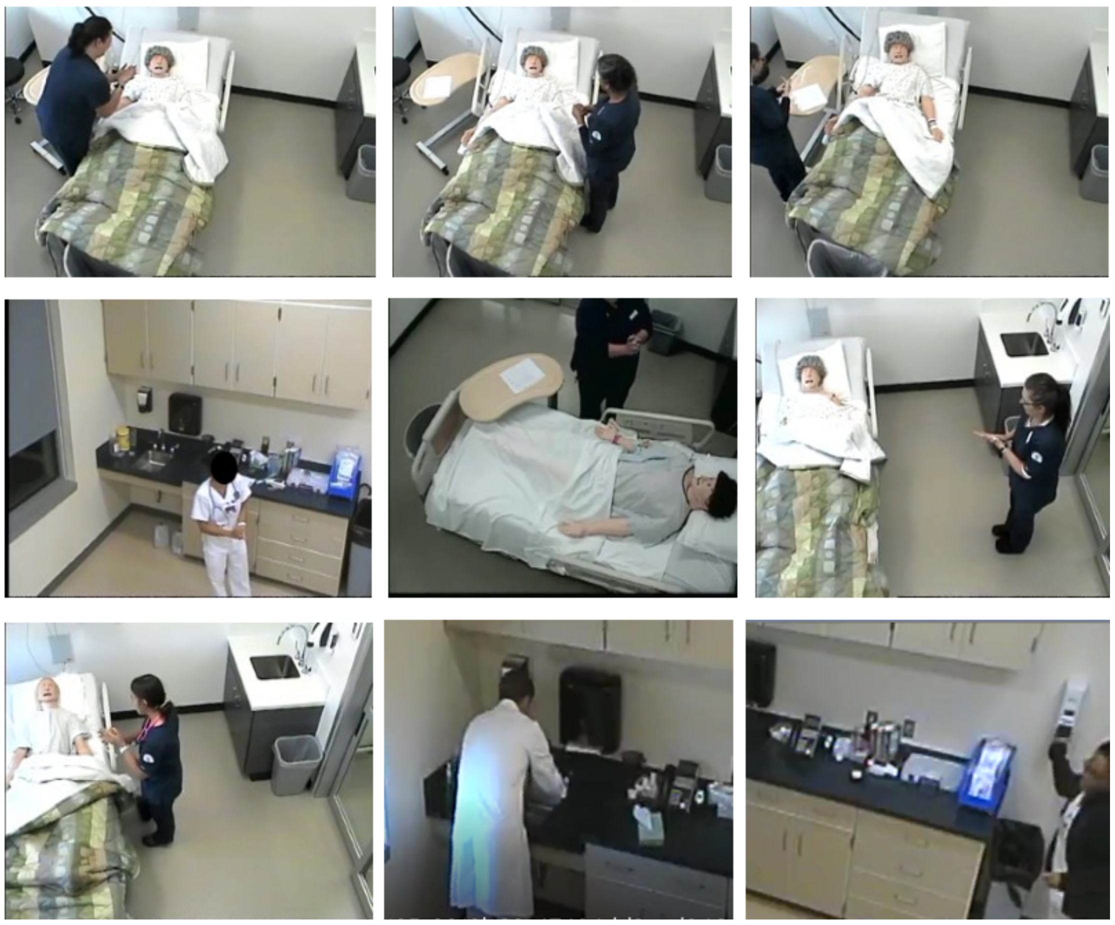


Figure 1.1: Examples from the dataset

1.3 Thesis outline

Chapter 2: Theory & Background

This chapter provides conceptual understanding on Deep Neural Networks. It also underpins various related works in the field of human activity recognition.

Chapter 3: Data Material

This chapter clearly explains about type and quantity of dataset. Also briefly touches upon complexities seen in the dataset.

Chapter 4: Proposed Method - Data Curation

This chapter explains about Data Curation tool which is developed and used extensively in this work for the creation of correct ground truth annotations.

Chapter 5: Proposed Method - Activity Recognition

This chapter explains about proposed methods and approach used in this thesis.

Chapter 6: Experiments & Results

This chapter list and compares the results from experiments conducted in methods specified in chapter 5.

Chapter 6: Discussions

In this chapter we briefly discuss our approach, results & improvements

Chapter 7: Conclusion

The conclusion of the thesis is presented here with proposed further work.

Chapter 2

Theory and Background

This chapter presents the theoretical background of this Master's thesis. The goal is that readers who are unfamiliar with the topics presented, can learn what is needed to understand the later contents of the thesis.

2.1 Deep Learning

This section gives a brief overview of the theory within the field of Deep Learning. It is intended to serve as an introduction to the field and to create a theoretical foundation on which the reader can rely for the rest of the thesis.

2.1.1 Historical Background

Since the start of the Internet, the amount of readily available general data has grown at an incredible phase. At first, this data mainly consisted of documents and web pages, but in the later years, this growth has expanded to include photos and even videos[7]. This has lead computer vision to become one of the biggest technological advances in the last decade. With a vast array of applications such as image recognition, self driving cars and surveillance computer vision has become an integral part of many business models. In many of these approaches, Image Classification plays a major role. This task is very demanding for computers, as images can contain multiple objects, be taken from different viewpoints and be occluded or severely cluttered. The goal has therefore been to develop agile algorithms capable of recognizing objects in complex scenes.

Traditionally, this was done using hand-crafted approaches such as Bag of visual Words (BovW) topped with a classifier such as a Support Vector Machine (SVM)[8]. These approaches produced the state-of-the-art results in image classification competitions such as ILSVRC[9] for several years. However, recent developments in Deep Learning has led to drastically increased performance and Deep Learning based approaches have taken over as the new state-of-the-art performers[[10], [11]].

Deep Learning is a field of Machine Learning specializing in statistical models called Deep Neural Networks. These models can learn complex hierarchical representations that correspond to multiple levels of abstractions. This is done through the use of multiple layers of non-linear processing units, called neurons, to transform data, where each layer takes the previous layers as input. This creates a flow of information, from the input through the network to the output. The way these models are able to learn such complex representations is through the use of the Backpropagation algorithm[12]. This algorithm works in several steps. First, the error, or cost, between the model output and the true output is calculated through the use of a cost function. Then the cost for each neuron in the network is calculated and propagated back through the network. The model weights are then updated based on these cost calculations, resulting in a gradual increase in performance for each weight update.

Since 2010 we have seen a drastic improvement in both natural language processing [13] and image classification through the use of Deep Learning[11], producing results that far exceed the competition. In the last five years alone, Deep Learning has completely transformed the field of computer Vision. This is not only due to the fact that these models learn so well, but also because of the introduction of modern GPUs and an exponential growth in available data[14]. Modern GPUs allow researchers to greatly parallelize the forward and backward passes through Neural Networks by utilizing the hugely parallel design of GPUs. This reduce training times by several time folds,leading to faster development and better models.

An inherent limitation of Deep Learning is the need for very large datasets for training. Since the weight update procedure has to be performed thousands if not millions of times for a even quite simple networks to converge on a good set of weights,the demand for large amount of data is obvious. Thus, with more data, we are able to explore more complex models and achieve better performance. Recently, the use of pre-trained models, already trained on large datasets have shown great results when used as a starting point for training models towards new tasks. This approach is known as Transfer Learning, where one can transfer many low-level representations learned on one dataset to another, drastically reducing the need for data. This has allowed for a much larger audience to acquire expertise and develop new models.

2.1.2 Dataset splits

The most common approach to training Deep Neural Networks is Supervised Learning. In supervised learning tasks, models learn features from labeled examples and try to approximate their predictions to the correct labels as much as possible. A common problem with this approach is a problem known as Overfitting. Overfitting happens when the model learns features that are not necessarily valid for real-world examples and become overfit to the training data. Such a model has not learned general concepts, but rather remembers the correct output for a given example in the training set. This results in poor performance in the real world. To combat the problem of overfitting it is common practice to divide the available data into three partitions, called the training set, the validation set and the testing set. It is then possible to check for overfitting during training, using the validation set performance as a guide. An important factor when this partitioning is done is to make sure that the test set is representative of the data the model will be working with when deployed. It is also important that the training set is representative of the validation set and the test set. There are many ways of separating the original dataset into training, validation and test sets, but a split of 60/20/20 or 50/25/25 are both quite common[15].

Training set

The training set is the partition used to train the model and is also, by far, the largest of the three partitions. This is a labeled set of data, containing the input data and the expected output. This expected output is then compared to the output of the model to calculate the cost for each example in the dataset during training.

Validation set

The validation set is used to validate and tune the model during the training phase. This is done by measuring the model's performance on the validation set, without allowing it to update its weights. This produces a good estimation for how well the model will perform on the test set. The performance on the validation set is also a good indication of when a model has become overfit to the training data. When the validation performance goes from increasing to decreasing during training, it usually indicates that the model has started to become overfit and further training will only further deteriorate model performance.

Test set

The test set is used fora final testing of the model. After the model has been tuned towards the optimal performance on the validation set, it is tested on the test set. This gives a good indication of how well the finished model will perform on new data and thus how well it will preform when deployed in the real world. It is very important that the test set is not used until the model has finished training and has been fully optimized towards the validation set. This is to avoid researcher bias and to ensure valid test results.

2.1.3 Neural Networks

Neural Networks are graphs that consist of one or more connected neurons,or nodes, with learnable weights W on their connections, or edges, as seen in figure 2.2. Each neuron also has a learnable bias b, which enables the neuron to activate even for zero-valued inputs. This is critical for successful learning as it helps the network to converge on a good set of weights and biases. A neuron receives a set of inputs x along its edges, computes the dot product over these inputs and its weights. It then follows it with an optional non-linear activation function f to produce an output y as shown in equation 2.1.

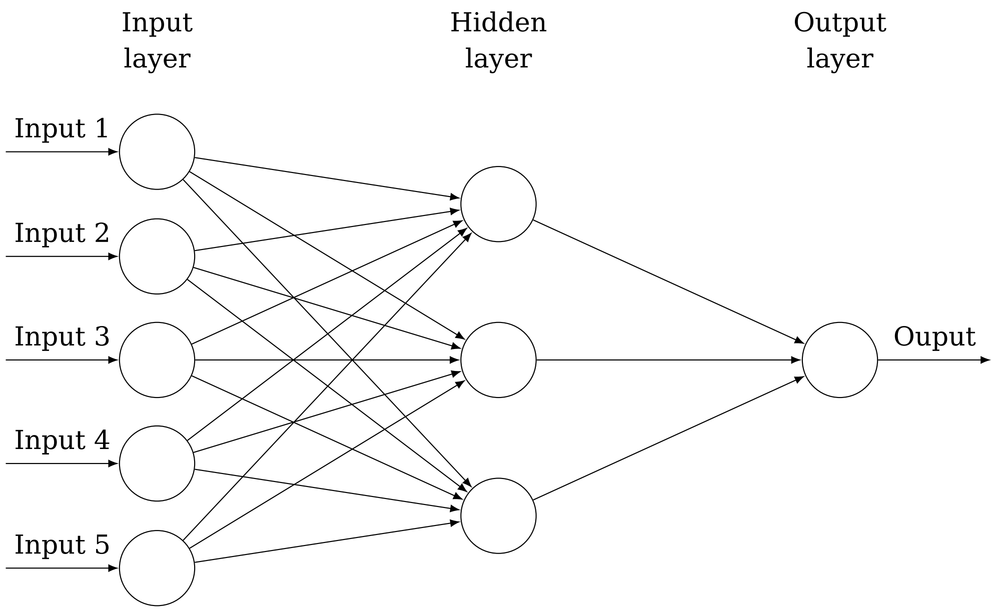


Figure 2.1: The figure shows a fully connected Neural Network with an input layer, one hidden layers and an output layer. This network is of the typical feed forward architecture, where all connections go forward through the network. Each of the connections between the neurons also have a weight W

Neural Networks are usually stacked in layers, where every layer in the network takes the previous layers as inputs. If the network consists of more layers than the input and output layers, the remaining layers are usually referred to as hidden layers, as we do not see either the input or outputs of these layers directly. An example of a simple Neural Network with one hidden layers is seen in the following equation

$$y = f(\sum_i W_i \cdot x_i + b) \tag{2.1}$$

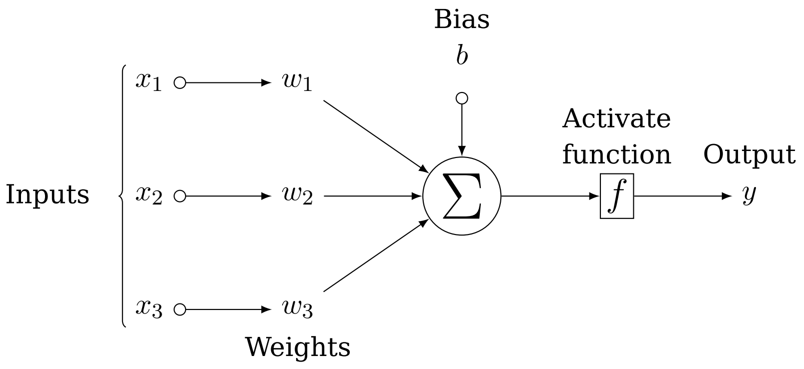


Figure 2.2: Neural Network formulation

The use of a non-linear activation function allows Neural Networks to approximate any function, including non-convex functions. The activation function takes a number and does a fixed mathematical operation on it to squash it withing a well defined range. The three most common activation functions today are:

- The Sigmoid activation function
- The Tanh activation function
- The ReLU activation function.
- The Softmax activation function.

Sigmoid

The sigmoid activation function, shown in Fig 2.3, takes a real-valued number and squashes it to arange between 0 and 1. It has the mathematical form presented in equation 2.2 This results in large positive numbers becoming 1 and large negative numbers becoming 0. The sigmoid activation function was historically frequently used since it closely resembles the firing rates of real neurons in real brains. However, it has seen a decline in the recent years due to the fact that it can kill the gradients.

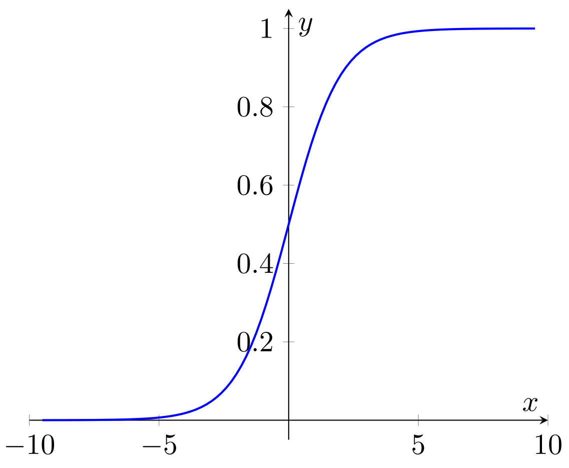


Figure 2.3: Sigmoid activation function

Since the activations of the neuron can saturate at the tails of the activation function, the gradient in these regions become very close to zero and vanish. This leads to almost no signal flow during the backpropagation phase and hence only very small or no weight updates are being performed. This in turn leads to a network that stops learning.

$$f(x) = \sigma(x) = \frac{1}{1+e^{-x}} \tag{2.2}$$

Tanh

The tanh activation function, shown in Fig 2.4, squashes a real-valued number to a range between -1 and 1. Just like the Sigmoid, this activation function suffers from the same saturation problem at its tails. The mathematical expression for tanh is shown in equation 2.3

$$f(x) = Tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{2.3}$$

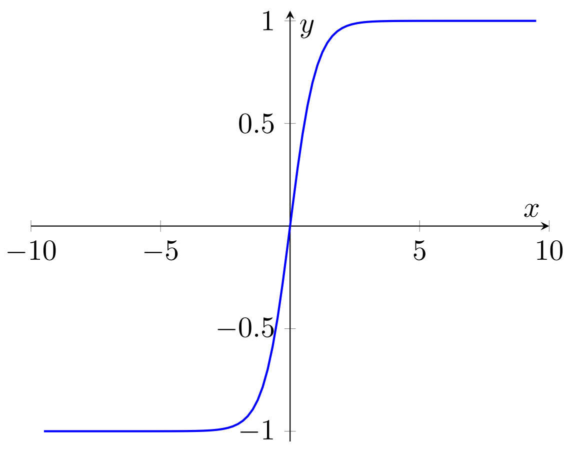


Figure 2.4: Tanh activation function

ReLU

The most popular activation function in recent years is the Rectified Linear Unit activation function as seen in Fig 2.5. The activation is thresholded at zero and has the mathematical equation shown in equation 2.4. This activation function does not suffer from the saturation problem that both the sigmoid and tanh do. This is due to its linear form and the ReLU has been shown to significantly accelerate network convergence. However, the ReLU activation function has one drawback. A large gradient flowing through a ReLU activated neuron can cause the weights to update in a way that results in the neuron never activating on a datapoint again, effectively resulting in a "dead" neuron. This is irreversible, but is somewhat avoidable by setting a good weight update parameters.

$$f(x) = \mathit{ReLU}(x) = \mathit{max}(0, x) \tag{2.4}$$

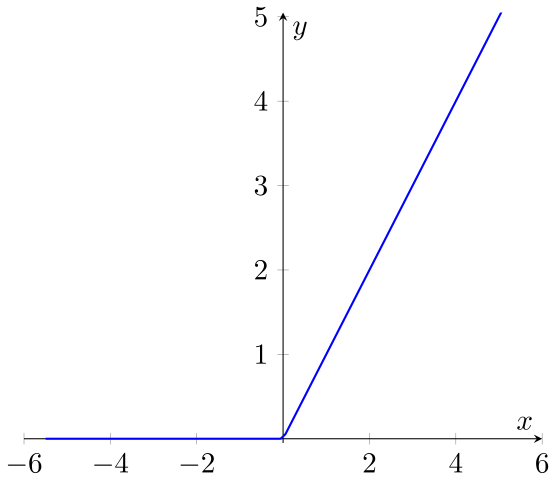


Figure 2.5: ReLU activation function

Softmax

Softmax activation function returns a probability distribution over the target classes in a multiclass classification problem.

$$\text{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \tag{2.5}$$

2.1.3.1 Loss function

The loss function can be defined as a function from a set of input values to class scores, parameterized by a set of weights W and a set of biases b. It follows from this assumption that some sets of parameters are better than others. If a network is given an image of a ball, but gives the ball class a very low score, we can assume that this particular set of parameters are not good. The loss function is therefore a measure of the quality of a particular set of parameters based on how well the network scores align with the ground truth labels in our training data. There are several types of loss functions from hinge loss[16] to cross-entropy loss[17], which all produce a loss function landscape, using all possible combinations of the parameters. This landscape can be traversed by changing the parameters of the network.

2.1.3.2 Optimization

The goal of optimization is to find the set of parameters that minimizes the loss function. This can be viewed as traversing the loss landscape, by updating the parameters, in order to find the lowest valley. There are several ways of doing this, but the most common strategy is to follow the gradient through gradient descent. To follow the gradient, we first compute the gradient of the loss function with our current set of parameters and then perform a parameter update in the negative direction of the gradient. This is done iteratively for each example or, batch of examples, until the optimal set of parameters are found. Following are some of the most commonly used optimizers:

Stochastic Gradient Descent

Gradient Descent algorithm can be done in batches or stochastically. Stochastic Gradient Descent(SGD) optimizer is an optimizer makes the neural networks converge by trying to shift towards the optimum of the cost function.For SGD, cost of one example for each step is calculated whereas in Batch gradient descent, the cost for all training examples in the dataset must be calculated. In SGD, since only one sample from the dataset is chosen at random for each iteration, the path taken by the algorithm to reach the minima is usually noisier than a typical Gradient Descent algorithm but it’s seen to be less expensive.

ADAM

Adam is a replacement optimization algorithm which uses adaptive learning rate algorithm over stochastic gradient descent for training deep learning models. ADAM requires less memory and is more efficient. It basically applies momentum over normal gradient descent algorithm. Momentum is an exponential weighted average of the gradients which makes the algorithm converge towards the minima faster.

2.1.3.3 Training Process

Training Neural Networks usually follows a set structure in which the network is fed some training data, a loss is calculated based on the outputs of the network and the true value for the data. The network then uses the backpropagation algorithm, to perform a backward pass to find the appropriate weights adjustments for all the weights and update the weights with these adjustments.

Epochs

When the network has seen all the available training data it has finished one Epoch of training. A network usually requires several epochs of training before it converges on a good set of weights.

Mini-Batches

In the earlier days of Neural Networks it was common to feed the network an individual training example, calculate the loss for this example and updating the network weights for this example through gradient descent in the backward pass. In recent years, however, it has become common to compute the loss over several training examples before preforming the backward pass. This collection of training examples is called a Mini-Batch. When using mini-batches it is very important to

2.1.3.4 Data Preprocessing

Data collected in the real world is generally suffering from several drawbacks in relation to machine learning. It may be incomplete, thus lacking values or certain attributes. It may be noisy,containing errors or statistical outliers, skewing the data. And it may be inconsistent, containing discrepancies in codes or labels, such as mislabeled data. Data preprocessing is a commonly used step to combat these issues as it transforms the raw data into an understandable format. In Deep Learning, there are several types of data preprocessing schemes, but the two most common are:

Zero-Centering

This is the most common form of preprocessing. To zero-center data, the mean is subtracted across every individual feature in the dataset. This results in centering the datacloud around the origin along all dimensions. For images it is common to perform this step by subtracting the the dataset mean from all images.

Normalization

The normalization process involves normalizing the data dimensions in order to make them approximately the same scale. The most common way of doing this is to divide each dimension by its standard deviation.

2.1.3.5 Regularization

As we described earlier, a common problem when training Neural Networks is overfitting. This happens when the network learns the details and noise in the training data to an extent that negatively impacts the model performance on the validation data. To avoid this problem, several ways of controlling the learning capacity of Neural Networks have been devised:

Dropout
Dropout is a regularization technique which involves keeping a neuron active during training with some probability p, and otherwise turning it of by setting it to zero. This essentially trains an ensemble of networks, consisting of all sub-networks that can be formed by removing non-output units from an underlying network.

Batch Normalization
Batch Normalization is a technique developed to tackle the problem of internal covariate shift in Deep Neural Networks. Internal covariate shift is the change in the distributions in network activations due to the change in network parameters during training. The Batch Normalization layer accounts for this problem through shifting its inputs to zero mean and unit variance for each mini-batch, resulting in a normalized input. The exact steps of the batch normalization transform applied to activation, x,over a mini-batch,B,is given in equation 2.6 and was first presented by Ioffe and Szegedy in [18].

Input: Values of x over a mini-batch: B = $x_{1...m}$;
Parameters to be learned : γ, β
Output :

$$y_i = BN_{\gamma,\beta}(x_i) \tag{2.6}$$

Data Augmentation
Data Augmentation is a method for boosting the size of the training set to help to avoid that the model memorizes it. There are many different ways to perform data augmentation, but it is most common to augment the data in the ways the model is supposed to be invariant to. If a model is supposed to be invariant to rotation, the

data augmentation could include various forms of rotation to the original data. Data augmentation can also be preformed online, meaning that the data is augmented with a probability p as it is being loaded, instead of having the augmented data stored. This reduces storage space and means that the model will be presented with differently augmented data every time.

2.1.4 Convolutional Neural Networks

Convolutional Neural Networks(CNNs) are very similar to regular Neural Networks as the same principles are being used and the network still expresses a single differentiable score function. The main difference lies within the fact that a CNN assumes that its inputs are matrices of numbers, such as images, for image classification, or sentence matrices, for natural language processing.

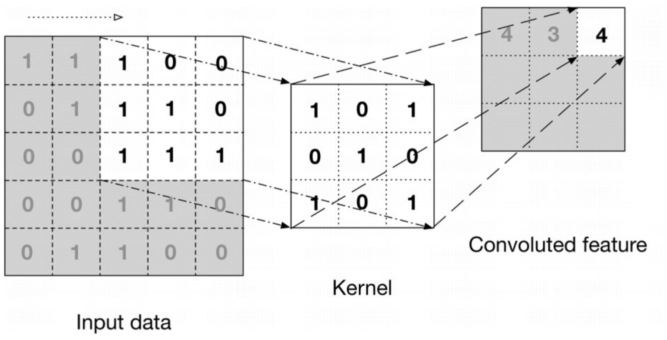


Figure 2.6: A figure showing an overview of the Convolution Operation. The activation map is computed by sliding(or convolving) the filter F over the input image I and computing the dot product between the filter and its current location on the input image. Figure adapted from [1].

This allows for the convolution operator to be encoded. CNNs consist of three main building blocks. Convolutional layers, pooling layers and fully connected layers. These layers are stacked on top of each other to form a finished CNN.

2.1.4.1 Convolutional Layers

The convolutional layers are the main layers of CNNs. These layers consist of a set of several learnable filters. The filters are slid, or convolved, over the width and height of the input volume, computing the dot product. This produces that filter's 2D activation map of the input as seen in figure 2.6. The filters act as feature extractors and activate when they see a particular type of visual feature that excites them. In the first, most basic layers, this can be edges or blobs of colors and in the later layers, we see more advanced patterns such as circles or faces. The filters, together with individual neuron biases are what is learned in the learning process for CNNs. A convolutional layer usually contains multiple different filters, which in turn produce multiple different activation maps. Thus, the convolutional layer produces a stack of these activation maps along the depth dimension called the output volume.

2.1.4.2 Pooling

It has become common practice to insert a pooling layer between a set of convolution layers in most CNNs. The pooling layer reduces the spatial size of the representation in order to reduce the number of parameters in the network. The pooling layer operates on each depth slice independently and resizes it in the spatial dimension. The most commonly used pooling version has a filter size of 2×2, a stride 2 as seen in figure 2.7. The most common pooling layer is the maxPool layer. The maxPool filter selects the maximum value over a square of 2×2 numbers and outputs that number. A stride of 2 corresponds to the filter being moved two steps to the side or down for each calculation. This results in the number of activations being decreased by 75% as seen in figure 2.7. There are also other functions such as AveragePooling and L2-NormPooling. However, MaxPooling is the preferred pooling function, as it often performs better in practice.

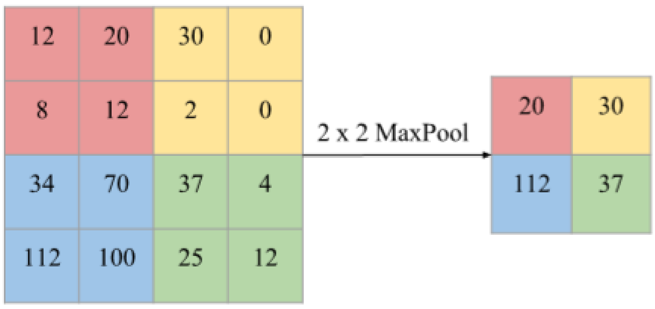


Figure 2.7: An illustration showing a 2×2 MaxPooling with stride 2. Each max is taken over a 2×2 square. The filter is then moved two squares for the next computation. Figure adapted from [2].

2.1.5 Recurrent Neural Networks

A Recurrent Neural Network works on the principle of saving the output of a particular layer and feeding this back to the input in order to predict the output of the layer. Recurrent neural networks were created because there were a few issues in the normal feed-forward neural network:

- Handling of sequential data
- Works only on current input
- Memorizing previous input

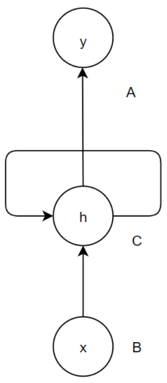


Figure 2.8: Fundamentals of Recurrent Neural Network(RNN)

The solution to these issues is the Recurrent Neural Network (RNN). An RNN can handle sequential data, accepting the current input data, and previously received inputs. RNNs can memorize previous inputs due to their internal memory. As shown in fig 2.8 “x” is the input layer, “h” is the hidden layer, and “y” is the output layer. A, B, and C are the network parameters used to improve the output of the model. At any given time t, the current input is a combination of input at x(t) and x(t-1). The output at any given time is fetched back to the network to improve on the output. Main problem with RNNs is that they suffer from short-term memory i.e. If a sequence is long enough, they’ll have a hard time carrying information from earlier time steps to later ones. So if we are trying to process a paragraph of text to do predictions, RNN’s may leave out important information from the beginning.

2.1.5.1 Long Short Term Memory(LSTM)

LSTMs were created as the solution to short-term memory problem of RNNs.They have internal mechanisms called gates that can regulate the flow of information. Long Short-Term Memory (LSTM) networks are a type of recurrent neural network capable of learning order dependencies in sequence prediction problems. The core concept of LSTM's are the cell state, and it's various gates. The cell state act as a transport highway that transfers relative information all the way down the sequence chain. It is the "memory" of the network.

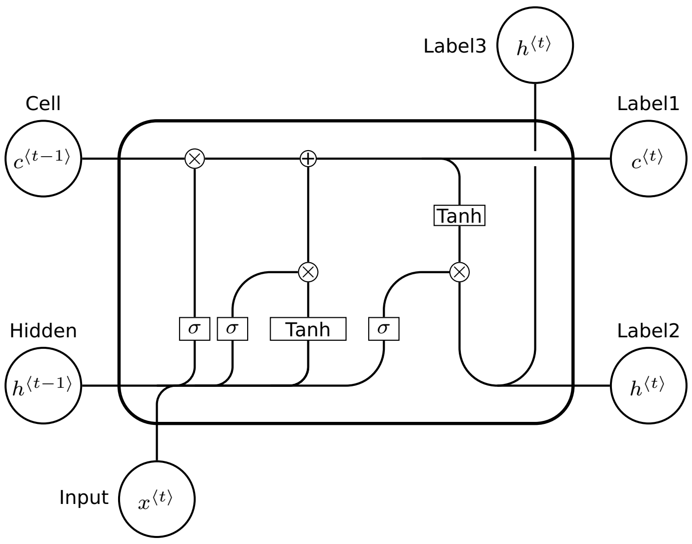


Figure 2.9: Long short term memory network

The cell state can carry relevant information throughout the processing of the sequence. So even information from the earlier time steps can make it's way to later time steps, reducing the effects of short-term memory. As the cell state goes on its journey, information get's added or removed to the cell state via gates. The gates are different neural networks that decide which information is allowed on the cell state. The gates can learn what information is relevant to keep or forget during training. There are three different gates that regulate information flow in an LSTM cell. A forget gate, input gate, and output gate.

Forget gate

This gate decides what information should be thrown away or kept. Information from the previous hidden state and information from the current input is passed through the sigmoid function. Values come out between 0 and 1. The closer to 0 means to forget, and the closer to 1 means to keep.

Input gate

To update the cell state, we have the input gate. First, we pass the previous hidden state and current input into a sigmoid function that decides which values will be updated by transforming the values to be between 0 and 1 where 0 means not important, and 1 means important.

Output Gate

Last we have the output gate. The output gate decides what the next hidden state should be. Hidden state is the state which contains information on previous inputs. The hidden state is also used for predictions. The output is the hidden state. The new cell state and the new hidden is then carried over to the next time step.

2.1.6 3D-Convolutions

Traditional CNNs are two-dimensional CNNs. This means that they are using 2D filters and produce a 3D volume of 2D depth slices as their output. It is, however, very possible to extend this type of layer to become three-dimensional Convolutional Layers. This is done by increasing the dimensionality of the filters to 3D and increasing the dimensionality of the input. This results in a 4D volume of 3D depth cubes as the output. For videos, this can be done by stacking sequential video frames together, producing a cube of frames as the input. The 3D filters are then convoluted over this cube in both the spatial and depth dimensions. This produces depth slices that not only learn features in a single image, but also how they transform through time in a video. This results the network learning spatio temporal filters that are able to extract useful features in both space and time.

2.2 Transfer Learning

Transfer learning is about leveraging feature representations from a pre-trained model, so that there’s no need to train a new model from scratch. The pre-trained models are usually trained on massive datasets that are a standard benchmark in the computer vision frontier. The weights obtained from the models can be reused in other computer vision tasks. These models can be used directly in making predictions on new tasks or integrated into the process of training a new model. Including the pre-trained models in a new model leads to lower training time and lower generalization error. Transfer learning as a technique is particularly very useful when the training dataset is small.

2.3 Performance Metrics

Performance of a data model is a direct way to measure its accuracy. In this thesis we are trying to solve a classification problem where a true class label is assigned to a video clip contains a washing hands activity and a false class label is assigned to the clip which do not contain any washing hands activity. This is a typical Binary classification problem, where we can only have two possible labels. Generally speaking, a yes/no question or a setting with 0-1 outcome can be modeled as a binary classification problem.

2.3.1 Confusion Matrix

Confusion Matrix is a tabular visualization of the ground-truth labels versus model predictions. Each row of the confusion matrix represents the instances in a predicted class and each column represents the instances in an actual class. Confusion Matrix is not exactly a performance metric but sort of a basis on which other metrics evaluate the results. Each cell in the confusion matrix represents an evaluation factor.

True Positive(TP) signifies how many positive class samples predicted correctly by the model.
True Negative(TN) signifies how many negative class samples predicted correctly by the model.
False Positive(FP) signifies how many negative class samples predicted incorrectly by the model. This factor represents Type-1 error in statistical nomenclature. This error positioning in the confusion matrix depends on the choice of the null hypothesis.

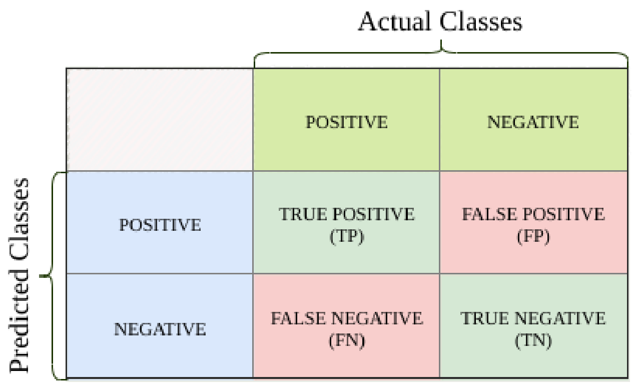


Figure 2.10: Confusion Matrix

.

False Negative(FN) signifies how many positive class samples predicted incorrectly by the model. This factor represents Type-II error in statistical nomenclature. This error positioning in the confusion matrix also depends on the choice of the null hypothesis.

2.3.2 Binary Classification Measures

Accuracy

The most simple and straightforward classification metric is accuracy. Accuracy measures the fraction of correctly classified observations. The formula is:

$$Accuracy = \frac{TP + TN}{samples} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.7}$$

Precision and Recall

An alternative measure to accuracy is precision. Precision is the fraction of instances marked as positive that are actually positive. In other words, precision measures “how useful are the results of our classifier”. The mathematical notation is:

$$Precision = \frac{TP}{TP + FP} \tag{2.8}$$

Recall is the fraction of true positive instances that are marked to be positive. It measures “how complete the results are” — that is, which percentage of true positives are predicted as positive. The representation is:

$$Recall = \frac{TP}{TP + FN} \tag{2.9}$$

Perfect precision is equivalent to no FPs (no Type I errors), while on the other hand, perfect recall means there are no FNs (no Type II errors).

F1-Score

F-1 score is the harmonic mean of precision and recall. It gives equal importance to Type I and Type II errors. The calculation is:

$$F\text{-}1\ Score = \frac{2\ \times\ Precision\ \times\ Recall}{Precision\ +\ Recall} \tag{2.10}$$

When the dataset labels are evenly distributed, accuracy gives meaningful results. But if the dataset is imbalanced, F-1 score measure is preferred.

ROC & AUC Curves

A well-known method to visualize the classification performance is a ROC curve (receiver operating characteristic curve). The plot of ROC curve shows the classifier’s success for different threshold values. Lowering the classification threshold classifies more items as positive, thus increasing both False Positives and True Positives. True positive rate is also known as Sensitivity where as True Negative rate is called Specificity. Sensitivity tells us what proportion of the positive class got correctly classified whereas specificity tells us what proportion of the negative class got correctly classified. In order to plot the ROC curve, we need to calculate the True Positive Rate (TPR) or Sensitivity and the False Positive Rate (FPR), where:

$$TPR = \frac{TP}{TP + FN} \tag{2.11}$$

$$FPR = \frac{FP}{FP + TN} \tag{2.12}$$

2.4 Human activity recognition

This section will give a brief overview of the field of Human Activity Recognition. Action Recognition and Action Detection Human activity recognition has spiked interests in several industries involved with computer vision in recent years. Human activity recognition is a field concerned with classifying human actions performed in videos. It can be separated into two subtasks:

- Human Action Recognition
- Human Action Localization

Human Action Recognition

Human Action Recognition involves classifying individual videos. For this task, each video contains only one class of action, and the goal is for the system to accurately classify the action performed in the video.

Human Action Localization

Human Action Localization, on the other hand, is concerned with detecting actions through continuous videos. This means that any given video contains multiple classes of actions and the goal of the system is to accurately segment the video into correctly classified segments. The field Human Activity Recognition has become an important research domain, spanning different applications, such as sport analysis, human computer interaction, and video surveillance. It is also of general interest to the field of computer vision as it expands the ability of machines to understand the contents of video. There are several standardized datasets for human Activity Recognition, but the most commonly used are the UCF-101[19] and Sports1M[20] Action Recognition datasets. These datasets include videos of different humans, performing several classes of actions from several different angles under a variety of conditions. Human Activity Recognition is considerably more challenging than regular image classification, as it relies on videos for inputs. This combines the challenges of both image recognition and sequence handling, as videos are constructed of sequences of single image frames. Since 2012, we have seen complete domination in both image recognition tasks and natural language processing tasks through the use of CNNs and RNNs. This has lead researchers to believe that a combination of these techniques could do the same for Human Action Recognition. Thus, recent years have seen a dramatic increase in use of Deep Learning architectures for Human Action Recognition tasks.

2.5 Related work

Video understanding is one of the core computer vision problems and has been studied for decades. Many research contributions in video understanding have focused on developing spatio-temporal features for videos. In this section, we review previous works that is closely related to this thesis. Following sections describes some of the interesting research work carried out on action recognition and action localization

2.5.1 Action Recognition

There have been a significant research and development in the field of human action recognition in videos. Earlier methods mainly involved human body parts tracking and human motion analysis [21]. Follow up methods focused on statistical representations for action recognition. Laptev 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, Bag-of-Words approach has been used to represent actions in the video. Schuldt et al.[22] classified actions by applying Support Vector Machines (SVMs) on the occurrence histograms. In general, SVM has been extensively used for classification, regression, novelty detection tasks, and feature reduction. It has been seen that It performs on par or marginally inferior to existing systems, when the number of training examples are a few due to the imbalance, although consistently better in terms of computation time. Wang et al.[23] 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][4] 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. Action recognition in stills images received less attention compared to videos. The work of Ikizler et al. [2008][24] 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. More recently, Tran et al. [2015][3] 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][25] 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, HMDB and Kinetics. 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 datasets achieve state-of-the-art performance on the both action recognition benchmarks as depicted in one of the earliest work on 3D Convnets[26]. This thesis contains experiments with both C3D and I3D networks are explained in more detail in the following sections. 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.

2.5.2 C3D Network

C3D are deep 3-dimensional convolutional neural networks with a homogenous architecture containing convolutional kernels followed by pooling at each layer. C3D network is well-suited for spatio-temporal feature learning compared to 2D ConvNet. C3D is commonly known for generic feature extraction where 3D convolutions extracts both spatial and temporal components relating to motion of objects, human actions, human-scene or human-object interaction and appearance of those objects, humans and scenes. It also has an ability to model temporal information better due to 3D convolutions and 3D pooling operations. In 3D ConvNets, convolution and pooling operations are performed spatio-temporally while in 2D ConvNets they are done only spatially. 2D convolution applied on an image will output an image, 2D convolution applied on multiple images (treating them as different channels also results in an image. Hence, 2D ConvNets lose temporal information of the input signal right after every convolution operation. Only 3D convolution preserves the temporal information of the input signals resulting in an output volume. The same phenomena is applicable for 2D and 3D polling.

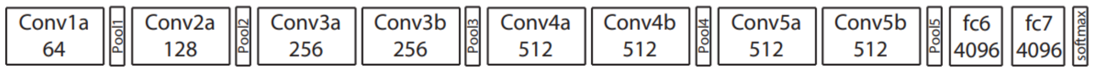


Figure 2.11: C3D Architecture[3]

C3D net has 8 convolution, 5 max-pooling, and 2 fully connected layers, followed by a softmax output layer. All 3D convolution kernels are $3 \times 3 \times 3$ with stride 1 in both spatial and temporal dimensions. Number of filters are denoted in each box. The 3D pooling layers are denoted from pool1 to pool5. All pooling kernels are $2 \times 2 \times 2$, except for pool1 which is $1 \times 2 \times 2$. The C3D model is given an input video segment of 16 frames and the outputs a 4096-element vector. Due to the compactness of C3D architecture, it's considered as a faster and efficient way to handle processing of real-time feeds. C3D starts by focusing on appearance in the first few frames and tracks the salient motion in the subsequent frames. C3D is generally also used as a feature extractor for various classification and action recognition tasks.

2.5.3 Inception 3D Network

A sucessfull 3D CNN architecture used in activity recognition is the Inception 3D (I3D) developed by Deepmind[27] and Carreira et. al [26]. Inception is a deep convolutional neural network architecture that was first introduced in 2014. It won the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC14). I3D is a two-stream activity recognition network based on the well-known CNN Inception v1 architecture. I3D recognizes activities by analyzing the temporal changes in RGB representation and optical flow representation of images in short video clips.

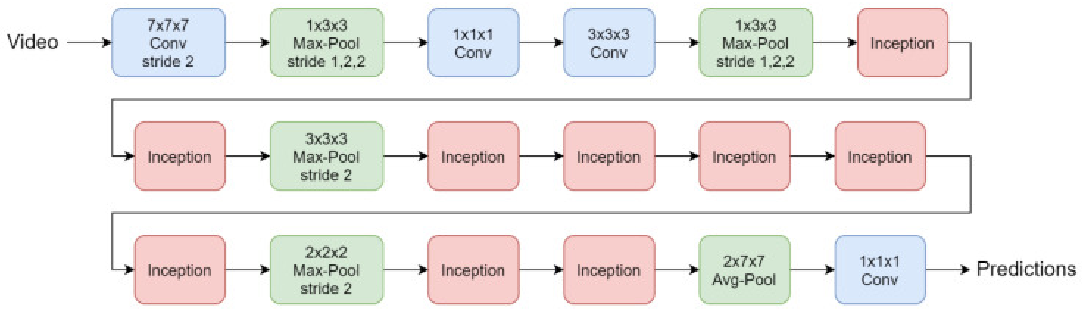


Figure 2.12: Inflated 3D (I3D) network architecture [4]

The architecture of I3D is created by inflating all the filters and pooling kernels in Inception v1 into a 3D CNN. Squared filters of size NxN is made cubic and becomes NxNxN filters. The pre-trained ImageNet weights from Inception v1 are repeated along the inflated time dimension and rescaled by normalization over N. The inflated version is further trained on the large activity recognition dataset, Kinetics[28] Dataset which has 400 different classes and over 400 clips per class collected from realistic, challenging YouTube videos. During training, a fixed length clip is forwarded trough the network and the class prediction is compared to the clip‘s true label.

Carreira et. al demonstrated that 3D CNN can benefit from pre-trained 2D CNN, and that transfer learning is highly efficient also in activity recognition. I3D network pre-trained on kinetics[29] provided state-of-the-art results on the activity recognition dataset called UCF-101.

The I3D neural network adds a convolution operation for adjacent temporal information, which can complete the action recognition of continuous frame. In order to expedite the deep network training speed, a batch regularization module is added to the network. Since the network is not sensitive to the initialization, so a larger learning rate can be employed. I3D increases the depth of the network, eight convolutional layers and five pooling layers are used. The size of the convolution kernel of each convolutional layer is 3 × 3 × 3, and the step size is 1 × 1 × 1 respectively, the number of filters is 64, 128, 256, 256, 512, 512.

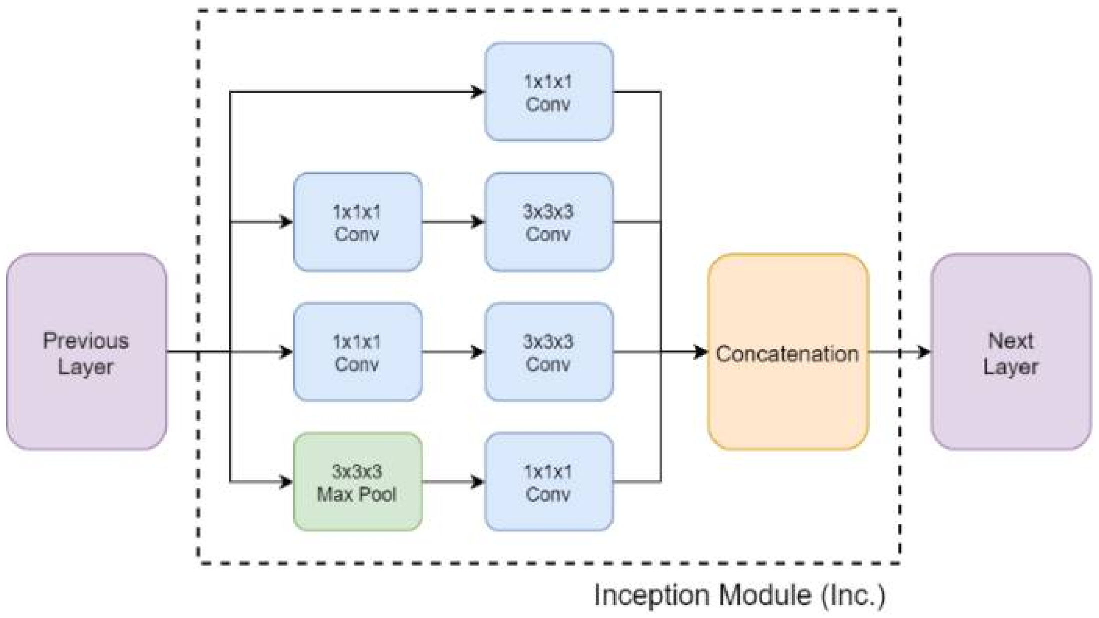


Figure 2.13: Inception module network architecture [4]

2.5.4 Action Localization

Many researchers have concentrated on temporal action localization in long untrimmed videos. Jain et al[30] introduced a sampling strategy to produce tubelets with motion information from super-voxels. Many state-of-the-art approaches for spatio-temporal action localization rely on detections at the frame level that are then linked or tracked across time. Goal of this theis is to leverage the temporal continuity of videos instead of operating at the frame level.

Deep convolutional neural networks (CNN) have demonstrated breakthrough performance for image feature extraction, more and more studies of temporal action localization focus on deep learning. Shou et al. used 3D ConvNets [31] 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. The fully end-to-end network takes a long video as input and outputs the temporal bounds of all action instances. Hou et al.[32] proposed a tube convolutional neural network to localize actions based on 3D convolution features. In certain developments reasearchers also tried to combine 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.

2.6 Video Classification Overview

Video Classification is the task of producing a label that is relevant to the video given its frames. A good video level classifier is one that not only provides accurate frame labels, but also best describes the entire video given the features and the annotations of the various frames in the video. One of the most important components of any Deep Learning project is an understanding of the input data being used.

Convolutional Network is modified to account for the temporal dependencies in videos. Usually, a stack of frames is concatenated on top of each other and inputted to the CNN. Classically, a CNN takes as input a (height x width x color channels) matrix. For example, this could be a 224 x 224 x 3 input tensor. In these experiments, previous frames are stacked on top of the color channel axis such that an input consisting of two images frames in the video has the shape 224 x 224 x 6. Karpathy et al.[20] propose 3 different strategies for combining frames as input to the CNN and contrast these approaches with a baseline model of classifying frames one at a time.

The Single Frame model is an example of classifying videos by simply aggregating predictions across single frames/images. The Late Fusion model combines frames by concatenating the first and last frame in the clip. The Early Fusion model takes a larger contiguous segment from the clip. Lastly, the Slow Fusion model has a much more sophisticated scheme in which 4 partially overlapping contiguous segments are progressively combined in the Convolutional Layers. Experimentation found the most individual success with the Slow Fusion strategy, although not substantially greater than the Single Frame model. The best overall results were found by averaging results across all models, (Single + Early + Late + Slow).

2.6.1 Data Curation

Data Curation is the process of discovering, integrating, and cleaning data and is one of the oldest, hardest, yet inevitable data management problems. Despite decades of efforts from both researchers and practitioners, it is still one of the most time consuming and least enjoyable work of data scientists. In most organizations, data curation plays an important role so as to fully unlock the value of big data. Unfortunately, the current solutions are not keeping up with the ever-changing data ecosystem, because they often require substantially high human cost. Meanwhile, deep learning is making strides in achieving remarkable successes in multiple areas, such as image recognition, natural language processing, and speech recognition[33].

2.6.2 Video classification architecture

A typical video classification architecture usually follow the basic steps defined in the Fig 2.14. First step involves collection of data in form of video, text, speech and image from various sources, databases or cloud. Dataset is then analyzed, cleaned and converted in subsequent steps. In data curation step(as explained in section 2.6.1), different types of data pre-processing tasks like enhancing, smoothing and noise reduction can be performed to improve the efficiency and accuracy of the classification task. Finally, a network model is fitted with tuned parameters that results in feature generation or classification.

2.7 Deep Learning Development Platforms

This section will give an overview of the Deep Learning libraries used during the implementation and testing phases of this thesis. All models were implemented using the Python APIs of the libraries presented.

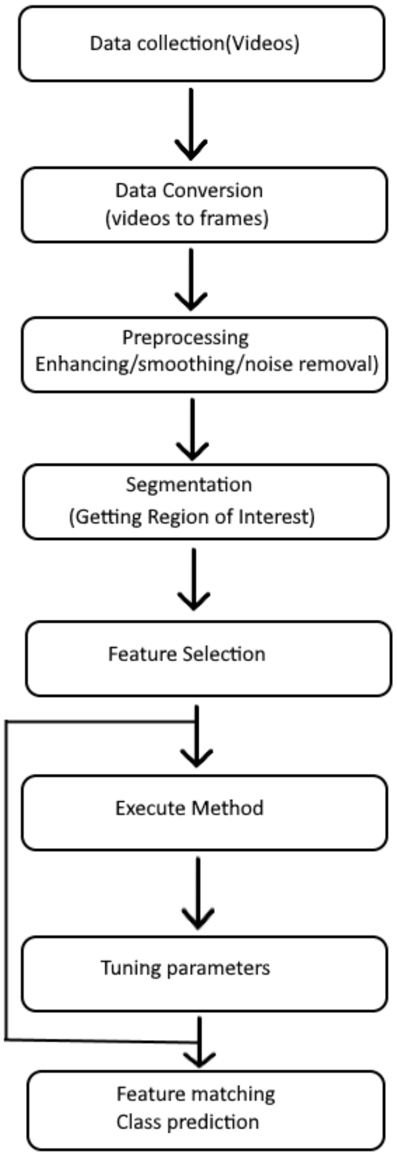


Figure 2.14: Basic steps in video classification

2.7.1 TensorFlow™

TensorFlow™[34] is an open source Machine Learning library developed by Google to meet their needs of a system capable of developing and testing Neural Networks. It uses data flow graphs to do numerical computations, where nodes and edges in the graph represent mathematical operations and tensors respectively. It allows the user to run code on both CPU and GPU, enabling faster computations through parallelization. TensorFlow provides an extensive suite of functions and classes that allow users to build models from scratch with abundant customization options.

TensorFlow also facilitate making checkpoints when performing experiments and an extensive amount of visualization options, making it a natural choice for research.

2.7.2 Keras

Keras[35] is the high-level API of TensorFlow 2: an approachable, highly-productive interface for solving machine learning problems, with a focus on modern deep learning. It provides essential abstractions and building blocks for developing and shipping machine learning solutions with high iteration velocity.

Chapter 3

Data Material

3.1 Dataset Overview

Dataset used in this thesis comprises of 4800 untrimmed video clips with annotations. These videos were captured using different types of cameras installed at different locations in a training & simulation facility. Based on observations we see that a particular training session can contain videos captured from training or simulation area, evaluation area, entry and exit points, patient monitoring screen. Fig 3.1 shows an overall data allocation into various categories.

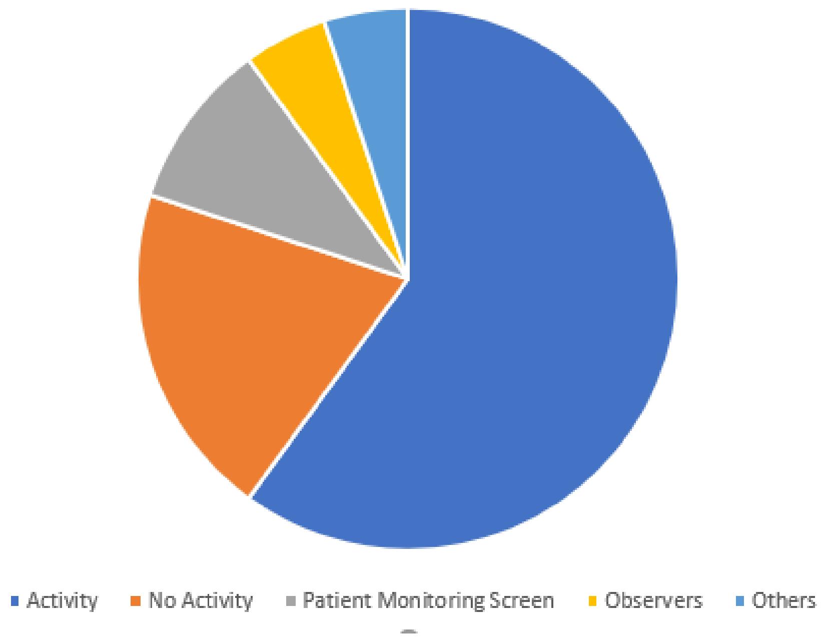


Figure 3.1: Initial data allocation under different categories.

Since the main objective of this thesis is to identify and recognise washing hands activity, those videos which fall into the Patient Monitoring Screen(PMS), Observers & Others are not relevant for this work. Hence, We make use of only Activity and a portion of No Activity videos for training and testing. Fig 3.2 shows an example of video clips that required filtering or correct categorization during the data preparation phase.

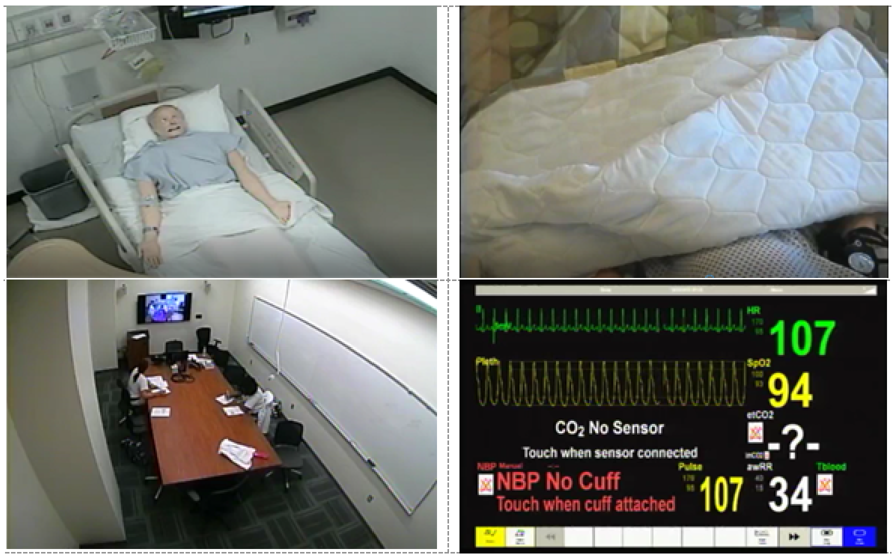


Figure 3.2: Shows different types of video clips in the dataset. Starting at top-left - No activity, others(camera angle), observers, patient monitoring screen respectively

Annotation provided with the dataset contains timestamps for Washing Hands activity which are manually annotated by an observer using an app. As shown in the listing 3.1, there could be multiple annotations per session where each session is composed of multiple videos captured using cameras installed at various locations in a given setting. For e.g. a static camera no 1 producing video clip SC01.m4v constantly points toward a manikin/patient bed, Camera no 2 producing SC02.m4v captures the activities performed by HCP from top view, Camera no 3 does the same from side view and Camera no 4 is installed in a room where observers are remotely observing or evaluating the HCP. We converted raw annotations to JSON for easy object based separation and interpretability.

```
{
  "SessionID": "file",
  "VideoFolder": "howardcc",
  "VideoFiles": "SC01.m4v,SC02.m4v,SC03.m4v,SC04.m4v",
  "Annotations": "wash hands (at 11 seconds),wash hands (at 120
    seconds)"
}
```

Listing 3.1: Annotation JSON Data Format

Fig 3.3 shows an example of washing hands activity performed by a single actor. There are many instances where multiple actors are seen performing different actions.

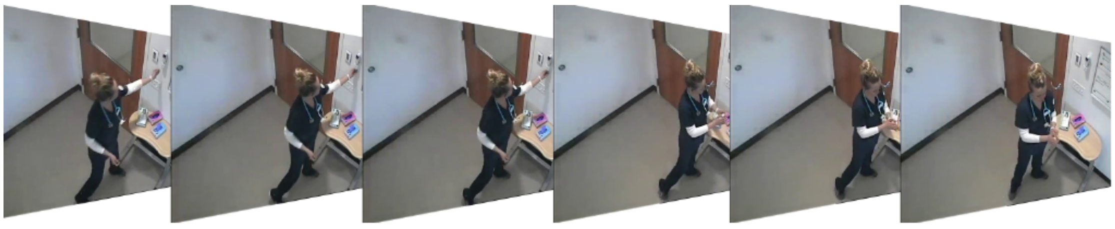


Figure 3.3: Trimmed video frames showing washing hands activity.

3.2 Dataset Challenges

Proposed deep Neural network methods in Chapter 5 required to be trained on a fixed length and size (i.e. fixed number of frames). Also, it is very important for the network to train and learn a specific type of activity. Hence, it becomes absolute necessity to perform a data curating step in the beginning which makes it easier to label the dataset in a supervised learning setting. Following are some of the most common challenges encountered with the dataset.

- Long untrimmed videos with varying length and incorrect annotations.
- Session wide annotations i.e. as seen in listing 3.1, each session contains multiple text based annotations with no specific reference to a particular video(s) containing the washing hands activity. Hence, there is a need to develop a utility which can crop each video clip with $\pm 5\,sec$ window on either side of annotated timestamp.
- Multiple annotation per session - We needed to go a step further with the clip extraction utility to crop a given video at multiple time step windows.

- Performed activity could not be seen due to camera angle, perspective and video quality
- Multiple actors performing different activities at the same time

In order to overcome the challenges with the given annotations, a **Data Curation Tool** 4 was developed and used.

3.3 Data Source & Extraction

In order to download the videos dataset by looping over existing annotations, a AWSDownloaderUtility was developed and used as a starting point for this thesis. This section depicts the procedure followed for downloading and extracting the necessary video clips. Initial video extraction process clips a slightly larger window of $\pm 5\,sec$ in order to remove the human punching error in the provided timestamps thereby serves the purpose of refining the ground-truth labels

Algorithm 1: Extract trimmed video clips

```
Result:
Input: Annotation file
Output: Trimmed videos
Establish connection to cloud repo;
Read annotation file;
for each session in Sessions do
    Extract annotations;
    for each annotation in annotations do
        for each clip in Session do
            Clip video with annotation-5 and annotation+5 seconds ;
            Extract clip into a specific folder;
        end
    end
end
```

3.4 Dataset splits

When splitting the original dataset into training, validation and test sets, we decided to split the video dataset based on available simulation environments so as to include a fair amount of examples from a each such environment. The reasoning behind this split was two fold:

1. The conditions for a particular environment might just overfit the model.
2. Splitting the dataset based on a specific simulation environment gives the best representation of the performance that can be expected if the model is deployed in a different setting and starts receiving new video data. However, splitting the data in this fashion also creates more challenging validation and test sets. Such a consideration also increase the possibility of the validation and test sets containing conditions, not well represented in the training set. This split was chosen intentionally in order to give a good representation of performance on new videos and to prove the robustness of our architectures through a challenging dataset split.

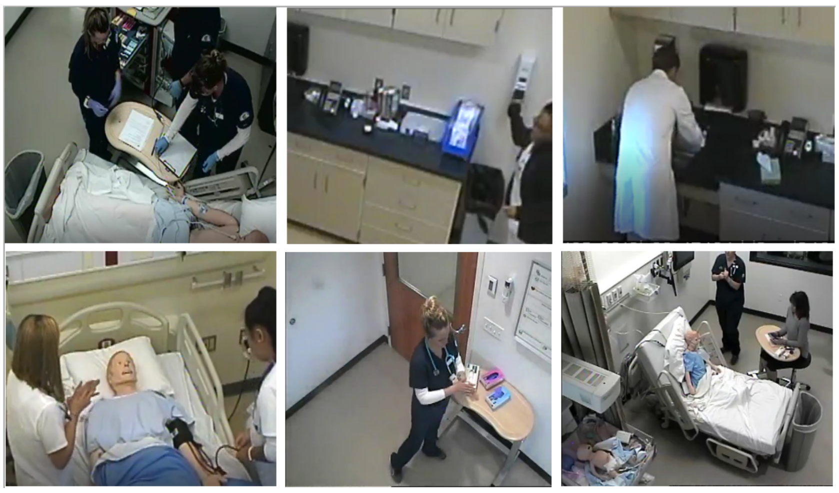


Figure 3.4: Shows different simulation/training environments

Although the dataset is quite large as a whole but, it still poses some challenges related to filtering relevant videos and then labeling them correctly. Most importantly, the distribution between **Washing hands** and **Not Washing hands** videos is not equal. Chapter 4 describes about Data Curation(DC) tool which was specifically developed and used to reduce the size of original 10.6 hours of training examples with 15 seconds clips down to roughly 3.5 hours with 3 sec action snippets. Using this approach, we increase the size of training examples and feed the models with relevant and concrete examples rather than using a fewer clips with increased background details. Also, in order to avoid our models to develop a learning bias towards Non washing hands behaviour, we are dependent on having an equal amount of both Washing and Not Washing hands frames in our training set. We also want an balanced distribution of Washing and Not Washing hands frames in our validation set in order to get an accurate representation of performance.

Another major factor is the length of each video which is not fixed. Since the videos ranges from 1000 to 20000 frames per video this produced some difficulties when splitting the data. The size of the split was set to approximate a 60%, 20%, 20% split, as much as possible. However to ensure an accurate representation of the models performance on new data, we included videos from both HowardCC and CMC hospitals. To achieve the test set size we wanted, we splitted long untrimmed videos into multiple sub-clips. The final dataset contains 3.5 hours of videos in form of 4000 video clips of 3 sec length, split over training, validation and test datasets of size 64%, 18%, 18% respectively. The dataset split is depicted in the following table.

Dataset Name	**Number of minutes**	**Number of frames**	**% of total dataset**
Training	122.01	122100	64%
Validation	41.04	41400	18%
Test	41.04	41400	18%
Total	**204.09**	**204900**	**100.00%**

Table 3.1: Shows break-up of final dataset

Chapter 4

Proposed Method - Data Curation

Due to noticeable challenges mentioned in Section 3.2, a Data curation tool was developed and used extensively in this thesis. Data Curation tool is a user-friendly graphical user interface based application which enables users to easily assign a class label to a video clip.

4.1 Data curation pipeline

Data Discovery is the process of identifying relevant data for a specific task. Typically data is often scattered across a large number of tables that could range in the tens of thousands. As an example, the user might be interested in identifying all tables describing user studies involving insulin. The typical approach involves asking an expert or performing manual data exploration [8]. Data discovery could be considered as identifying tables that match a user specification (e.g., a keyword or an SQL query).

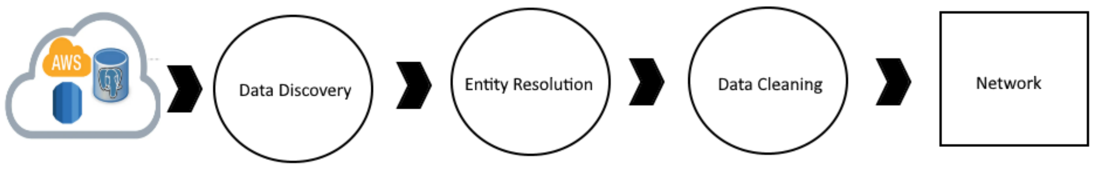


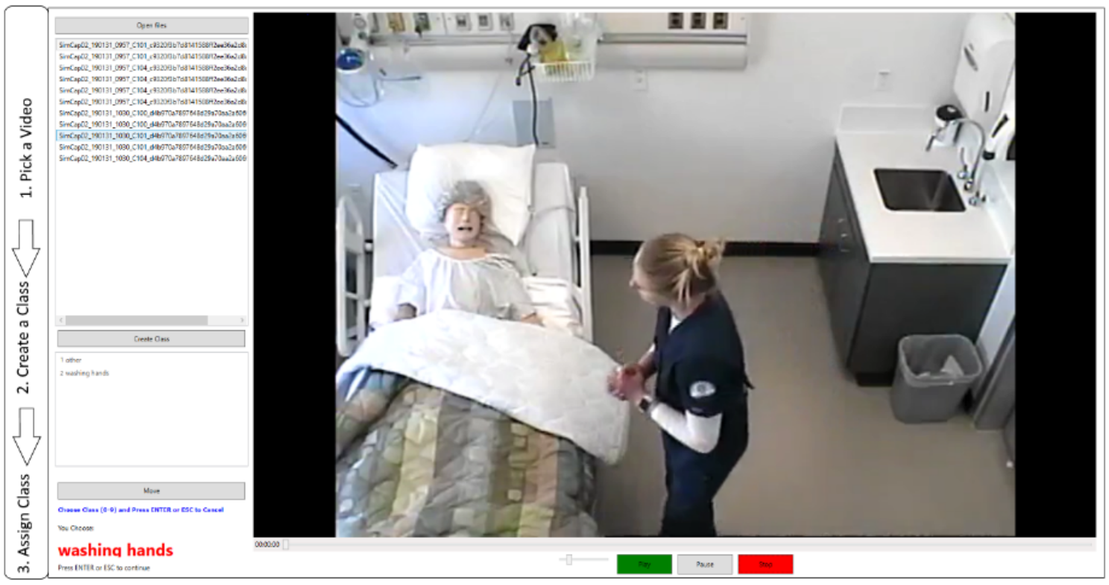
Figure 4.1: Data curation pipeline

4.1.1 Data discovery

With regards to the dataset used for this work, we start the process by discovering and matching video files using ground-truth annotation data and looping over all evaluation sessions. Typical process of data discovery includes extraction, filtering and clipping the files for necessary time interval i.e $\pm 5\,sec$.

4.1.2 Entity resolution

Entity resolution (ER) is the task of disambiguating records that correspond to real world entities across and within datasets. One of the major goal and preliminary contribution involves development of a tool which helps classify data manually through human intervention. This manual labelling tools allows user to load a set of video files, create a number of classes and assign a class per video. In order to achieve efficiency while annotating the videos, this tool listens to keypress events, increase/decrease video playback speed and moves on with the next video in the playlist while filtering out video clips which have already been assigned a label.



SimCap02_190131_0957_C101_c9320f3b7d8141588f2ee36a2c8
SimCap02_190131_0957_C101_c9320f3b7d8141588f2ee36a2c8
SimCap02_190131_0957_C104_c9320f3b7d8141588f2ee36a2c8
SimCap02_190131_0957_C104_c9320f3b7d8141588f2ee36a2c8
SimCap02_190131_0957_C104_c9320f3b7d8141588f2ee36a2c8
SimCap02_190131_0957_C104_c9320f3b7d8141588f2ee36a2c8
SimCap02_190131_1030_C100_d4b970a7897648d29a70aa2a606
SimCap02_190131_1030_C100_d4b970a7897648d29a70aa2a606
SimCap02_190131_1030_C101_d4b970a7897648d29a70aa2a606
SimCap02_190131_1030_C101_d4b970a7897648d29a70aa2a606
SimCap02_190131_1030_C104_d4b970a7897648d29a70aa2a606

1 other
2 washing hands

Choose Class (0-9) and Press ENTER or ESC to Cancel

You Choose:

washing hands

Press ENTER or ESC to continue

00:00:00

Figure 4.2: Graphical User Interface - Data Curation Tool

Data Curation Tool

https://vimeo.com/543367628

Fig 4.3 shows flowchart for Data Curation Tool in detail.

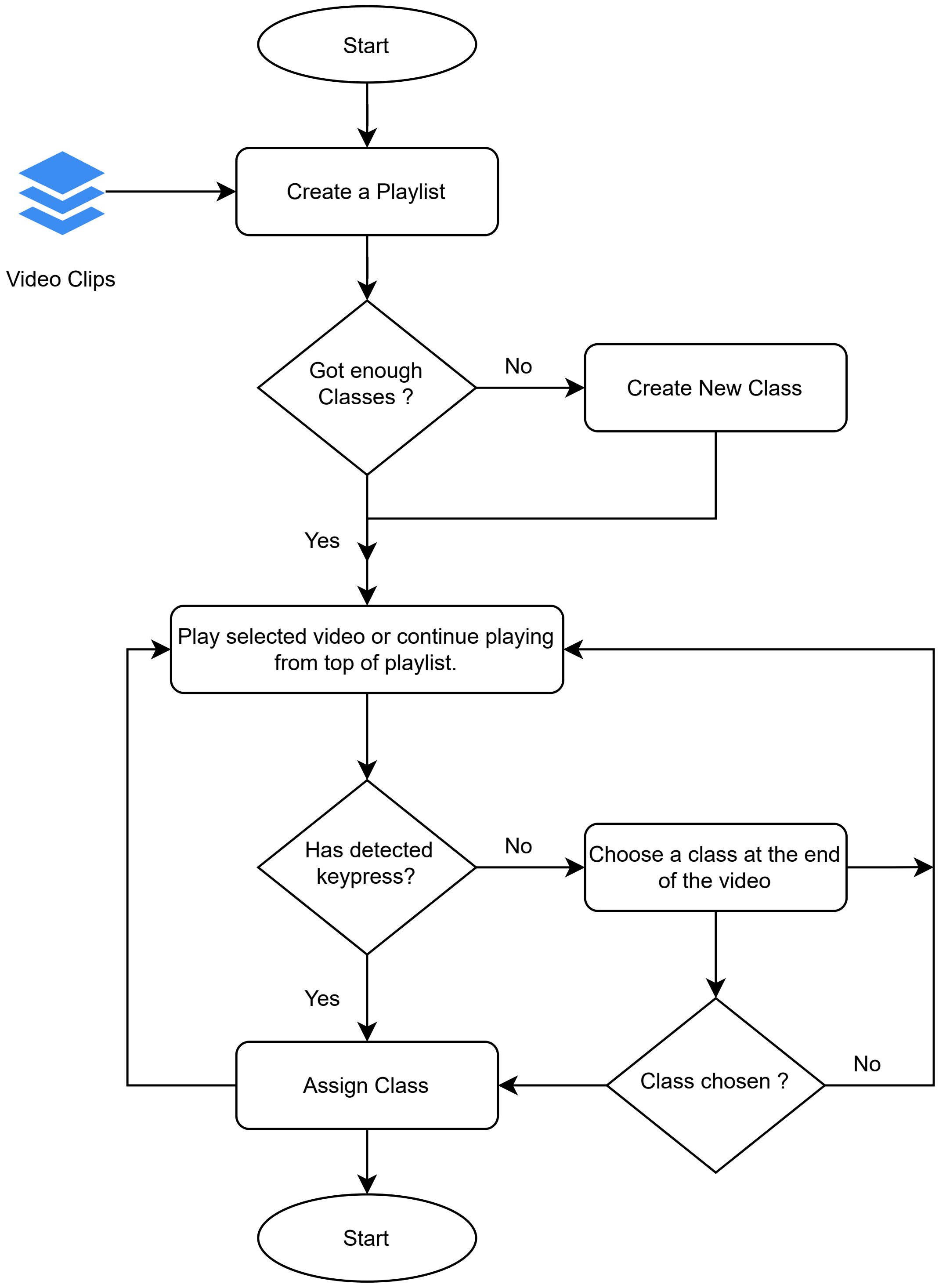


Figure 4.3: Flowchart for Data Curation Tool

Additional functionality which allows generation of temporal points was also developed in Data Curation tool. **Temporal point(s)** are the moments in the video clips indicating an action instance. This corresponds to the scenario where an annotator would simply click press a key to indicate an action while the video is being played, instead of precisely specifying the time boundaries. We then create a candidate interval around that time point with a fixed size of 45 frames in 3 seconds i.e 15fps. Program also allows annotators to find continuous regions where they can input multiple key strokes over a temporal domain while the activity is being performed. We then extract the region based on continuity of annotations over an interval. Region of non-continuous annotation portion is clipped out as an instance of negative bag whereas a positive instance contains the clips extracted within a boundary of 3 sec.

4.1.3 Data cleaning

Data Cleaning means the process of identifying the incorrect, incomplete, inaccurate, irrelevant or missing part of the data and then modifying, replacing or deleting them according to the necessity. As a part of data cleansing process, we remove irrelevant video files for e.g. videos showing only monitoring screen, manikin, only audio etc.

Chapter 5

Proposed Method - Activity Recognition

This chapter presents an overview of the proposed method and explains training & testing pipeline with the implementation details.

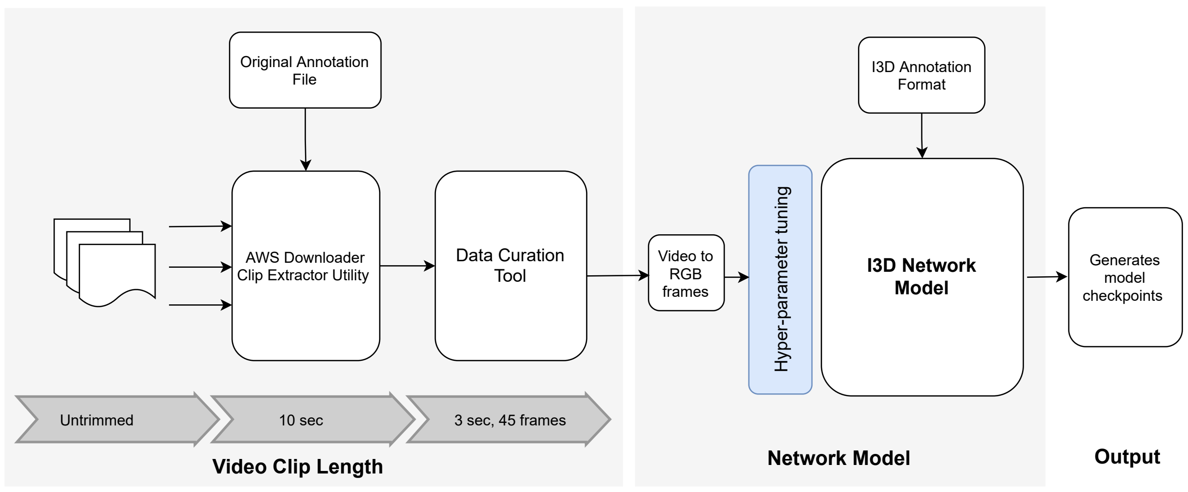


Figure 5.1: Pipeline for proposed method

5.1 Method Overview

Fig 5.1 presents proposed pipeline depicting an overall approach applied in this work. Proposed pipeline can easily be explained in two steps : 1) Data Preparation stage 2) Activity detection. Overall approach is explained in the following sections.

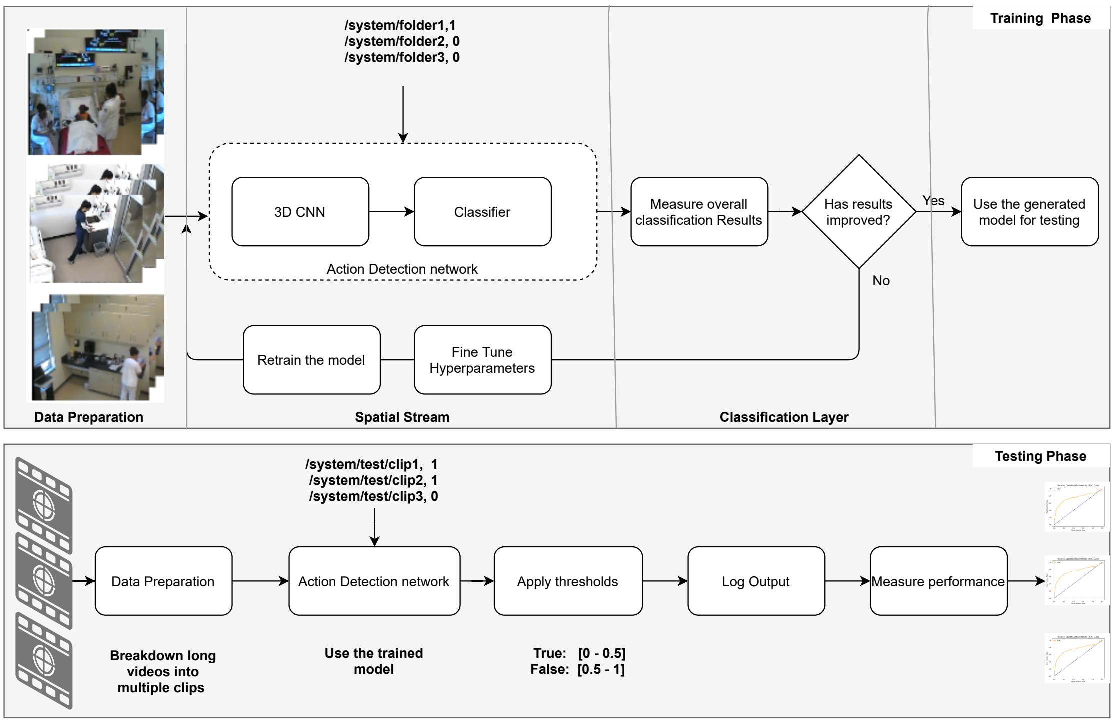


Figure 5.2: A visualization of training and testing pipeline

Fig 5.2 shows proposed training and testing approach in detail. It starts with a data preparation layer where input data and annotation format is created in such a way that it's easier to plug-in into an existing network model. In the next layer, an action detection network is fed with fixed length frames per clip to understand spatio-temporal features in the dataset. Finally, a fully connected layer with activation function classifies the clips into WashingHands or NotWashingHands class. A number of iterations with refining and fine tuning of hyper parameters is needed to reach the desired performance levels.

5.1.1 Training Setup

5.1.1.1 Data preparation

As a part of data preparation phase, video clips are converted into fixed length RGB frames and an annotation format is prepared. For most of the commonly used network models an annotation format contains path for input data and ground truth labels. Table[5.1, 5.2] shows annotation format needed for I3D and C3D action detection network models.

Annotation	Path	Label
/train/simcap01 0	/train/simcap01/	0
/train/simcap02 1	/train/simcap02/	1
/train/simcap03 0	/train/simcap03/	0

Table 5.1: Annotation format used for I3D

Annotation	Path	# of Frames	Label
/train/simcap01 1 0	/train/simcap01/	1-16	0
/train/simcap01 17 1	/train/simcap02/	17-32	1
/train/simcap01 33 0	/train/simcap03/	33-49	0

Table 5.2: Annotation format used for C3D

5.1.1.2 Spatial stream

The spatial stream consists of a Convolutional Neural Network(CNN), taking individual video frames as input. Spatial Stream could be able to learn a correlation between the spatial properties of health care providers, and which class of behavior is exhibited in a particular frame. For learning long term temporal information, different previous works have used a dual streams as input. It consists of a 3D-Convolutional Neural Network (3D-CNN), taking Optical Flow, generated from the videos, as input. By using Optical Flow as input, the Temporal Stream could learn motion features. Based on time limitations on this thesis, we could not complete our experiments using Optical flow.

5.1.1.3 Final Classification Layer

Since we only have two classes in our dataset, Washing hands and Not Washing hands, we will be using the same final classification layer for all the architectures presented in the remainder of the thesis. Since we use the same dataset for all the methods, we needed to change annotation formats and make initial adjustments in the data preparation phase. Our classification layer consists of a single output, softmax activated, fully connected layer. The softmax activation returns the probability our output to a range between 0 and 1 based on the given threshold, thus enabling us to represent our two classes as 0 and 1 for Not Washing Hands and Washing Hands respectively. When calculating the model accuracies, we round the outputs to the nearest integers by forcing every number < 0.50 to 0 and every number 0.50 to 1. Thus we have the following prediction intervals for our two classes:

1. **Not Washing Hands** = [0.0, 0.50]
2. **Washing Hands** = [0.50, 1.0]

Hyper-Parameters	**Description**
Learning Rate	Controls the rate at which the model learns by scaling the gradient’s effect on weight updates
Decay rate	Decays learning rate reduces the learning rate over time
Data Length	Number of frames per clip
Batch Size	Number of training samples to be fed into the network before model’s internal parameters are updated
Epochs	Number of times training dataset pass through the network

Table 5.3: List of common hyper-parameters for Activity detection network

5.1.1.4 Hyper-Parameter Tuning

The action detection network depends on different tunable hyper-parameters. In order to compare the results, all methods were initially set to use same hyper-parameter values. Idea behind hyper-parameters tuning is to find a value which should converge the network to a global minima and improve the speed of convergence. Table 5.3 shows a list of most common hyper parameters among all the methods.

5.1.2 Testing Setup

5.1.2.1 Data Preparation

Process of creating data for testing set differs slightly from training set. In case of testing set, about 100 unseen videos of varying length were collected and cropped into multiple 3 sec clips i.e. for a 5 minute long video 100 sub-clips were generated. All sub-clips were manually labelled using the Data Curation approach mentioned in Chapter 4. Main reasons behind such a data preparation method is two-folded:

- Easier to verify model correctness by looking at clip probabilities.
- Performing activity localization on sequentially arranged clips gets easier. Multiple clips can be combined by doing reverse engineering. In addition, activity localization timelines are produced using the resulted probabilities.

Annotation	**Path**	**Label**
/train/clip01 0	/train/clip01/	0
/train/clip02 1	/train/clip02/	1
/train/clip03 0	/train/clip03/	0

Table 5.4: Annotation format used for Testing set

5.1.2.2 Model Evaluation

Annotation format in table 5.4 is used as input to the best model generated as a result of model training. Testing process predicts per clip probabilities, which are then used to do a binary classification based on threshold values specified during the evaluation phase. We produces a confusion matrix based on results in Washing-hands and NotWashingHands classes and use performance metric which calculates sensitivity and specificity of the model.

5.2 Implementation Details

5.2.1 Computational platform

To evaluate the proposed action recognition system, we train and test the models on Tesla P100-PCIE-12GB GPU Server. The experiments were conducted using Python 3.5 that utilizes Tensorflow backend with Keras library and NVIDIA CUDA 9.0 library for parallel computation.

5.2.2 Source References

We referred to open source implementation of different renowned networks:

- **I3D** https://github.com/LossNAN/I3D-Tensorflow
- **C3D** https://github.com/lianggyu/C3D-Action-Recognition
- **Five Video Classification Methods**
  https://github.com/harvitronix/five-video-classification-methods

Developed and used following sources to be able to comprehend and implement the network models used in the remainder of this thesis:

- **Data Curation Tool**
  https://github.com/dhimanak/UiS-VideoActivityRecognition/tree/master/MediaClassifier
- **AWS Downloader and extraction Utility**
  https://github.com/dhimanak/UiS-VideoActivityRecognition/tree/master/AWSDownloaderUtility

5.3 Network Models

Several state of the art activity recognition methods were experimented and implemented as an attempt to solve the objectives of this thesis. We will try to emphasize more on two renowned and most commonly used methods namely I3D and C3D and come up with the best optimal network model which has shown better results on our dataset.

5.3.1 I3D Network Implementation

In order to train I3D network (as presented in section 2.5.3), input files with annotation format specified in section 5.1 needs to be formulated. As a starting point, videos clips are converted into RGB frames and sampled with different frame rates during each training experiment. the videos are resized preserving aspect ratio so that the smallest dimension is 256 pixels, with bilinear interpolation. Pixel values are then re-scaled between -1 and 1. During training, we randomly select a 224x224 image crop with 3 channels. I3D's Inception module is pre-trained on Kinetics[28] and Imagenet[36] datasets.

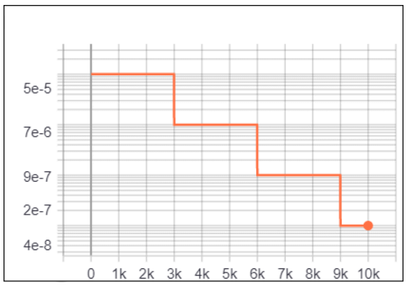


Figure 5.3: Learning rate

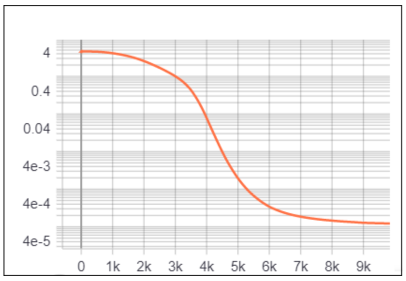


Figure 5.4: Model Loss

Figure 5.5: Fig representing learning rate and model loss plots during I3D network training

I3D network architecture presented in section 2.12 For RGB, the videos are resized preserving aspect ratio so that the smallest dimension is 256 pixels, with bilinear interpolation. Pixel values are then rescaled between -1 and 1. During training, we randomly select a 224x224 image crop with 3 channels. RGB I3d Inception module is pre-trained on Kinetics dataset and Imagenet datasets. Fig 5.5 shows I3D training plots.

5.3.2 C3D Network Implementation

In order to use C3D implementation, RGB frames from each clip were ordered sequentially and associated annotations were prepared in the format as shown in section 5.2.

C3D network resizes all video frames to 128×171 in a C3D Network. The input dimensions are 3×16×128×171. Random crops with a size of 3×16×112×112 of the input clips are used during training. The networks have 5 convolution layers and 5 pooling layers (each convolution layer is immediately followed by a pooling layer), 2 fully-connected layers and uses a softmax loss layer to predict action labels. The number of filters for 5 convolution layers from 1 to 5 are 64, 128, 256, 256, 256, respectively as shown in C3D network architecture here 2.11.

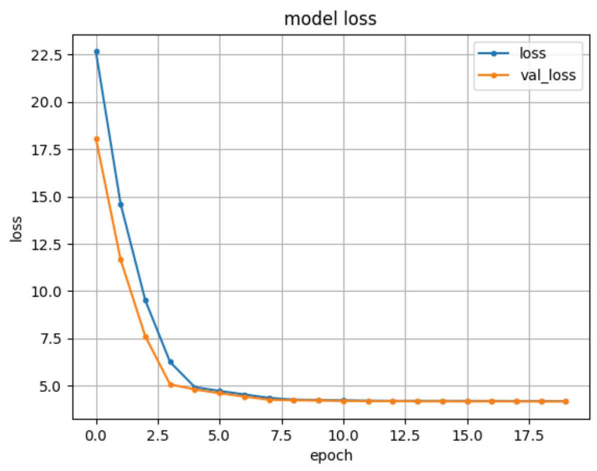


Figure 5.6: C3D model training

C3D network needs video input as sequences of frames or video files. In the case of video files (.mp4,.m4v .avi, .mov), machine needs to have codecs, opencv, and ffmpeg installed properly. In the case of using frames, C3D assumes that each video is a folder with frames which are numbered starting from 1 to N (number of frames). The frame names are formatted as "`video_folder/%06d.jpg`".

We have also generated I3D features in .npy format for further analysis and experiments.

Hyper-parameters used for C3D network are as shown in Table 5.3

Chapter 6

Experiments & Results

6.1 Preliminary Experiments

Section 2.5 presented several state of the art human activity recognition approaches which considered 3D convolution networks as a basis for their work. In order to baseline the results, different experiments were conducted to supplement our decision of exploring further on 3D CNN with multi layer perceptron approach. Therfore, following approaches were considered to baseline and develop our understanding on the dataset.

- Classify one frame at a time with a CNN
- Keras time-distributed CNN and passing the features to RNN
- Extract features with a CNN, pass the sequence to a separate RNN

First approach uses transfer learning by using pre-trained weights on inceptionV3 and ignores the temporal features as it attempts to classify each clip by looking at a single frame. Next, we trained the network on Keras time-distributed CNN for also capturing temporal information and passing these features to RNN in a single network layer. Finally, a 3D CNN is experimented with fewer convolution and pooling layers followed by LSTM layer. Following table shows training and validation accuracies achieved using all 3 methods.

Model	Train Accuracy	Validation Accuracy	Top-1
CNN - Single Frame	46.6%	46.2%	48.7%
Keras Time Distributed	49.2%	52.5%	51.5%
3D CNN	52.5%	53.2%	53.8%

Table 6.1: Table showing results of preliminary approaches

For single-frame & Keras time-distributed models, training & validation accuracy ranges between 45-51%. Keras time-distributed claims to learn temporal features however the network is too shallow to understand temporal features, hence the accuracy is not significant. On the other hand, 3D CNN with fewer layers increases the number of parameters significantly due to 3D kernels and pooling layers however it also fails to produce an acceptable accuracy but performs better than Single frame and Keras time distributed. This further strengthens our belief that a multi-layer perceptron(MLP) i.e. a deeper 3D CNN network should be experimented so that the model learn spatio-temporal features when shorter video clips are used as input. As explained in Chapter 5, We experimented on two different methods namely C3D & I3D to baseline our approach on RGB frames.

6.2 Experiment 1

This was more of a proof-of-concept experiment where annotation format on a smaller dataset were prepared and used on state-of-the-art I3D and C3D implementations. These implementations were adapted to work with our dataset. It required us to modify parts of code where logic for data preparation exists. No hyper-parameter tuning was performed in this experiment. This test just made sure that a complete code implementation cycle works end-to-end with raw data. No significant performance gains were recorded in this experiment.

6.3 Experiment 2

In this experiment, a total of 1000 video clips were divided into training and validation set where each 10 sec clips was downsampled to 90 frames. Following hyper-parameters were used while training the network.

Hyper-Parameters	C3D	I3D
Learning Rate	0.005	0.0001
Optimizer	SGD	ADAM
Max steps	12000	12000
Number of Frames per clip	90	90
Batch Size	16	16
Training Accuracy	**78.7 %**	**95.2%**
Testing Accuracy	**56.7 %**	**60.2%**

Table 6.2: Shows hyper-parameters and model accuracies

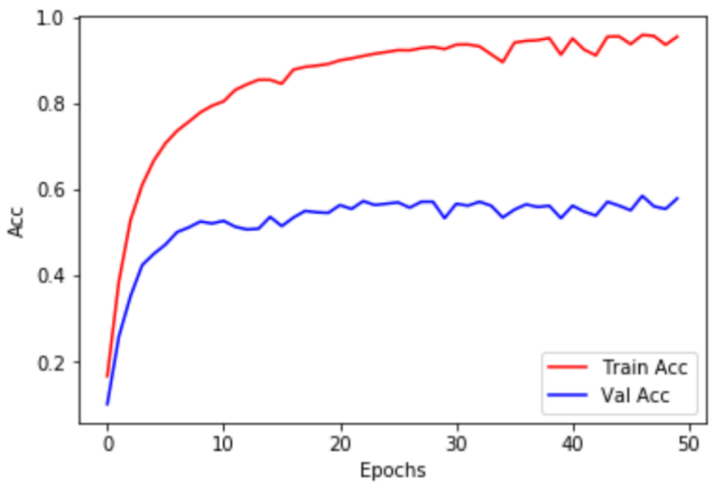


Figure 6.1: Shows training and validation accuracies which suffers from overfitting due to class imbalance

Fig 6.1 shows that training accuracy kept increasing while the validation accuracy hovers around 50%. This clearly indicates that we should have a balanced dataset with shorter clips. This experiment suffered from **overfitting** due to class imbalance and longer video clips(i.e. 90 frames).

6.4 Experiment 3

Based on observation from the previous experiment, significant changes were made to the input data and annotations. Table 6.3 depicts all hyper-parameter settings.

Following noticeable changes were made to the input data:

- Number of input video clips increased to 4000 by annotating and splitting existing 10 sec clips to 3 sec.

Hyper-Parameters	**C3D**	**I3D**
Learning Rate	0.005	0.005
Optimizer	SGD	SGD
Max steps	12000	12000
Number of Frames per clip	45	45
Batch Size	16	16
Training Accuracy	**71 %**	**73.2%**
Testing Accuracy	**61.7 %**	**64.2%**

Table 6.3: Shows hyper-parameters and model accuracies

- We record every other frame which reduces it to 45 frames per clip.
- Balanced data allocation into both the classes.

6.5 Experiment 4

This experiment attempts to fine-tune hyper-parameters in order to achieve better accuracy and produce an efficent model. Based on previous experiments, For I3D model, training and validation loss stops improving after 9000 steps. Test accuracy hovers around 65% with the current settings.

Hyper-Parameters	**C3D**	**I3D**
Learning Rate	0.001	0.0001
Optimizer	ADAM	ADAM
Max steps	10000	10000
Number of Frames per clip	45	45
Batch Size	16	16
Training Accuracy	**72.5 %**	**74.3%**
Testing Accuracy	**64.7 %**	**67.2%**

Table 6.4: Shows hyper-parameters used and model accuracy

Hence, we cut-down number of steps on I3D. I3D with SGD consumes less memory, however we don't see any significant improvements in training and validation accuracies. C3D on the other hands performs slightly better when learning rate is reduced and optimizer is changed from SGM to ADAM.

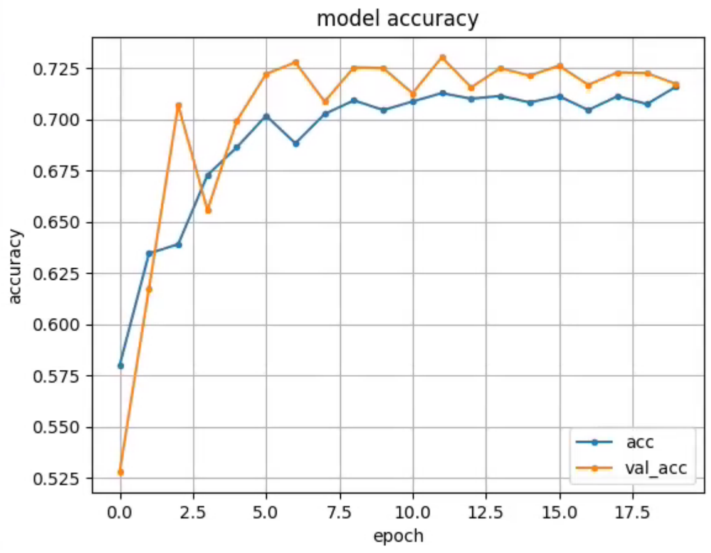


Figure 6.2: Shows training and validation accuracies with C3D approach

6.6 A closer look at results

Based on above experiments, it becomes evident that both C3D and I3D network models nicely adapt to the activity recognition problem. I3D being a deeper network tries to also learn temporal features better than C3D. Instances where washing hands activity is recorded over multiple frames across many clips, networks often tend to show a lag in activity localization(especially with C3D where a clip can only contain 16 frames) as shown in Fig 6.10. Also, reducing the number of frames per clip plays a vital role in achieving the accuracy. Following are the instances where networks models often show false negatives :

- Activity is not clearly visible for e.g. instances where static camera manages to only capture hands of health care providers.
- Camera angle obscures the washing hands activity completely.
- Video Quality and occlusion makes it difficult for the model to recognise the activity
- Multiple actors performing different activities
- Activity is visible only for just a fraction of second.

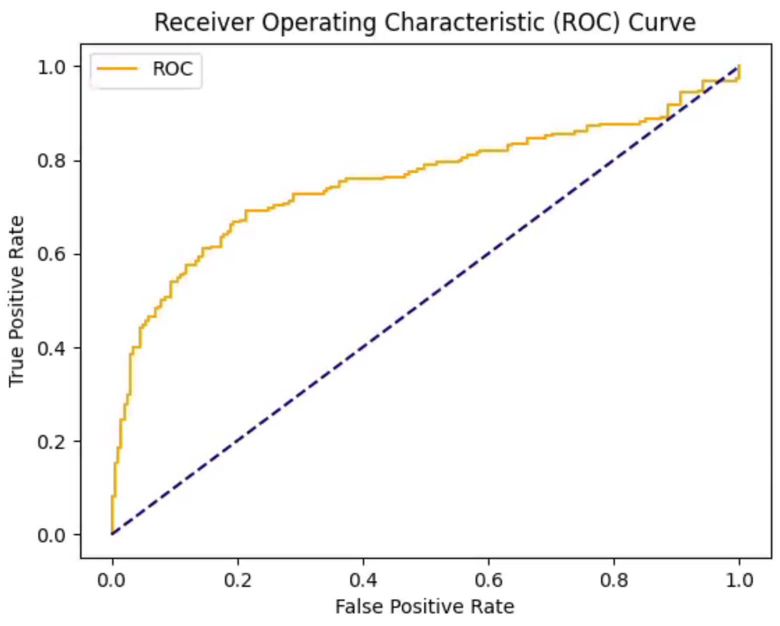
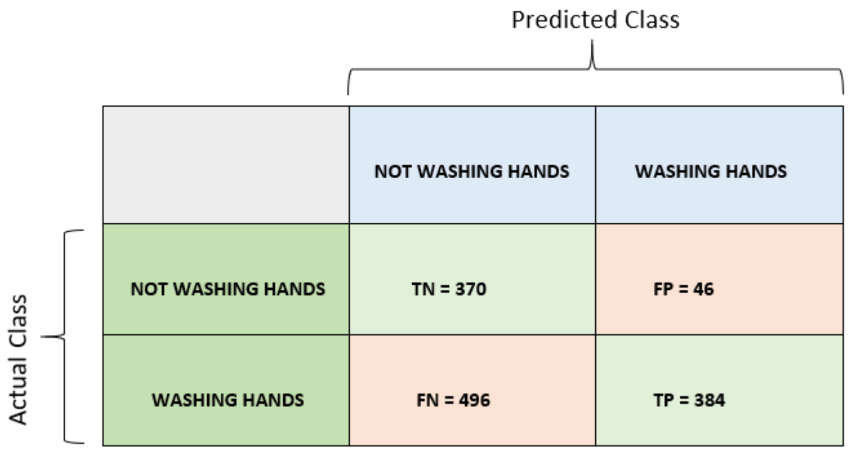


Figure 6.3: Visualization of ROC curve using I3D approach



	Predicted Class	
Actual Class	NOT WASHING HANDS	WASHING HANDS
NOT WASHING HANDS	TN = 370	FP = 46
WASHING HANDS	FN = 496	TP = 384

Figure 6.4: Visualization of Confusion Matrix

Fig 6.3 presents the ROC curve for the best model applied on the testing dataset. Best model gives an overall Test accuracy of **accuracy: 0.672, AUC:0.84**. Fig 6.4 shows the confusion matrix on test dataset with 1656 video clips.

Fig 6.5 shows a visualization where model wrongly classifies an action as washing hands activity. Fig 6.6, 6.7 shows instances where model correctly localized washing hands activity. Fig 6.10 shows activity localization on 4 different videos.

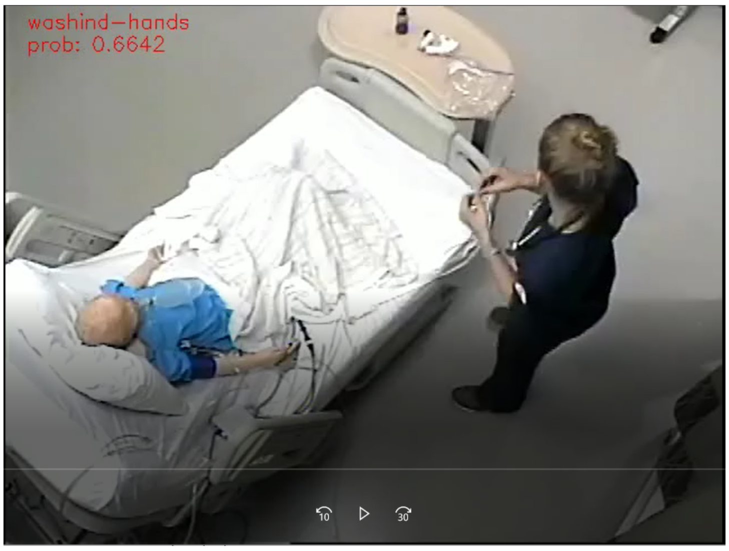


Figure 6.5: Shows an instance where model wrongly classifies an action as washing hands

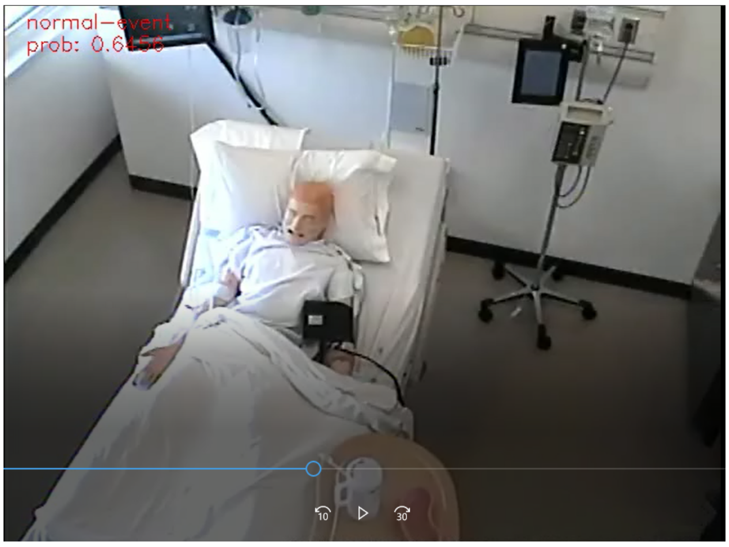


Figure 6.6: Shows an instance where model correctly classifies it as a normal event

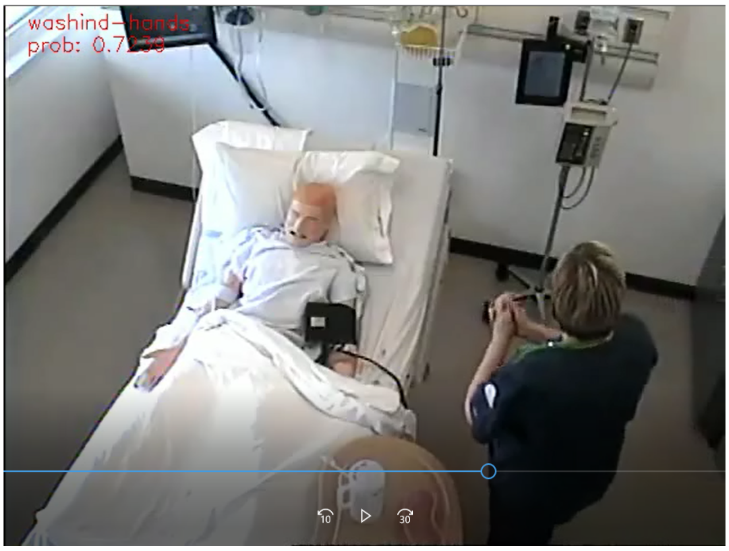


Figure 6.7: Shows that model correctly classifies the video clips as positive class i.e. Washing hands

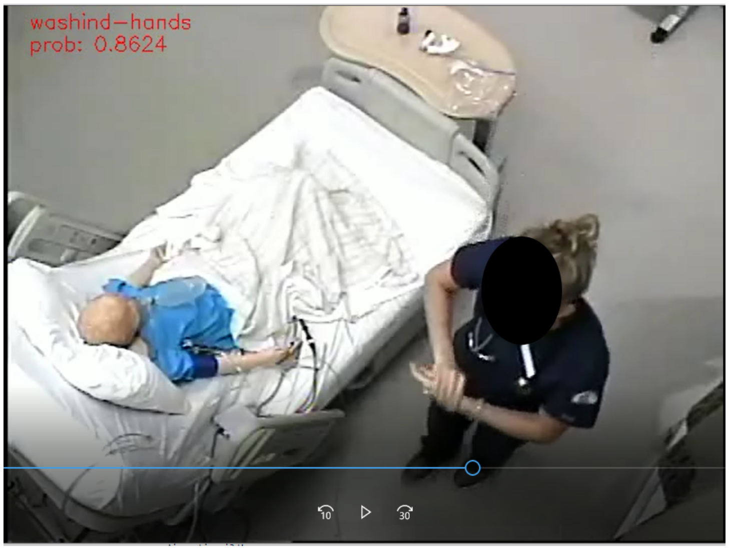

[redacted]

Figure 6.8: Shows that model correctly detects washing hands activity

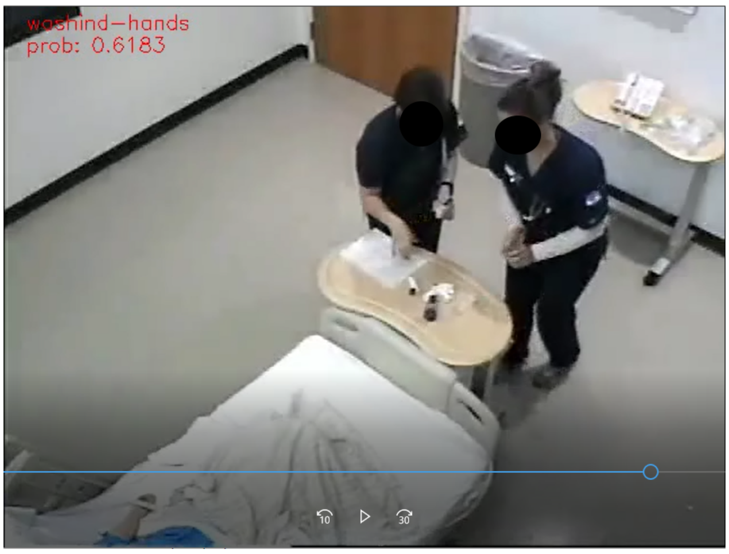


Figure 6.9: Shows that model correctly detects washing hands activity

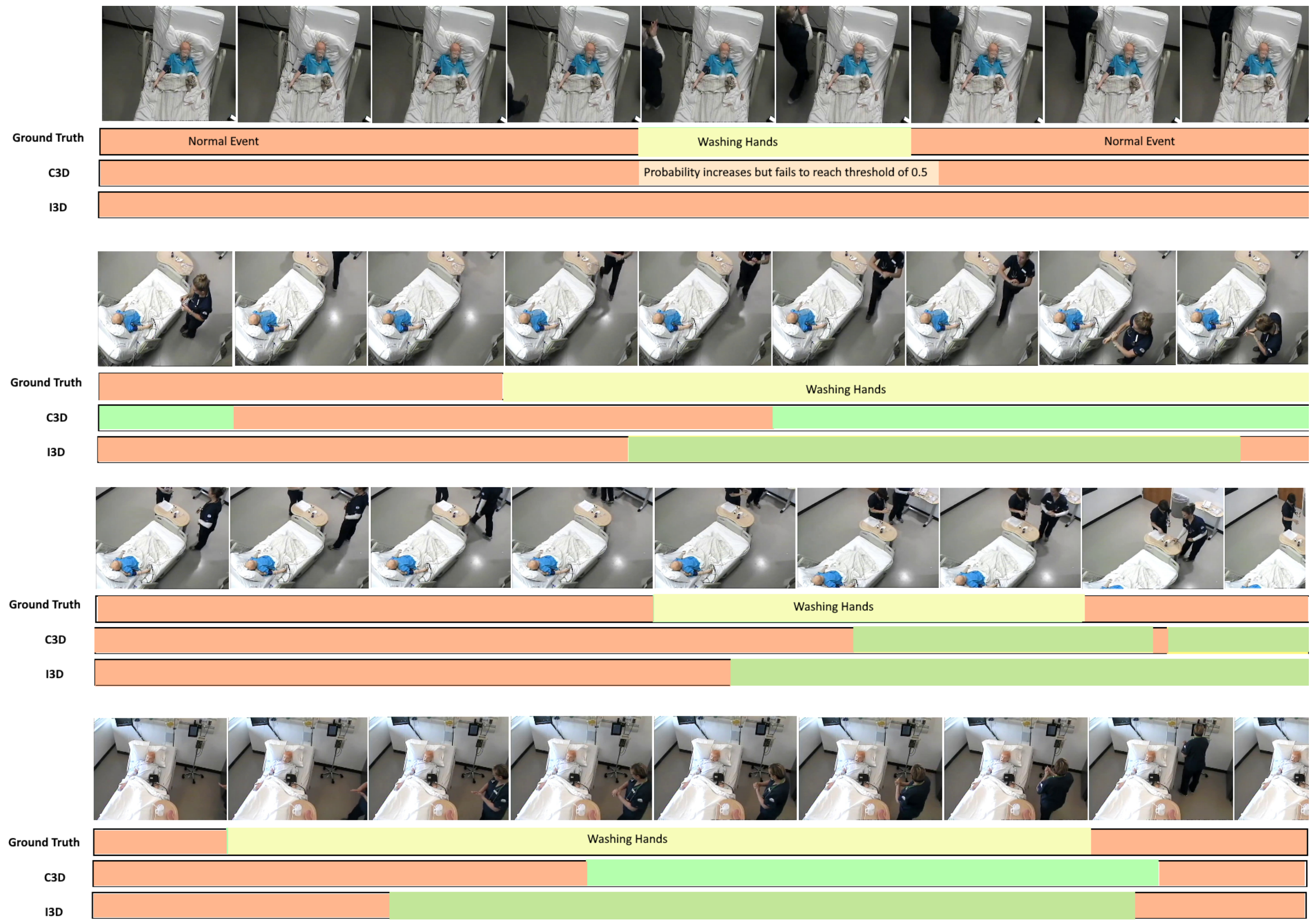


Figure 6.10: Shows Activity Localization on different videos and compares ground-truth with C3D and I3D models.

Chapter 7

Discussions

This chapter discusses the achieved results and addresses limitations within the experiments. The reference results are used as a baseline for comparison

7.1 Dataset Challenges

The dataset included varied length video recordings with inconsistent picture quality due to camera settings, environment settings, light conditions etc. Camera location, angles, perspectives and distance from the region of interest also play a vital role in determining the quality. It turned out that around 40% of the video data was discarded as it was not relevant for detecting washing hands activity due to various challenges discussed in chapter 3. Moreover, provided annotations were found to be unreliable and required further cropping and re-labelling. Analyzing the test results, it appears that motion blur, multiple objects, occlusion, activity occurring close to corners/edges of camera's aspect ratio were common factors for failed detections. Better camera angles and consistently high frame rates could limit, or even eliminate such issues.

Data Curation tool discussed in chapter 4 apparently solved the dataset challenges by filtering, cropping and annotating but it also required human intervention and supervision. This tool is good for quick data verification, creation of test datasets for proof of concept application but with deeper networks and ever-growing need of data, using such a tool can quickly become resource intensive and tedious. Therefore, an approach with weak supervision that can group similar objects by using cluster based classification must be tried.

7.2 Network configuration & Pipeline

Different methods were attempted in order to solve the activity recognition problem. C3D and I3D network models produced better baseline results compared to the preliminary basic network models like single frame classification & aggregation, time-distributed Keras with RNN. We tried to tune the network by adjusting hyper-parameters like learning rate, samping rate, number of steps, optimizers but training and validation accuracies did not improve. Many related works [4],[37] emphasized on using two stream i.e. a network model taking RGB frames and optical flow frames as input and fuse them toward the end of the network pipeline. This approach has shown state of the art results on UCF101 [19]- an action recognition data set having 101 action categories.

Different network models were tried so that we can compare the results and choose a baseline approach with optimal network configurations. For a fair comparison we tried to adopt similar network configurations as much as possible. However, there are differences in critical parameters for C3D & I3D networks such as input image size and number of frames per clip which can compromise the fairness of the comparison to some extent.

Experiments with shorter video clips greatly improved the performance. However, the network models used do not use any attention mechanism which means that it may require a lot of data for deeper video understanding. A better approach could be to use self-attention mechanism which can focus on a particular type of action for e.g. hands motion and body movement. Another approach would be to generate actor tubelets and use those as input to the network model[30].

7.3 Results

To better analyze the explained methods and the contributions of each one of them, the results obtained for mentioned datasets are compared in table 6.1 and table 6.4. For each method, the achieved accuracy values for the video datasets is also shown. In terms of network performance, we are basically looking for a model with a fewer false positives.

Compared with all the methods used in this thesis, C3D and I3D network models outperformed throughout all experiments. Top results were attained in Experiment 4 where the networks are trained with shorter video clips. Performance of both C3D and I3D models showed much better results compared to preliminary approaches discussed in table 6.1. Reason C3D works well is because it can create hierarchical representations of the spatio-temporal data but the main issue with C3D and in general 3D ConvNets is that it requires a lot of parameters because of 3D kernels which thereby also increase the dimensionality of the model. On the other hand, I3D uses pre-trained weights i.e. the implementation we adopted has been first trained on imagenet[36] and then on kinetics[28] dataset. Also, C3D is a shallow network model with 8 layers compared to I3D model which is a much deeper network, while having much fewer parameters because of pre-training. Hence, we propose I3D network model for further any improvements in this thesis.

In Experiment 1 & Experiment 2, all methods are trained on 10 sec long videos clips (down-sampled to 90 frames). This means that for C3D network training, we had to create 5-6 sub-clips per videos because of network model limitation of 16 frames per clip. In a fully supervised setting, generating correct annotations for all the sub-clips can quickly become tedious and resource intensive task. This was not the case with I3D model because of the flexibility and network architecture. Overall results with these experiments were not so significant but it triggered a need for shorter clips and further model tuning.

Experiment 4 shows that number of false positives have reduced manifold due to the fact that a balanced dataset with shorter clips and reliable annotations is fed into the networks. C3D reported network's training and validation accuracy around 72%

with a slightly lower test data accuracy of 65% as shown in table 6.4. Experiment 5 is an attempt to better the results by tuning the hyper-parameters.

We have a noticeable performance improvements with correct labelled annotations and shorter video clips.

Chapter 6 shows certain instances where model predicted false and true positives. Analyzing the results led to the conclusion that I3D seems to outperform other networks models and methods when applied on detecting and localizing **Washing-Hands** activity. However, model results have only been evaluated on RGB frames which can easily be extended to also include optical frames and self-attention mechanisms explained in various human activity recognition problems.

Chapter 8

Conclusion

The objective of this thesis was to recognize and localize WashingHands activity using Deep learning techniques. During the course of this thesis, we managed to experiment with different deep learning methods in order to baseline & stabilize our results. However, initial data analysis posed multiple challenges with the ground-truth labels which triggered a need for a Data Curation tool as explained in Chapter. In this thesis, a 20-25% of the time was spent on development of Data Curation tool and using the tool to create shorter video clips and annotate them correctly. Main motivation behind developing this tool was user friendliness, reusability and adaptability in any given setting.

After data preparation, several preliminary experiments were conducted on top of extracted C3D features such as LSTM, Keras time-distributed followed by plain 3D CNN. Among these preliminary methods, 3D CNN with few layers performed better than others i.e. training accuracy improved over time but resulted in too many false positives for e.g. instances where health care providers entrying and exiting the rooms, standing near the sink & talking with motion of hands were classified as true class. Hence, this triggered a need of trying a deeper network which can learn the spatio-temporal features on available video dataset.

Later, we decided to use existing state-of-the-art renowned network models namely C3D and I3D. In order to work with C3D and I3D which required us to organize the dataset and create specific annotation formats. Several experiments were conducted. In the earlier experiments, it became obvious that video clips of 10 sec length downsampled to 90 frames was a very broad experiment (i.e. network had too much to learn) which resulted in too many false positives. Hence, we needed to further crop our video dataset to 3 sec video clips and consider every other frame to make it 45 frames per clip. This experiment with fine-tuning hyper-parameters achieved better results i.e. fewer false positives and better AUC. I3D performed better in terms of testing accuracy on unseen videos. Overall accuracy attained by I3D network was 68%.

8.1 Future Work

Multiple Instance Learning (MIL) A key bottleneck in creating training data is that there is often an implicit assumption that it must be accurate. However, it is often infeasible to produce sufficient hand-labeled and accurate training data for most deep learning tasks. This is especially challenging for DL models that require a huge amount of training data. Also, generating labels manually is a tedious and time consuming task. We should instead experiment with a weakly supervised approach as explained below.
Multiple Instance Learning is an algorithm for supervised learning, where annotations are not provided for each instance, but only for bags of instances. The main assumptions in this regime are that a positive bag should contain at least one positive instance, while a negative bag should contain only negative instances. Generalized variants of MIL presume the presence of more than one positive instance per bag, for problems such as content-based image retrieval. General approach and implementation has been nicely put up [38]

Attention based networks Based on the experiments and results, we see that model accuracy stops to improve after a certain point. A possible improvement to this work would be to add self-attention or using attention based networks for e.g. Dai et al. [37]

Optical flow Due to time constraint on this work, current network models were only trained on RGB frames. This can be easily extended to also include strong temporal correlation by adding optical flow data. Many different networks and related works

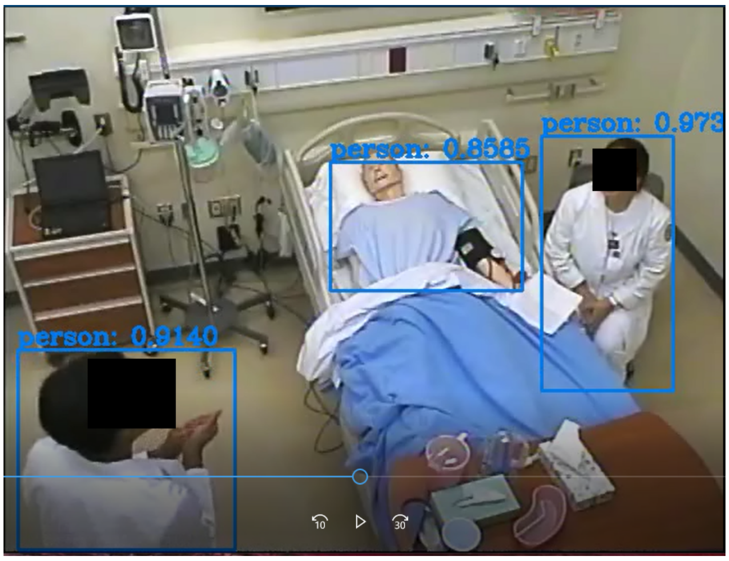


Figure 8.1: Frame 1

have shown better results when optical flow data is also included to make it a two-stream network model.

Tubelet proposal network In an attempt to generate proposals from videos, we used pre-trained YOLO v3 weights to detect health care providers but due to time constraint this was not materialised. However, the idea is to detect actors with their bounding boxes as show in fig 8.1 and create actor tubelets by actors across multiple frames and train the networks.

With Tubelet proposal network approach, rather than sending whole RGB frame, multiple tubelets can be served as input to the network thus reducing the background noise and allows network to focus only on details. Similar work was done here [39]

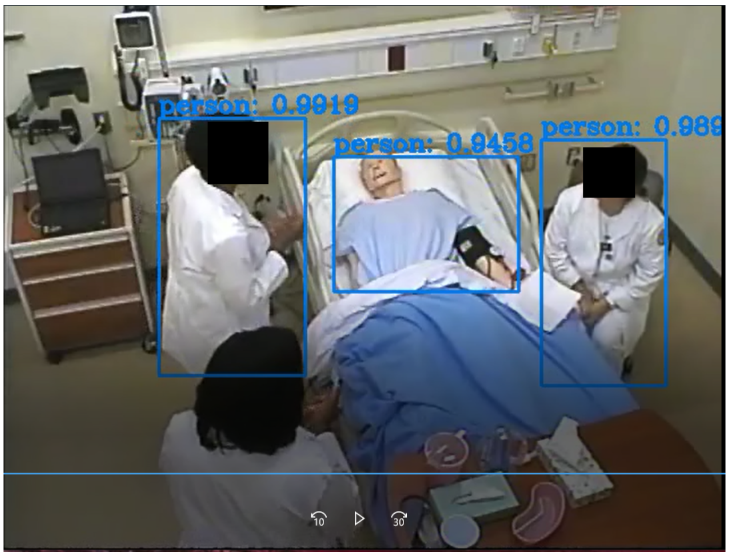


Figure 8.2: Frame 2

Bibliography

[1] Chris A Mack. *Deep Learning - Josh Patterson Adam Gibson*. 2009.

[2] Computer Science Wiki. Max-pooling / pooling — computer science wiki,, 2018. URL https://computersciencewiki.org/index.php?title=Max-pooling_/_Pooling&oldid=7839. [Online; accessed 13-June-2021].

[3] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks, 2015.

[4] Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. *Advances in Neural Information Processing Systems*, 1, 06 2014.

[5] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005.

[6] Xiaoyuan Zhu, Zhiyong Yang, and Joe Tsien. Action recognition using natural action structures. *BMC Neuroscience*, 13, 07 2012. doi: 10.1186/1471-2202-13-S1-P18.

[7] pewresearch. https://www.pewresearch.org/internet/2013/10/28/photo-and-video-sharing-grow-online/.

[8] Ales Leonardis and Horst Bischof, editors. *Proceedings of the ECCV04 Workshop on Statistical Learning in Computer Vision*. ., 2004.

[9] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein,

Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vision*, 115(3):211–252, December 2015. ISSN 0920-5691. doi: 10.1007/s11263-015-0816-y. URL https://doi.org/10.1007/s11263-015-0816-y.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, May 2017. ISSN 0001-0782. doi: 10.1145/3065386. URL https://doi.org/10.1145/3065386.

[12] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[13] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(76):2493–2537, 2011. URL http://jmlr.org/papers/v12/collobert11a.html.

[14] Ibrahim Abaker Targio Hashem, Ibrar Yaqoob, Nor Badrul Anuar, Salimah Mokhtar, Abdullah Gani, and Samee Ullah Khan. The rise of "big data" on cloud computing: Review and open research issues. *Information Systems*, 47:98–115, 2015. ISSN 0306-4379. doi: https://doi.org/10.1016/j.is.2014.07.006. URL https://www.sciencedirect.com/science/article/pii/S0306437914001288.

[15] Jerome Friedman Trevor Hastie, Robert Tibshirani. *The Elements of Statistical Learning*. 2017.

[16] Claudio Gentile and Manfred K. Warmuth. Linear hinge loss and average margin. In *Proceedings of the 1998 Conference on Advances in Neural Information Processing Systems II*, page 225–231, Cambridge, MA, USA, 1999. MIT Press. ISBN 0262112450.

[17] Reuven Y Rubinstein and Dirk P Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer Science & Business Media, 2013.

[18] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR. URL http://proceedings.mlr.press/v37/ioffe15.html.

[19] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *CoRR*, abs/1212.0402, 2012. URL http://arxiv.org/abs/1212.0402.

[20] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[21] Aaron Bobick and J.W. Davis. The recognition of human movement using temporal templates. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23:257 – 267, 04 2001. doi: 10.1109/34.910878.

[22] Christian Schüldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: A local svm approach. volume 3, pages 32 – 36 Vol.3, 09 2004. ISBN 0-7695-2128-2. doi: 10.1109/ICPR.2004.1334462.

[23] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2013.

[24] Nazli Ikizler, R Gokberk Cinbis, Selen Pehlivan, and Pinar Duygulu. Recognizing actions from still images. In *2008 19th International Conference on Pattern Recognition*, pages 1–4. IEEE, 2008.

[25] Gül Varol, Ivan Laptev, and Cordelia Schmid. Long-term temporal convolutions for action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(6):1510–1517, 2018. doi: 10.1109/TPAMI.2017.2712608.

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

[27] Deepmind. https://deepmind.com, .

[28] Deepmind. https://deepmind.com/research/open-source/kinetics, .

[29] Deepmind. https://github.com/deepmind/kinetics-i3d, .

[30] Mihir Jain, Jan Van Gemert, Hervé Jégou, Patrick Bouthemy, and Cees GM Snoek. Action localization with tubelets from motion. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 740–747, 2014.

[31] Du Tran, Jamie Ray, Zheng Shou, Shih-Fu Chang, and Manohar Paluri. Convnet architecture search for spatiotemporal feature learning, 2017.

[32] Rui Hou, Chen Chen, and Mubarak Shah. Tube convolutional neural network (t-cnn) for action detection in videos. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[33] Saravanan Thirumuruganathan, Nan Tang, Mourad Ouzzani, and AnHai Doan. Data curation with deep learning [vision], 2019.

[34] TensorFlow: Large-scale machine learning on heterogeneous systems. URL https://www.tensorflow.org/. Software available from tensorflow.org.

[35] Francois Chollet et al. Keras, 2015. URL https://github.com/fchollet/keras.

[36] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[37] Cheng Dai, Xingang Liu, and Jinfeng Lai. Human action recognition using two-stream attention based lstm networks. *Applied Soft Computing*, 86:105820, 2020. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2019.105820. URL https://www.sciencedirect.com/science/article/pii/S1568494619306015.

[38] Waqas Sultani, Chen Chen, and Mubarak Shah. Real-world anomaly detection in surveillance videos. *CoRR*, abs/1801.04264, 2018. URL http://arxiv.org/abs/1801.04264.

[39] Kai Kang, Hongsheng Li, Tong Xiao, Wanli Ouyang, Junjie Yan, Xihui Liu, and Xiaogang Wang. Object detection in videos with tubelet proposal networks. In *CVPR*, 2017.