university of groningen / faculty of science and engineering

UNIVERSITY OF GRONINGEN,
THE NETHERLANDS

MASTER'S THESIS
ARTIFICIAL INTELLIGENCE

# Learning Deep Spatio-Temporal Features for Human Activity Classification

*Subilal Vattimunda Purayil*
*s3587630*

December 5, 2020

supervised by:
Dr. Hamidreza Kasaei, Artificial Intelligence, University of Groningen
Dr. Marco Wiering, Artificial Intelligence, University of Groningen

# Contents

# List of Figures

# List of Tables

# Abstract

The last decades have seen a growing interest and demand for Human Action Recognition (HAR) systems. HAR is widely used, and its presence can be seen in the application areas ranging from Human-Computer interaction (HCI), robotic interactions to video surveillance, autonomous driving, home-based rehabilitation, etc. While offering attractive opportunities for modern systems development, existing HAR systems often fail to recognize human actions precisely.

In this thesis, we propose a new model for HAR using deep learning technologies. Our model encompasses the *3D Convolutional Neural Network*, which learns the local spatio-temporal features from video input; and a *bidirectional Long Short-Term Memory*, which learns the long term temporal dependency from the spatio-temporal features to classify the actions. We perform a comparative study of the proposed model with six other baseline models implemented using deep learning methods with different modalities. Extensive experiments demonstrate that our proposed model outperforms baseline models and other state-of-the-art approaches in terms of accuracy. We believe that our work is a step closer to developing accurate real-world HAR applications.

# Acknowledgements

I would like to sincerely thank my supervisor, Dr. Hamidreza Kasaei, for providing me with full freedom and time to explore and pursue my own ideas. Heartfelt thanks to Dr. Marco Wiering for supervising my thesis and for providing constructive comments. I appreciate the time and effort they took to proof-read this document. I firmly believe that without the support of both my supervisors, this work would not have been successful.

Finally, I would like to thank my dear friend, Sowmya Ravidas, for assisting in the draft and review of this document. This was a hard task because of the topics she never faced before, and hopefully, I believe she learned a few things from this topic.

# 1 Introduction

There has been a growing interest in Artificial Intelligence research, and a large amount of these works have been devoted to developing an efficient mechanism for human-robot collaboration. Such mechanisms aim to facilitate a method of interaction between humans and robots. This is primarily achieved by enabling robots to recognize human actions and providing a corresponding response through counteractions. Typically, these actions are identified from a stream of visual data or a recorded video. However, in order to establish a smooth interaction between humans and robots, it is crucial to recognize human actions without any error. Human Action Recognition is the task of identifying an action from a video that depicts an action performed, using technologies like Computer Vision (CV), Machine Learning (ML), Deep Learning (DL).

We know that actions performed by human beings can be very similar to one another. For instance, running, jogging, and walking are very similar due to the same pose that is depicted while performing such actions. Moreover, the same actions performed by different individuals can be different from one another. For example, videos that are recorded outdoors can contain several entities performing various activities. These factors make it complicated for a robot to recognize individual action accurately. Furthermore, some actions can be instantly recognized from the video stream, whereas others require the processing of redundant data before those actions can be identified. This means that each video frame needs to be processed differently [3]. This is because various actions performed in the videos may have different motion speed, viewing angle, pose, and appearance. Successful representation of pixels from the video can help draw a boundary between different action categories, which again is an arduous task. Accurate recognition of action from videos, i.e., by considering all of the variations mentioned above, is still an open problem [3].

Traditional action recognition models use handcrafted methods to extract features from videos; and employ classifiers such as SVM [4], decision trees [5], KNN [6], etc., to classify the actions from those features. However, such traditional models often fail to study effective action representations from large datasets due to the non-automated user based feature learning [7]. These models use very deep and complex networks that induce a very high computational complexity and are carefully designed for specific datasets, which requires domain expertise. Hence, these models are ad hoc and cannot be generalized for different dataset scenarios [7].

With the recent applications of deep learning in several fields such as text/image classification, natural language processing, and handwriting/audio recognition has not only provided novel solutions for previously unsolved tasks but also paved the path for new research directions in the field of activity recognition. Deep learning has enabled way to remarkably robust and potent generic solutions for many problems.

Deep learning methods learn the spatio-temporal features automatically using layer-wise operations. The spatio-temporal features are outputs of deep learning networks that represent information from both spatial and temporal domains extracted from

videos. In comparison to the traditional approaches, deep learning methods are not only generic but are less complex and have a relatively lower computational cost. Deep learning methods can perform both feature representation and classification using the same network, which makes this a simpler approach than traditional models.

There are ongoing works that focus on extracting actions from video streams using deep learning algorithms. There have been a few successful attempts to classify the action using the information processed from spatio-temporal features. A two-stream CNN is one such successful approach that incorporates two different neural networks using layer-wise fusion, one handles information from the still images, and the next one stacks the motion flow. The spatial stream exploits the appearance data of each frame of the action video, and the temporal stream captures motion information from optic flow [8, 9, 10]. Another approach for extracting video information is the deep-wide network, which recognizes actions with the help of the details extracted from spatial-temporal features of skeletons sequence in actions [11].

All the above deep learning based approaches showcase only a mediocre accuracy in the range of 70% - 90%. In this thesis, we aim to extract the action representation features from video and train a classifier to perform action classification from these representations with higher accuracy than the existing models. A better accuracy for any action recognition model directly correlates to the better spatio-temporal features used for the classification of actions, because the classification of actions relies primarily on these features. In this thesis work, we implement a deep learning hybrid model, which is a combination of two deep learning algorithms; 3D Convolutional Neural Networks [12] and bidirectional Long Short-Term Memory [13]. Our model extracts the spatio-temporal features from videos, and these features are used to classify the actions accurately. Our model is based on deep learning algorithms, such as Convolutional Neural Networks (CNN) [14], Recurrent Neural Networks (RNN) [15], and Long Short-Term Memory (LSTM) [16, 15] for extraction of features and classification of actions.

## 1.1 Research Question

This thesis aims to develop a model that can effectively recognize human actions from a video stream. Our model is based on two deep learning algorithms: Convolutional Neural Networks and Bidirectional Long Short-Term Memory (Bi-LSTM). One of the main challenges faced by the existing human action recognition models is the erroneous and incomplete identification of complex spatio-temporal features inscribed in the data. In this thesis, we obtain these spatio-temporal features with at most efficiency and accuracy. To this end, we define the following research questions:

1. How can we develop a hybrid model that combines 3D-CNN with Bi-LSTM for action recognition?

2. Can the spatio-temporal representation learned by the proposed model perform accurate and efficient action classification?

3. How can we experimentally evaluate the performance of the proposed model?
4. How can we analyse the performance of the proposed model in comparison to other state-of-the-art models?

## 1.2 Outline

The rest of this thesis is structured as follows. In Chapter 2 we describe the necessary background information on existing action recognition frameworks and deep learning algorithms. The methods and experiments are discussed in Chapter 3, and the results of the experiments are presented in Chapter 4. In Chapter 5, we provide answers to the research questions, and finally subsection 5.2 concludes this thesis and draws possible directions for future works.

# 2 Theoretical Background

This chapter provides an overview of the spatio-temporal features and the current methods that use these features to classify human actions. Furthermore, we discuss the deep learning algorithms that are commonly used to implement classifiers.

## 2.1 Spatio-temporal Features

Spatial information encompasses various objects along with their positions and background details. In particular, the information consists of graphic details such as points, lines, polygons, pixels, etc., along with their geographic attributes such as location, size, shape, orientation. Furthermore, spatial information will include the relationship between various spatial objects. On the other hand, the temporal information is usually obtained from multiple frames by extracting all the relevant information that is varying according to time. The temporal context captures information of objects or pixels which are moving in consecutive frames. Any video can be split into spatial and temporal information, and identification of action from any video requires the use of both this spatial and temporal information. The motion information of various objects in a video is captured in temporal information, which can lead to easy and accurate classification of actions from it.

The basic properties of an efficient spatio-temporal feature include simple to perform related operations, generic across various kinds of video types, compact for flexible processing, and efficient to represent all the details. The methods for extraction, processing, and utilization of the spatio-temporal features should be **simple** in terms of memory utilization, computational complexity, and computational time. If the spatio-temporal features are not simple in operations related to them, it would add to the complexity of models using them, which are generally expensive in computation due to the use of complex deep learning algorithms used to make intelligent decisions. Modern-day video mechanisms have made great improvements in the type and capabilities; and in order to represent all of these kinds of videos, it is necessary that the spatio-temporal features should be **generic** across all video types. Moreover, the generic feature representations should also be discriminating in operations and efficient on operations with all kinds of video types. The features should also be **compact**, which helps for efficient processing, storing, and retrieval of feature operations. Lastly, the features must be **efficient** enough to fit for various applications like action classification, gesture recognition, etc. The efficiency includes the extraction and processing of the features in the best way with minimum computations to befitting to perform the classification task accurately [17].

The spatio-temporal feature becomes a conclusive factor to various applications, which requires information about the geography of objects along with their time-varying dependence to perform a task. Action recognition, video classification, emotion recognition, and gesture recognition use spatial-temporal features to perform predictions from videos.

## 2.2 Action Recognition and Applications

The action classification, also known as action recognition, can be defined as the accurate identification of actions performed by humans in a video. The action recognition has been used in a large range of day-to-day scientific and security-based applications, thereby making it an interesting area of research for people working in computer vision, machine learning, and deep learning [18]. The surveillance systems powered with action recognition can help in automated detection of humans performing illegal actions in a restricted zone [3]. The human-robot interaction-based applications, such as, personal robots also require action recognition to understand the nature of human actions as a communication medium. The modern entertainment systems for gaming and stimulation take human actions as input, and identify the anticipated activity such as dance fights, etc., from them to provide the best entertainment experience [19, 20]. Automated driving cars demand proper monitoring of driver actions in order to provide directions for a safe journey [21]. Video retrieval models that search specific actions in a large library of videos also make use of action recognition methods [22, 23].

### 2.2.1 Challenges in Action Recognition

Although a number of works have been proposed for action recognition task, there are still several open problems. Firstly, action recognition is a complex task that operates on video data and performs classification with videos using deep learning or machine learning models, which often have much room for misclassification of videos, thereby causing an error in the recognized action. Secondly, the videos recorded from different angles, such as behind, sides, or in front of the objects, will appear different. There will be a difference in the appearance of actions recorded from different viewpoints; therefore, the classification of such actions requires a classifier model with intelligence. Finally, the actions happening outdoor can possess more background noise than videos recorded indoors, which adds to the complexity of the classification of actions [24]. Action such as running or walking can be performed at different speeds and variations that can cause misclassification easily. Action dataset can also posses inter-class similarity, which is the similarity between actions from different classes in appearance, which could create confusion between those classes for classification [18]. Furthermore, an enormous amount of recognized human actions cannot be fit for any dataset. The dataset should incorporate intra-class and inter-class variations of action classes, which can be used to train any classifier for effective classification of human actions. However, the dataset available for action classification is limited in terms of availability and number of action classes in available datasets. The non-availability of a fully-fledged action recognition dataset is also a major challenge [3]. All video frames in the video may not be predictive. For example, some actions can be determined from initial frames alone, whereas others require long frame width. Thus, there is a challenge about the length of frames of videos to be used for the classification. The optimum process is to select the required number of frames which can efficiently perform the classification. Any model that analyses the entire set of video frames or more frames than optimum

frames width for classification will end in a computationally expensive model. The model classifier cannot converge to the precise class if we use a lesser frame than the optimum frame number. A classifier's training for an action classification will require learning action classes from videos, which is computationally expensive and may even take several days for completion [25].

### 2.2.2 Action Recognition Framework

The action recognition frameworks consist of models or pipelines taking an input video, processing the video, and predicting the action from that video. The framework generally comprises two components, which are action representation and a classifier. The action representation performs the conversion of the input video to a feature map, which consists of all the necessary information required to identify action in that video. The classifier performs the exact identification of action from features generated from the input video. Modern deep learning-based approaches incorporate both action representation and classification with one single model. In this research thesis, we implement and experiment models with deep learning frameworks for action classification.

**Action Representation** The action recognition will require to operate on input videos, which may vary from one another in terms of background, motion speed, camera angle, and pose variation. The method commonly followed is action representation, which is the extraction of the information from these videos that can distinguish human action. It is challenging to obtain a representation that can efficiently perform action classification from different kinds of videos. The action representations should be easy to compute, increase the inter-class discrepancy, remove background noise, minimize intra-class variations, and improve classification performance. The action representation is broadly classified into handcrafted representations and deep learning-based representations. Again, The handcrafted representations are further divided into two main kinds holistic representation and local representation.

**Holistic Representation** This method uses the shapes and silhouette of the human body to form features that capture all the motion information from videos. The feature extracts all the motion information of the entire human subject from the video that can provide a clue for the action occurring in the video. This method primarily masks a rectangular area around the human body, performing an action and continuously capturing pixel changes to this area [3]. The rectangle area used to capture motion happens to add a lot of background information, redundant data, and noise. The Motion Energy Image (MEI) is one popular holistic representation method that extracts features from the area of the human body with the motion dynamics in them. The features include spatial distribution of the area where the action occurs, and the motion dynamics are captured with the change in brightness [26]. The Motion History Image (MHI) is most similar to MEI, and the difference is that instead of using brightness, this method uses pixel intensity to capture motion dynamics in more detail. The Motion History-

Volume (MHV) method uses 3-dimensional silhouettes to capture the human shape and motion dynamics [27]. This method removes viewpoint problems in two-dimensional features. The computer vision-based optical flow algorithms can also extract motion information efficiently. The optical flow captures the pixel intensity between frames where displacement occurs. Lastly, the disadvantage of holistic representation methods is that they require accurate segmentation of shapes in the video, which can be a source of low accuracy [28, 29].

**Local Representation** The local representation extracts features from the local region of interest where motion information is predominant. The local features do not include information about the human body, and localization of the human body is not required. The region of interest for action recognition in videos is detected using space-time points and motion trajectory methods. The local features include the shape and motion information of the area of interest and its boundary area information [3, 22]. The Spatio-Temporal Interest Point Detector (STIP) is the filter or detectors used to find the salient areas of spatial and temporal information in videos. Implementation of Harris3D interest point detector, which is the enhancement of the Harris detector in spatio-temporal dimension, is one of the primary work in the area of STIP [30, 31, 32]. The Gabor detector gives a better result than the Harris3D interest point detector and employs a Gaussian spatial detector with Gabor temporal filter [33]. Various other methods of spatio-temporal interest point detector are Hessian filter method, 3D SIFT, HOG 3D, local trinary pattern, etc [34, 35, 36, 37, 38]. Another set of methods include the optical flow-related mechanisms to identify the area of interest, which are histogram of optical flow (HOF) and motion boundary histograms (MBH) [39, 34].

STIP methods perform better for short context videos, and they are not suitable for retaining information for a long duration. Motion Trajectories can be another effective local representation method to find action representations for long context information analysis and capture. These methods obtain the points of interest using a detector and follow these points in consecutive frames to mark their movement [3, 22]. The Kanade-Lucas-Tomasi (KLT) detector is one of the popular motion trajectory-based methods which extracts the action features using eigenvectors of $2 \times 2$ window in video [40, 41, 42].

**Deep Learning Based Representation** The handcrafted feature-based action representation requires domain expertise for proper selection and design of methods. Another method very popular these days is the deep learning-based representations, which are powerful and easy to generalize. The Deep Learning-Based Representation model learns to generate representation all by itself. The deep learning methods used are convolutional neural networks and recurrent neural networks. The convolutional neural network can be used to capture either of the spatial and temporal information. A fused two-stream CNN can be used to learn spatio-temporal information from videos [9]. The 3D CNN, which is an extension of CNN, is also very effective to learn spatio-temporal features from videos [43]. All the methods used in this research thesis

adopts deep learning-based action representation.

**Action Classifier** The classifier performs the final task of prediction of action class in the framework using the action representation of videos. The classifier takes the features as input and tries to predict the action class from the input. The selection of a classifier depends on the complexity of the task, type of data, and dataset size. The classifier can be based on machine learning algorithms such as Decision tree, Support Vector Machines, KNN, Artificial Neural Network, Convolutional Neural Networks, etc.

## 2.3 Neural Networks

A neural network is a parallel and distributed computational model inspired by neurons from the human brain. The elementary computational processing unit in every neural network is a neuron. The basic neuron comprises a set of inputs, weights corresponding to each input, activation function, and the output, as shown in Figure 1 [44]. The product of weights with the inputs is summed and passed over an activation function to obtain the neuron's output. The activation function decides whether the information is beneficial to pass on to the next neurons for further processing. This neuron is technically a perceptron [45] which is interconnected with many other such perceptions to form a network.

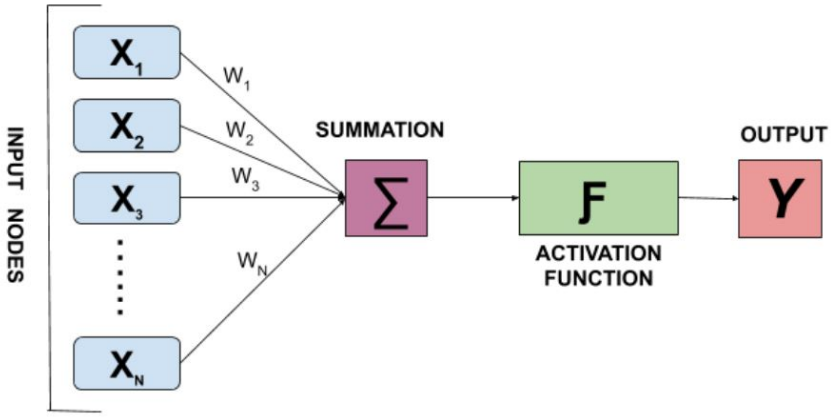


Figure 1: The basic structure of a neuron on neural networks.

These neural networks are very efficient in processing data such as images, videos, speech, language corpus, etc. There are many variations of neural networks depending upon the type of data and application of these data. The popular models of neural networks includes Feedforward Neural Network, Convolutional Neural Networks [46], Recurrent Neural Networks [47], Long Short-Term Memory [48], Auto Encoders [49, 50], Radial Basis Function Neural Network [51], etc. The applications in which neural networks are deployed successfully include simple classification, image classification,

face recognition, speech recognition, computer vision, machine translation, Sentiment Analysis, predictions, video recognition, etc.

### 2.3.1 Activation Functions

The activation function is analogous to the action potential in the human brain's biological neurons, which decides whether information should be passed on to the next neurons. The activation functions are mathematical functions whose input is the sum of the product of weights of the neuron and data input to the neuron and determines the output of the neuron. The activation function is a simple gate that can OFF and ON a particular neuron in the network. The different kinds of activation function used in neural networks are Step, Linear, Sigmoid, Tanh, ReLU, Leaky ReLU, Exponential Linear Unit, Swish, and Softmax.

## Feedforward Neural network

The Feedforward Neural networks are a neural network where data flows in a unidirectional path from input to output. Each layer is a set of nodes that share the same depth. The input layer is the entry connections to a neural network where the input data to be processed are given. Similarly, output nodes are the exit gate of a network that provides output to the world. The hidden layers are layers between the input layer and the output layer that do not interfere with the outside world. The depth of the hidden layer depends upon the computational requirement of the network. An example of such a network is shown in Figure 2.

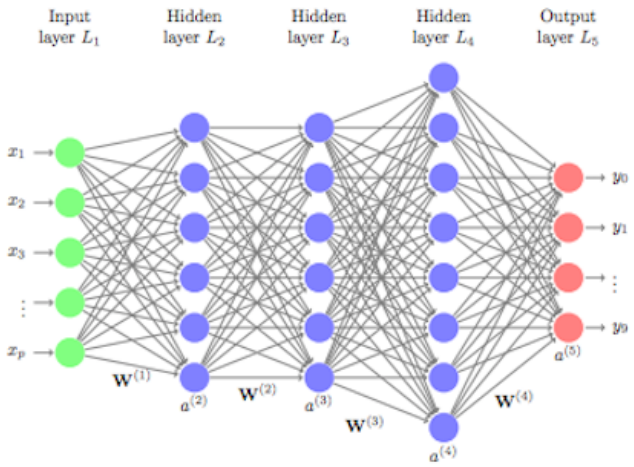


Figure 2: The basic structure feedforward neural networks with of the fully connected layer with complete connections between them [1].

[1] http://uc-r.github.io/feedforward_DNN

## 2.4 Convolutional Neural Networks

The Convolutional Neural Networks are also known as CNN, was introduced first by Yann LeCun [14]. The Convolutional Neural Network is an advanced feedforward neural network specializing in grid-like or pixel data. The heart of the CNN is convolution operation. The CNN can be analogous to a multi-layer feedforward neural network that employs convolution operation instead of general matrix multiplication.

CNN's are heavily deployed in image classification, face recognition, object recognition, image recognition, and video recognition. The general structure of CNN is shown in Figure 3. The input to CNN is general pixel data such as two-dimensional and three-dimensional images or grid structured data like time series data. The input is passed through a set of a convolutional layer, activation layer, and pooling layer to extract information from images. Finally, this information is passed through a flatten layer to make it a one-dimensional array, and a softmax layer is used for classification.

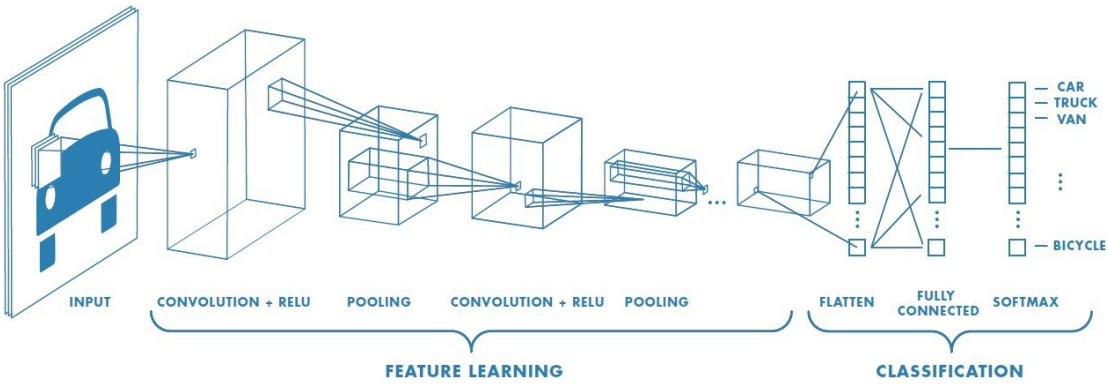


Figure 3: General structure of CNN[2].

### 2.4.1 Convolution Operation and Convolutional Layer

The convolution operation is a mathematical linear function of two real-valued arguments. The heart of Convolutional Neural Networks is the convolution operation. Mathematically, Convolution operates on two functions, and outputs is another function which expresses the change of one function with respect to another. Consider any real-valued function x(t) and a related weighted function w(a) then the convolution operation can be defined as [1]:

$$s(t) = \int x(a)w(t-a)da$$

here, the first function is generally termed as input, and the second function is called the kernel. The output of the convolution operation is often referred to as a feature map.

[2]https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

The discrete convolution operation can be defined as [1]:

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a)$$

The convolution operation can also be denoted by [1]:

$$s(t) = (x * w)(t)$$

The data handled by deep learning approaches are multidimensional in nature; therefore, the convolution operation has to be fit to operate on multidimensional input and kernel. The multidimensional convolution function's variant used in most of the modern CNN is given below [1]:

$$S(i,j) = (I * K)(i,j) = \sum_{m}\sum_{n} I(i+m, j+n)K(m,n)$$

The convolutional layer is the most significant and computationally expensive layer in CNN, which employs convolution operation. The convolution layer has the kernel size and number of kernels programmed. The kernel size gives the convolution layer's receptive field, which is the input area that gets captured in each kernel repetition step.

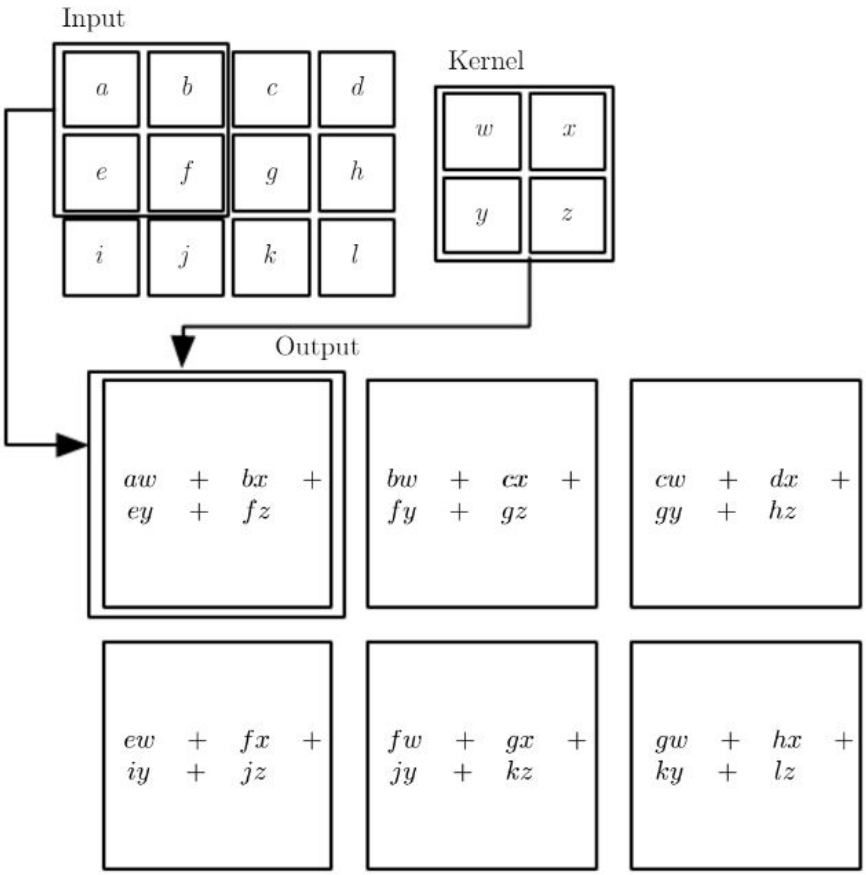


Figure 4: Operation of 2D convolutional layer [1].

Any convolution layer performs a basic convolution operation on the input of that layer and the specified number of kernels of specified kernel size. The kernel repeatedly strides over a comparatively huge input image performing convolution operation at each step. The repetition steps are generally controlled by a programmable stride value of the convolution layer. This operation of convolution layer is schematically shown in Figure 4. When the stride value is 1, the filter is applied across each element in the input one after another in a progressive manner. When stride values are higher, the filter skips a higher number of elements after each step, resulting in the omission of important features.

### 2.4.2 Activation Layer

The output of the convolutional layer is passed on to the activation function equipped layer. This layer performs activation operation on each element of the convolution output. The activation layers add non-linearity to the decision making computation of CNN.

### 2.4.3 Pooling Layer

The next layer in CNN after the convolution layer and Activation layer is the Pooling layer. The pooling layer primarily performs the reduction of the spatial dimensions of the output generated by the convolution layer. The output depth is not reduced with pooling operation, whereas the height and width are altered. A window of specified size strides over input elements and replaces the window elements with statistical summary value in pooling operation. The basically used summary statistic operations include Max, Sum, Average and global pooling. The pooling enables translation invariance in CNN, which means the presence of features is significant, not their location of features in the image. The max-pooling layer is one of the most popular pooling methods used where the sliding window matrix strides over the input and replaces with the maximum value in the window and repeats till the end of the image. Thus, the image gets down-sampled, which helps combat the overfitting problem and reduces the computational complexity of CNN. The max-pooling layer operation is illustrated in Figure 5.

### 2.4.4 Flatten Layer

The flatten layer is used to convert any multidimensional data to a single directional data. This layer converts multidimensional data from the CNN-pooling pipeline to a single dimension vector that fits for an artificial neural network for later processing. This layer acts as an intermediate layer between the convolution layer and the fully connected layer.

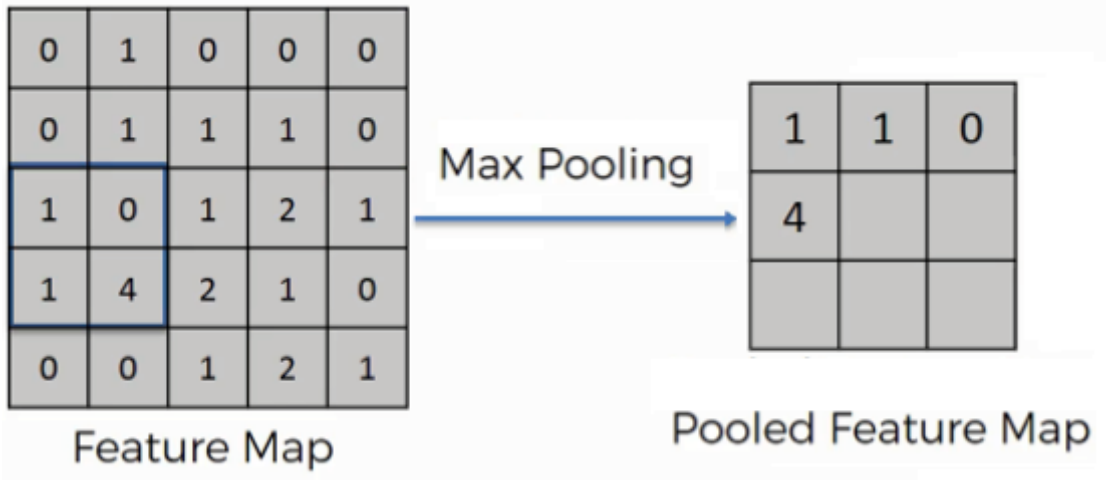


Figure 5: Basic max-pooling layer operation [1].

### 2.4.5 Fully Connected Layer

The last layer of CNN is a fully connected layer, which resembles an artificial neural network with every neuron from the previous layer connected. These layers perform logical reasoning of the features from the convolutional layer to classify the images accurately.

## 2.5 3D Convolutional Neural Networks

The 3D Convolutional Neural Networks (3D CNN) is an extension of convolutional neural networks that can perform classification to analyze a set of images or video. In 3D Convolutional Neural Networks, the convolution and pooling layers are replaced with a 3D convolution layer and 3D pooling layer. The 3D convolution operation has cube-shaped kernel weights convoluted over a set of frames input. The operation can find a relation between pixels of different frames. The 3D convolution operation can be expressed as below where y is the output of the convolution layer, F is an activation function, W is the weight matrix, V is input, and b is bias [12].

$$y = f\left(\sum_i \sum_j \sum_k W_{ijk} V_{(x+i)(y+j)(t+k)} + b\right)$$

The 3D convolution and pooling enable the capture of both spatial and temporal information from the input. Each 3D kernel weight represents only one single feature, which is across the cube. A stack of frames is extracted from the video and provided as input to the 3D CNN, and features from this 3D CNN will contain both spatial and temporal information [12].

## 2.6 Recurrent Neural Networks

The Recurrent Neural Networks [15] are a powerful class of neural networks that operate on sequential data. RNN is specialized to extract temporal information from time-varying data or patterns. The RNN updates the captured information from every time step in a set of nodes called the hidden state. The hidden states act as a temporary memory which helps to process the sequential data.

In Recurrent Neural Networks, the same operation repeats for every time step of input. Unlike the other deep learning methods with an individual set of parameters at each layer, RNN reuses the same parameters at every layer. The RNN generates output data for each time step of the sequential input, which depends on input and states of the previous time step.

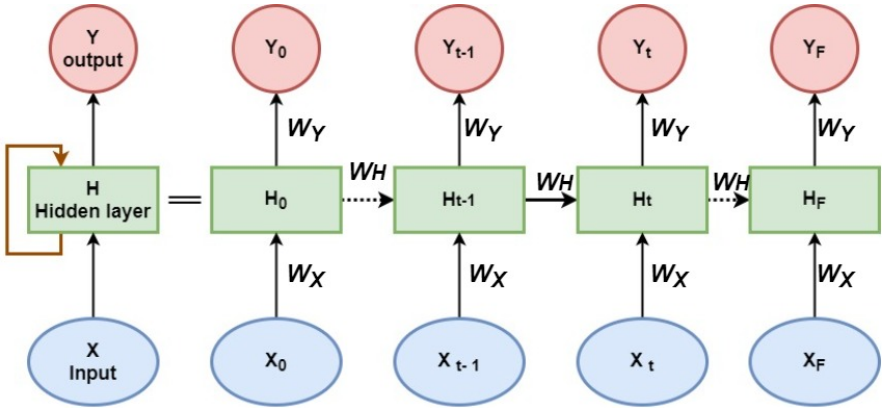


Figure 6: RNN architecture and unfolding through time.

Figure 6 shows a simple RNN and its unfolding with time. The unfolding of RNN means to represent complete network for a full sequence of input. The network has input $X_0,\ldots,X_{T-1},X_T,\ldots,X_F$, corresponding sequence output for those inputs are are $Y_0,\ldots,Y_{T-1},Y_T,\ldots,Y_F$ and the hidden states are represented by $H_0,\ldots,H_{T-1},H_T,dots,H$. The three parameters which are shared among all time steps are $W_X,W_H,W_Y$ which are weight matrix from input to hidden layer, hidden layer to next hidden layer and hidden layer to output [52].

$$H_t = \sigma(W_X x_t + W_H H_{t-1} + B_H)$$

$$Y_t = \sigma(W_Y H_t + B_Y)$$

The above equation represents the output of the hidden states and sequence where $\sigma$ is the activation function and B is the bias. The parameter $W_X,W_H,W_Y$ are updated while training of network using Back Propagation Through Time(BPTT). BPTT is an extension of gradient descent employed in RNN to compensate for its complexity due to dynamic processing from time-delayed updating of RNN input [47]

## 2.7 LSTM

The recurrent network is very capable of obtaining features with temporal information. The action classification application extracts the temporal features from the video frames. An RNN with a long sequence is vulnerable to forgetting the earlier inputs in sequence [16, 15]. Also, when the network is very long, much computation is performed on neural networks, the gradient which controls the values in the network shrinks to a very minimum. Thus, such a gradient will not contribute much to learning and is referred to as vanishing gradient [53]. Long short-term memory [48] introduced with a solution for combating problems such as vanishing gradient and long term dependency problems encountered in vanilla RNN flavors. The extensive indefinite continuous input sequence to LSTM causes the network to grow indefinitely, resulting in the breakdown of the network. This problem was solved with a novel forget gate that helps the internal networks state reset in appropriate time [54].

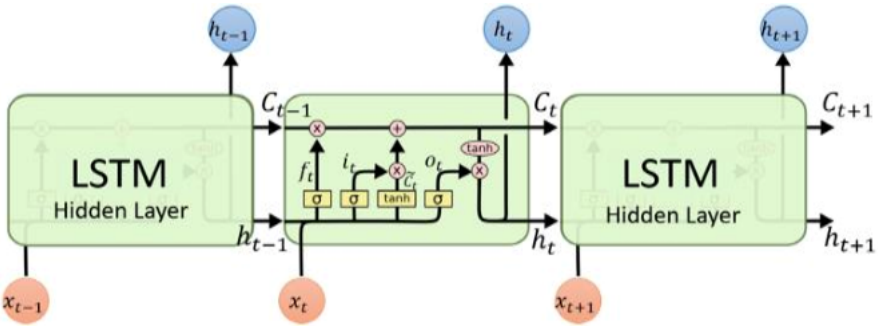


Figure 7: LSTM architecture[3].

The LSTMs are composed of a chain-like repeating structure of four neural layers interacting in a different manner, as shown in Figure 7. This basic unit repeating is called the memory cell or LSTM cell. The cell internally maintains a cell state, which represents all global information acquired by the network, and this cell state helps to compute the output of every state known as the hidden state. The cell state processes the sequence of information using three gates: forget gate, input gate, and output gate.

Firstly, the forget gate performs the removal of information from the cell state, which is not significant for the task performed with the network. The forget gate task operates with the help of a single neural layer with sigmoid activation. The output from the previous time step and input of the current time step is concatenated and provided to the sigmoid layer, and the output of this layer is added to the cell state using an element-wise multiplication operation. The operation can be represented in the equation below [3]:

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$

The next part is the input gate, which contains two gates: the sigmoid and tanh layers. The input is the same as the previous gate, which is the concatenation of input

and output of the previous layer, fed into two neural layers with sigmoid and tanh layer. The outputs of both layers are concatenated element-wise and later add to the cell state of the current cell. This operation will help identify the information that should be retained from current input and added with cell state. The equations below show the operations of this state [3].

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

The last part is the output gate, which deals with generating the output of the current cell. This gate forms with one neural layer with sigmoid activation and one with tanh activation. The sigmoid activation layer obtains the parts of the cell state that needs to be filtered. The output of the cell obtained by element-wise multiplication of cell state passed through a tanh and outputs of the sigmoid layer. We mentioned below the equation for the output from the tanh layer and output of the cell[3].

$$o_t = \sigma(W_o [h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * \tanh(C_t)$$

This output is the hidden state of the LSTM as well.

## 2.8 Activation Functions

The activation function has already been explained in previous sections. In the following part, we will describe a different kind of activation function used in this thesis.

### 2.8.1 Relu

The rectified linear unit is very commonly used in deep learning models and showcases superior performance due to the sparsity. Relu is a piece-wise linear function that grounds the negative part to zero and maintains a positive part. The Relu can be formally described as below [55, 56, 57]:

$$f(x) = \begin{cases} x_i & \text{if } x_i \geq 0 \\ 0 & \text{if } x_i < 0 \end{cases}$$

### 2.8.2 Sigmoid

The sigmoid or Logistic Activation Function with an S characteristic curve graph. The sigmoid range between zero to one is making it perfect for applications with probability prediction. The function of the sigmoid is described below: [58, 59].

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

[3] https://colah.github.io/posts/2015-08-Understanding-LSTMs/

### 2.8.3 Tanh

The hyperbolic tangent function is very similar to the sigmoid function, where the range is between (-1, 1).In the tanh function, negative values are mapped to negative, and zeros values are mapped to zeros. This activation function is more suitable for a two-class classification model. The tanh function is much faster than sigmoid function. The tanh function can be formulated as below [60, 61]:

$$f(x) = \tanh(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})}$$

### 2.8.4 Softmax

The softmax function, also known as the normalized exponential function, is the multi-dimension generalization of the exponential function. The equation of the softmax function is given below:

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

# 3 Methodology and Experiments

This chapter focuses on the methodology and the experiments performed. Firstly, we describe the datasets used for our experiments. Next, we discuss the details of our proposed model followed by the explanations of the baseline models we consider. For each model, we present their pre-processing steps, architecture, and implementation details.

## 3.1 Dataset

The activity recognition requires RGB video-based dataset and the size of this dataset must be large in order to correctly identify the actions. For the purpose of training and testing models, we use UCF101 [2] in this thesis. It is one of the largest datasets available for human actions that consists of user-uploaded realistic videos with cluttered background and camera motion. This dataset contains 13,320 video files with a total time duration of approximately 27 hours. These videos are categorized into 101 classes for supervised algorithms [2] and have a resolution $320 \times 240$ with a frame rate of 25 fps with audio. Figure 8 display a sample frame from each action class in the dataset.

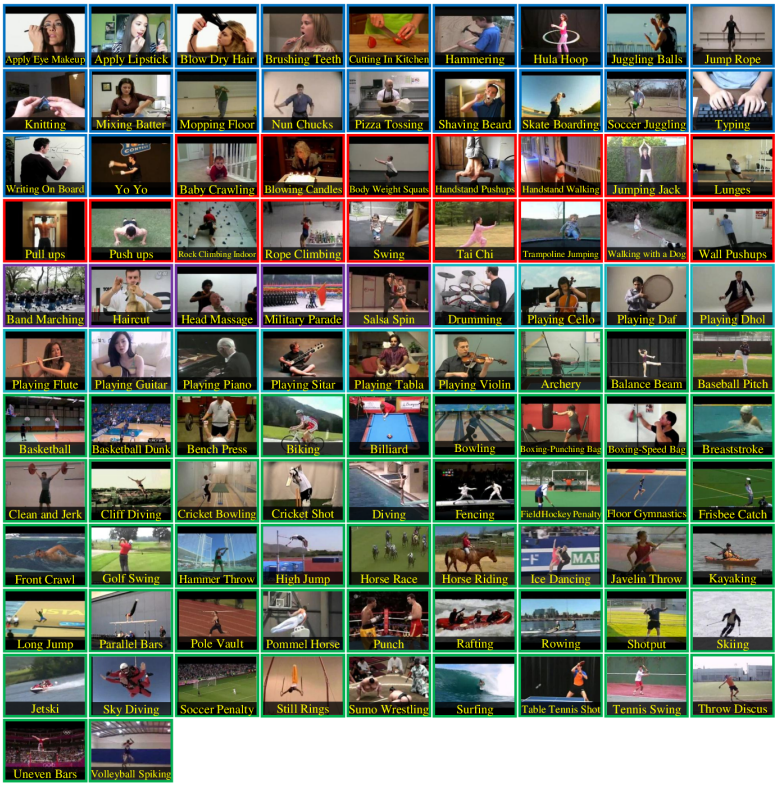


Figure 8: Sample video frame from each class of UCF101 [2].

## 3.2 Proposed Action Recognition Model

In this thesis, we propose a hybrid model, a combination of two deep learning algorithms for action recognition. In particular, our hybrid model is a combination of 3D CNN and Bi-LSTM, which takes a set of video frames as input and outputs the action class. The 3D CNN extracts the spatio-temporal information from the dataset videos. This spatio-temporal information is the output features vector from the 3D CNN after processing the video's frames. For extracting the sequential information, we use a Bidirectional Long Short-Term Memory. The Bi-LSTM is a stack of two simple LSTM, where one runs forward direction, and the other runs backward direction to process the sequences. This ensures that the model extracts sequential spatio-temporal information more effectively by analyzing sequence from both forward pass and backward pass of features from 3D CNN [62].

### 3.2.1 3D CNN and Bi-LSTM Based Action Recognition

Consider actions such as running, javelin throw, high jump, long jump, cricket ball all of them have a running phase initially. The initial frames of all these actions would classify them as running, whereas the later frames will conclude that it is a different action, which starts with running. Hence, considering only the initial set of frames for analysis often results in the wrong classification. In some other cases, all the initial frames will depict different actions, but later frames would be similar. Thus, initial frames have to be given priority in this case. Considering these two cases for accurate identification of action requires both forward and backward direction analysis for contingencies in video frames. For classifying long videos that contain actions, one method is to classify using intermittent frames extracted by skipping a specific number of frames. Thus, we can extract and analyze the frames from all parts of videos equally. If we skip a large number of frames in between, then classification may not be efficient because the temporal information between two frames would be lost. Thus, even the frames that are taken intermittently may not help in precise classification.

For accurate action classification, it is significant to identify the exact number of frames that are required to perform this task from videos because analyzing entire videos is computationally impossible. Adopting the exact same number of frames from small and large videos for action classification may not be an intelligent method. Also, using entire frames from each action video could be computationally very expensive. Implementing a method that extracts a specific number of frames according to the video's length is the best solution to this problem, but this may require an advanced method to identify the number of frames that has to be extracted, which again adds to computational complexity [63].

We can solve the above-mentioned concerns by extracting a specific number of frames as a set from videos not just once, but a fixed amount of time from different parts of a video. Thus, we can engage frames from every part of the video to analysis, and from every position, we extract a set of frames, not just one to capture temporal information from them. This way, we ensure we have captured spatial and temporal

information from all parts of the video. Above all, using this method of extracting frames can investigate the long term or global temporal dependency in the input. We experiment on a hybrid architecture that is a combination of 3D CNN and Bi-LSTM. Our method consists of three steps:

1. **Extraction of local spatio-temporal features**: We split the video into several small videos with a fixed length, and we use 3D CNN to extract the local spatio-temporal features from these segments of the video.

2. **Extraction of global temporal features**: The Bi-LSTM model takes input as the sequences of the spatio-temporal features. The Bi-LSTM computes the global temporal features from local spatio-temporal features.

3. **Classification:** All the features which are extracted using the Bi-LSTM is analyzed, and actions are classified using a simple linear classifier. We use a softmax classifier in this model.

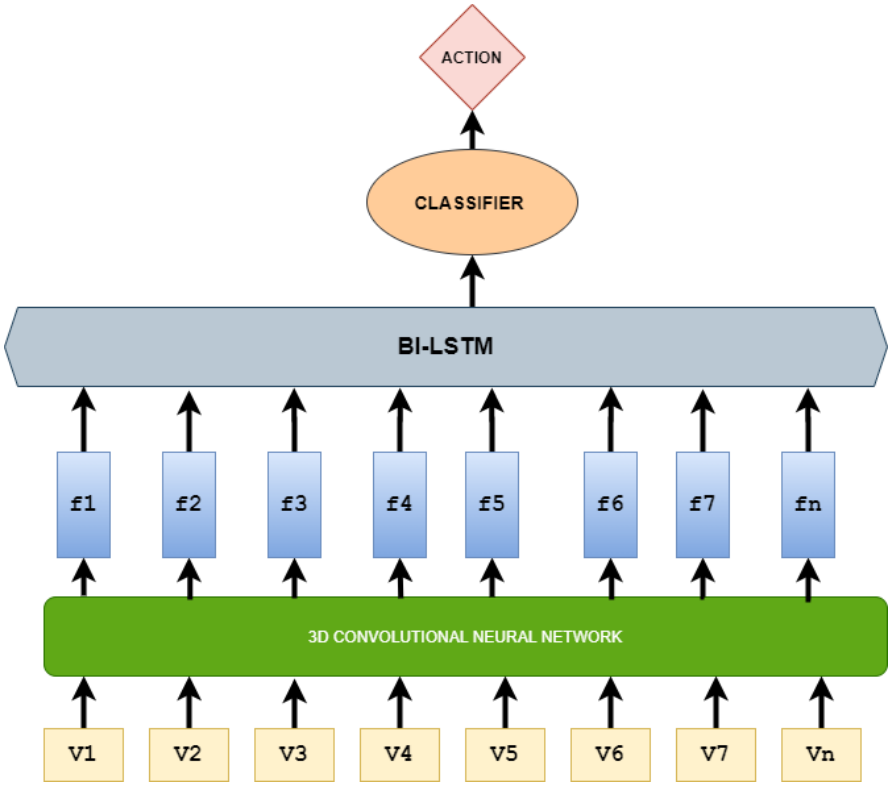


Figure 9: The diagram of complete architecture of the 3D CNN and Bi-LSTM based model.

This experimental model uses 3D CNN and Bi-LSTM to extract both local spatio-temporal and long term temporal information. The extracted spatio-temporal features

can be used with a simple linear classifier like softmax for the classification of actions. This is the first approach ever for the action classification problem, which utilizes 3D CNN and Bidirectional LSTM. The 3D CNN extracts local spatio-temporal features; hence the 3D CNN network needs not to be very deep, and LSTMs is the best choice to learn complex sequences from the input. The short term spatio-temporal features are extracted using the 3D CNN network, and long term spatio-temporal dependencies are explored in both forward and backward directions in sequence using Bidirectional LSTM. Figure 9 depicts the proposed model, $v_1, v_2, \ldots, v_n$ are extracted frames sets that are provided as input to 3D-CNN and features $F_1, F_2, \ldots, F_n$ are the outputs from 3D-CNN which are passed on to Bi-LSTM. The Bi-LSTM and a classifier are employed to identify the corresponding action class of the video.

**Implementation Details** We extract the features from video using 3D CNN, which is pre-trained on the UCF 101 action recognition dataset [2]. The feature is fed to Bidirectional LSTM to obtain global temporal features and extract action class from them. The entire process is depicted in Figure 9. The model is implemented with Keras library with a TensorFlow as the backend. A powerful laptop with a Core i7 processor and Nvidia Geforce RTX 2060 graphics processor was used to experiment. The UCF 101 dataset has already generated three sets of train/test splits for the action recognition task. For each split of train/test samples, we split the train samples again to the train and validation set. Each video is split into 25 splits for the entire dataset, and each split will contain 10 RGB frames to identify the action. The 25 video splits are fed to 3D CNN to obtain the corresponding 25 feature vector of size 512. The 25 feature vector is passed on sequentially to Bi-LSTM to identify temporal features and predict accurate action class from features.

**3D CNN Section** The 3D CNN can effectively identify visual patterns from videos directly from their pixels without any preprocessing. The trainable filters and local pooling operations of 3D CNN have the ability to detect hidden patterns in frame pixels, which identifies spatial and temporal information in them [64]. The method includes extracting spatio-temporal information of many small segments of videos extracted from one video. The model extracted 25 video splits from each video, with each video split has a length of 10 frames each. 3D CNN is required to exact only local spatio-temporal features, which means a simple shallow model would be sufficient. There we are using the same architecture used in Section 3.3.5.

The first task in this method is to train the 3D-CNN architecture and obtain a decent accuracy for the model for the action classification task. The procedure used to train the network is described in detail in Section 3.3.5. After the network reaches the saturation point of accuracy, the model weights have to be saved. This network is schematically shown in Figure 15. The next task is to implement the 3D CNN architecture again, the same as the network before. This network will be used to extract the features, not to classify any actions. Therefore the last dense layer in the network with softmax activation was removed, and the network terminates after the first dense layer. After

removing the final layer, the output of this network will be a feature array of length 512. Now the model weights saved in the first task after training are loaded to the new modified network. All 25 splits of every video are passed to the 3D CNN network and to produce a 512-byte feature for each. Thus, for every video, $25 \times 512$ sized array of features will be saved.

**Bi-LSTM Section** LSTM is the proven deep learning algorithm that is the best choice for learning long term dependencies from sequences. We have employed a bidirectional LSTM to study the global temporal pattern in features in forward and backward directions. For every instance of video, a feature map of size $25 \times 512$ is extracted from a pre-trained 3D CNN model. This feature vector forms the input for the bidirectional LSTM. The model architecture consists of a bidirectional LSTM with 512 neurons and a sequence length of 25. A dropout layer follows the Bi-LSTM layer with a 0.2 rate, and the last layer is a fully-connected layer with a softmax classifier. The final stage of this model is the classification of action using the softmax classifier. From the sequence of features from videos, the softmax can calculate the M-way class score. The loss function used in this method is categorical cross-entropy, and the optimizer used is Adam with learning rate of 0.0001. The batch size for input was fixed as 32 and the model was trained for 100 epochs to get a saturated accuracy rate.

## 3.3 Baseline Action Recognition Models For Evaluation

The central purpose of this work is to develop a skilled deep learning architecture that derives spatio-temporal features from videos using deep learning algorithms. This architecture is then used to implement a model for the action recognition task. The performance action recognition task is used to verify the effectiveness and precision of the spatio-temporal features. Further in this work, we compare the proposed model with five other successful models to evaluate the spatio-temporal features.

Firstly, the CNN model inspired by the large scale ImageNet model ResNet50 is fine-tuned to extract the spatial features alone and study the significance of spatial features in action recognition. The second model uses CNN based approach with optical flow input to analyze the temporal context data. The next model is based on two-stream CNN, which uses one stream each to extract the spatial and temporal information by processing video input. It is worth mentioning; both networks get fused to a single spatio-temporal network [10].

The fourth approach for comparison with the proposed method is based on LSTM. In this method, the input video frames are processed by simple CNN for their features, and LSTM processes these features to find its sequential information. In the proposed method, a pre-trained ResNet 50 is used for the CNN part, which provides a feature map of 1024 dimensions. To extract more detailed sequential information, a long short-term memory method is used. A softmax classifier determines the final action category from the final states of the LSTM network [65]. The final model is a 3D CNN based approach to extract spatio-temporal features and classify action by

analyzing these features.

There are a few good datasets available online for carrying out the aforementioned experiments. For which we have chosen UCF101 [2] for training and testing the action recognition models. First, we implement all the baseline models for evaluation, train them on the UCF101 dataset, and study their performance on action recognition tasks. The proposed model will be evaluated on this dataset using the accuracy obtained. We try to identify the best fitting model that achieves the highest performance.

### 3.3.1 Spatial CNN Based Action Recognition

In this section, we discuss the first baseline model to evaluate the effectiveness of the proposed spatio-temporal feature-based action recognition method of the thesis project. This model is a CNN-based approach for the identification of action taking place in the video. CNN is a prevalent approach to analyze the spatial features of images. The action recognition requires analyzing both spatial and temporal context, whereas conventional CNN cannot interpret the temporal information from data. Intuitively, there is a wide range of actions that can effortlessly be identified with the objects associated with them.

The presence of some particular objects in videos clues the action strongly associated with them [9]. For instance, objects like the bow and arrow are always associated with action arrow shooting. Similarly, the gun is associated with action shooting; the violin is with playing the violin, etc. Conversely, actions such as walking, running, jogging cannot be accurately identified with spatial details alone. This baseline model attempts to learn the significance of spatial context in action recognition.

This baseline model for action recognition using CNN can exploit the advantages of CNN concepts like large scale ImageNet classification [66] and Transfer learning [67]. We depend on large scale image classification networks, which are proven models for large scale object classification, for the accurate classification of objects in the video. These recognized objects guide to the action performed in the video. The transfer learning technique helps boost the efficiency of any network by sharing the weights from an already trained similar model on a similar classification task. In this model, we employ a successful large scale classification model and transfer weights of a well-trained model to improve the classification accuracy of video objects. The model is implemented for action recognition task with weights for the image classification loaded cannot be expected to perform effectively. Thus, fine-tuning of the model was required for closing the gap between the two tasks. The model loaded with pre-trained weights is trained again with the action recognition dataset in order to tune to the action recognition task.

The input for the model is a frame from the video that can help in the classification of the action. But every single RGB frame may not be equally discriminative; therefore, every single frame will not be able to classify actions without flaws. The identification of the best frame is an arduous task because every frame is affected by occlusions, cluttered background, and motion blur [68]. This problem can be cleared by using input as an image with multiple RGB frames as sub-shots.

The model is trained with frames from videos and a label of action performed in the video. The model learns to identify the objects present in frames accurately with the help of ResNet50 and investigates the link between the action and these objects. The patterns of spatial features of these objects are analyzed and learned by the network to perform the classification task. After training, any videos provided to this model will be scanned for objects, and action linked with the objects is identified. This model's advantage is that they are computationally less expensive than other models used for action recognition such as optical flow, two-stream networks, 3DCNN, and LSTM based models. This model has the easiest and less time-consuming training procedure. Transfer learning also reduces the time for training and complexities due to dataset size-related issues.

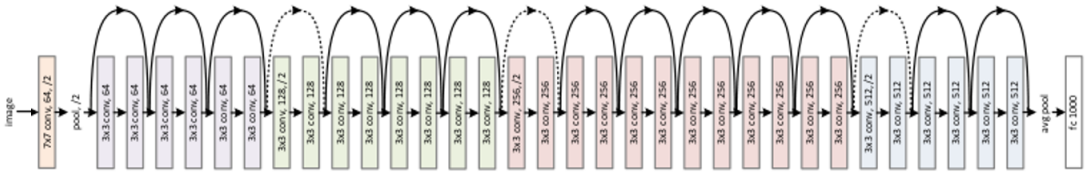

Figure 10: Basic ResNet50 model[4].

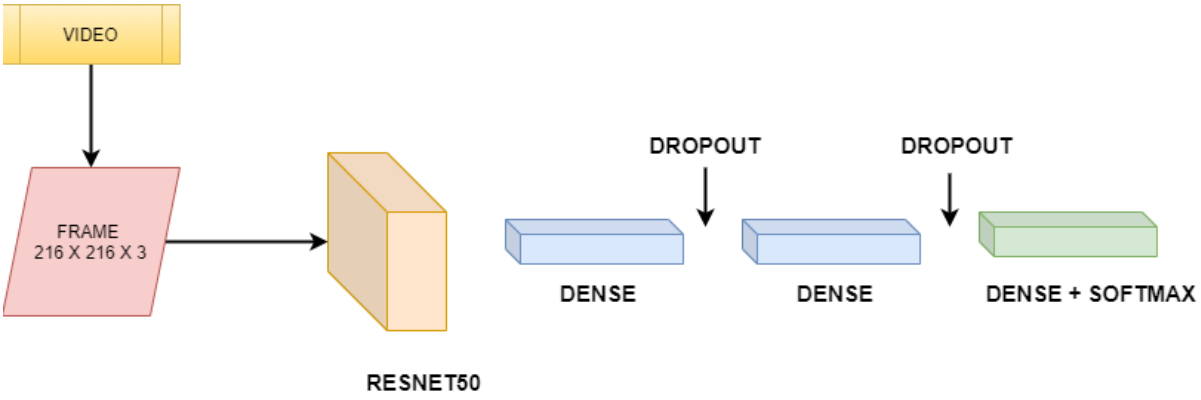


Figure 11: Spatial CNN model used for action recognition.

**3.3.1.1 Implementation Details** The model is developed with the Keras framework with TensorFlow as the backend. The Keras implementation of ResNet50 [69] architecture is used as the base network for this implementation, and the pre-trained weights for the same ImageNet are loaded to this base network. The ResNet architecture is developed with skip connection structures, which boosts the efficiency and

[4]https://www.kaggle.com/keras/resnet50

performance of classification tasks. The base network's loaded weights will be frozen and are not allowed to participate in any further fine-tuning on the model. The basic structure of the Reset50 base model is schematically shown in Figure 10. On the top of the base model, we add a flatten layer followed by two sets of dense and dropout layers and finally, a softmax classification. This added layer forms the top network for the model. This top network will be fine-tuned with the UCF101 dataset. The complete pipeline of this model is shown in Figure 11.

The batch size is fixed as 32. From each video file, 16 frames were randomly extracted and resized to height, width, channel dimensions to 216, 216, 3. The video frames were normalized and consolidated together into a single input image with each frame as a sub-shot. The input tensor is formed with a size of 32, 16, 216, 216, 3, which are the dimension of batch size, sub-shot, height, width, and channel. We used an SGD optimizer with the learning rate 0.01, the decay rate of 1E-6, and a momentum of 0.6. The UCF 101 dataset was used for training the model, and this dataset consists of 101 action classes. The categorical cross-entropy was used as the loss function because the task was a multi-class classification.

The early stopping regularize has been used to monitor for the model learning improvements. If the improvement is not evident at the end of epochs, the training can be stopped to avoid over-fitting. We fixed the number of epochs as 100. The training video sample size was selected as 10000, and the validation sample was taken as 3000.

### 3.3.2 Temporal CNN Based Action Recognition

This section discusses the implementation details of the second baseline model used to evaluate the proposed action recognition model using spatio-temporal features. The evaluation model is implemented using a convolutional neural network, which can learn the temporal context from the videos using the optical flow algorithms. The model uses the stacked optical flow displacement fields from consecutive frames of videos from the UCF101 dataset to classify actions. The motion information from the optical flow between the set of frames is efficient for action classification.

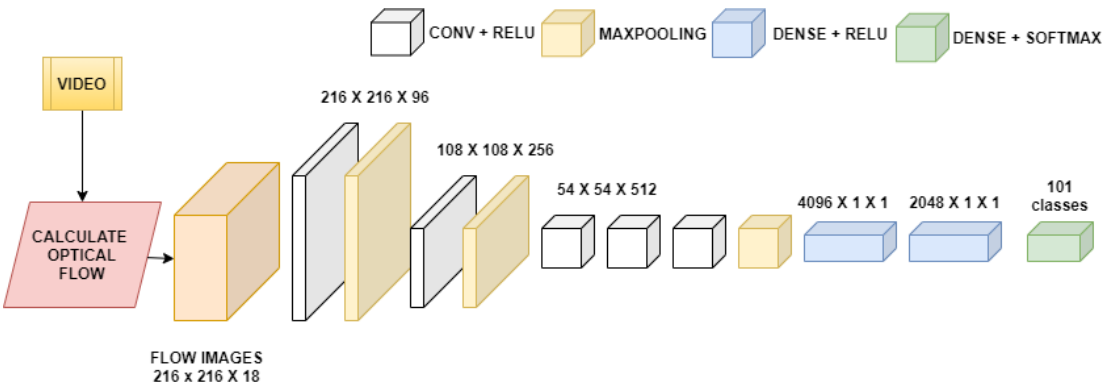


Figure 12: The CNN model for temporal based action recognition.

All the frames from the video files do not participate in extracting features; instead, a set of 18 frames are used by which a lot of redundant data can be avoided. There exist a direct correlation between the accuracy of optical flow and accuracy of action recognition. Also, optical flow is considered as the best motion feature representation for action representation [70]. The optical flow in simple terms is the difference between magnitude and orientation of pixels between a few frames [71]. The optical flow calculates the movement of frames, pixels, or objects of images. In optical flow-based classifications, the flow of motion in frames is the similarity aspect for the classification of actions to the same class. Every action performed will have a set of the characteristic path of motion flow. CNN captures the trajectory of motion boundary in detail for action identification. The optical flow is calculated with the Gunnar-Farneback algorithm [72]. The algorithm computes optical flow by comparing frames one after the other. The first frame is analyzed for the region of interest, and the features present in the region of interest with its location details are extracted. The upcoming frame is also analyzed for all the details like the previous frame; if the locations of the features are different, then the median of the distance between all features is saved. Similarly, all the frames will be compared with this save frame, correspondingly updated, which produces the final output as an optical flow diagram. This method is known as dense tracking, which tracks the motion of all objects in the frames. The dense tracking method is a very accurate way of calculating optical flow comparing to other algorithms but it is a prolonged process.

The UCF101 videos are preprocessed to generate optical flow images. The optical flow images are saved as arrays, and these optical flow data are used to train the model. The model operates on optical flow data; therefore, we cannot use the advantage of large-scale ImageNet classification methods.

**3.3.2.1 Implementation Details** The shape of input to the CNN is $216 \times 216 \times 8$. The first set of layers include a convolutional layer, batch normalization, Relu activation, and max-pooling. The convolutional layer's configuration is with 96 filters of each kernel size $7 \times 7$ and stride 2. The second set of layers also follows the same layers as the previous layer, but the convolution layer changes to 256 filters of each kernel size $5 \times 5$ and stride 2. The third and fourth sets of layers have convolution layers with 512 filters of each kernel size $3 \times 3$ and stride 1, and all other layers are the same. After these layers, a Flatten layer is added, followed by a dense layer of 4096 neurons and a drop out layer with a rate of 0.9. Finally, a dense layer with 2048 units and a dropout layer with a rate of 0.9 is followed by a dense layer with a softmax classifier.

The configuration and layers of the model are shown in Figure 12. Firstly, we would generate the optical flow data for every video to be trained. The model uses an SGD optimizer with a learning rate of 0.01 for the training. The loss function for the training of the network is categorical cross-entropy, and the metrics under monitor while training is the accuracy value. The model checkpoint callback function is used to save the model weights after every significant improvement in the accuracy. Also, to control

overfitting, the early stopping callback stops training when there is no improvement found while training a few epochs. The model was trained for 100 epochs.

### 3.3.3 Two-Stream CNN Based Action Recognition

The third baseline for evaluation of proposed spatio-temporal features based action recognition is discussed in this section. In the two previous sections, we have discussed models with spatial information based on CNN and a temporal context-based CNN, and both these methods use either spatial or temporal context to classify. This baseline approach used both spatial and temporal information to classify the actions from video by constructing a model that combines both the previous CNNs. This provides the advantage of both spatial and temporal features in the classification of action. The architecture, as shown in Figure 13, consists of two streams of CNNs, one computes spatial features, and the other computes temporal features. The outputs are determined by taking the fusion of scores of both individual CNN's. In order to fuse the two networks, we experiment on two fusion methods, which are Average and Maximum fusion. The model architecture is shown in Figure 13.

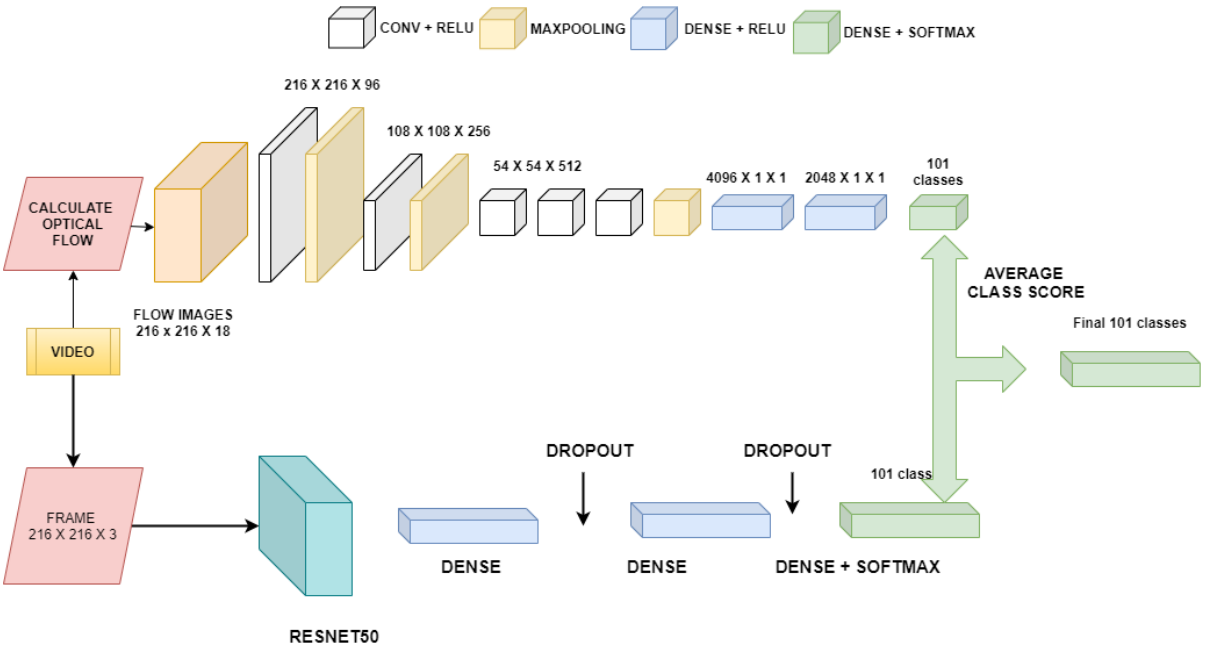


Figure 13: Two-Stream CNN based action recognition model.

**3.3.3.1 Implementation Details** The two models from both previous sections are implemented in the same order as explained before, and the weights saved from both individual task is loaded to this model. The architecture is not feasible to be trained together; therefore, the only prediction of action is performed. The video is loaded and provided as input to each stream after corresponding preprocessing for each

stream, and simultaneous prediction is performed on this network. The output from both spatial CNN and temporal CNN is retrieved and these output tensors are merged, which forms the final output of the classification. The model is not feasible to train them together; therefore, the model shares weights that have been trained individually. The model is loaded with already trained weights. Thus, there are no parameters to be programmed for the model.

### 3.3.4 CNN-LSTM Based Action Recognition

This section discusses another baseline model implemented with the convolutional neural network and long short-term memory for evaluation of the proposed research thesis model. This evaluation model has two sub-models, the first sub-model extracts features using a CNN from frames, and the second part learns the continuity in features using LSTM. The CNN is a robust algorithm in image representation and classification tasks, which makes it the best candidate for extracts features from frames. CNN can extract all hidden patterns in images and marks down the tiniest changes in images. The random ten frames from the video are extracted and passed to CNN one by one to predict features of length 1024 for each frame.

The feature extraction is performed with the spatial CNN explained in Section 3.3.1, and we reuse the saved weights of the trained spatial CNN. The feature extraction CNN is first implemented, which is the same as spatial CNN without the final dense layer with a softmax classifier. The features are extracted from the last layer of feature extraction CNN, a dense layer with 1024 neurons.

The temporal features were analyzed from the extracted features. An LSTM based model is employed in this approach for temporal analysis. LSTM can extract any hidden features from the feature sequence and avoids the exploding and vanishing problem. The features are provided to LSTM in each timestamp, and LSTM analyses the temporal context from the changes found in the features. Finally, a softmax classifier is used to classify the actions from the temporal context of features. The model architecture is schematically shown in Figure 14.

**3.3.4.1 Implementation Details** All the videos from the UCF101 dataset are preprocessed and normalized and provided as input to the feature extraction CNN to predict the features from them. For every video, the selected 10 frames are predicted using the CNN, and all 10 features are saved in an array for later temporal context analysis.

The ten video frames are extracted randomly and preprocessed by resizing and normalizing frames. The frame is resized to dimension $216 \times 216 \times 3$. The CNN implementation details are explained in Section 3.3.1.1. The features array of length 1024 is passed on to LSTM. The input shape for the LSTM is $10 \times 1024$, which are sequence length and feature size, respectively. LSTM based model is utilized for sequential learning. The model contains an LSTM layer with 1024 internal nodes and also enabled returns sequence. A dropout layer and the flatten layer is added after the LSTM

layer. The last layer is a fully-connected layer with 101 neurons, which is the number of classes in the classification task. Also, this layer has a softmax activation function, which helps in classification. The model is trained using backpropagation through time using Adam optimizer with learning rate 0.01 and categorical cross-entropy loss function. The RNN is trained on CNN features of videos from UCF101 training split and early stopping is utilized to avoid over-fitting. The maximum training epochs were fixed as 300.

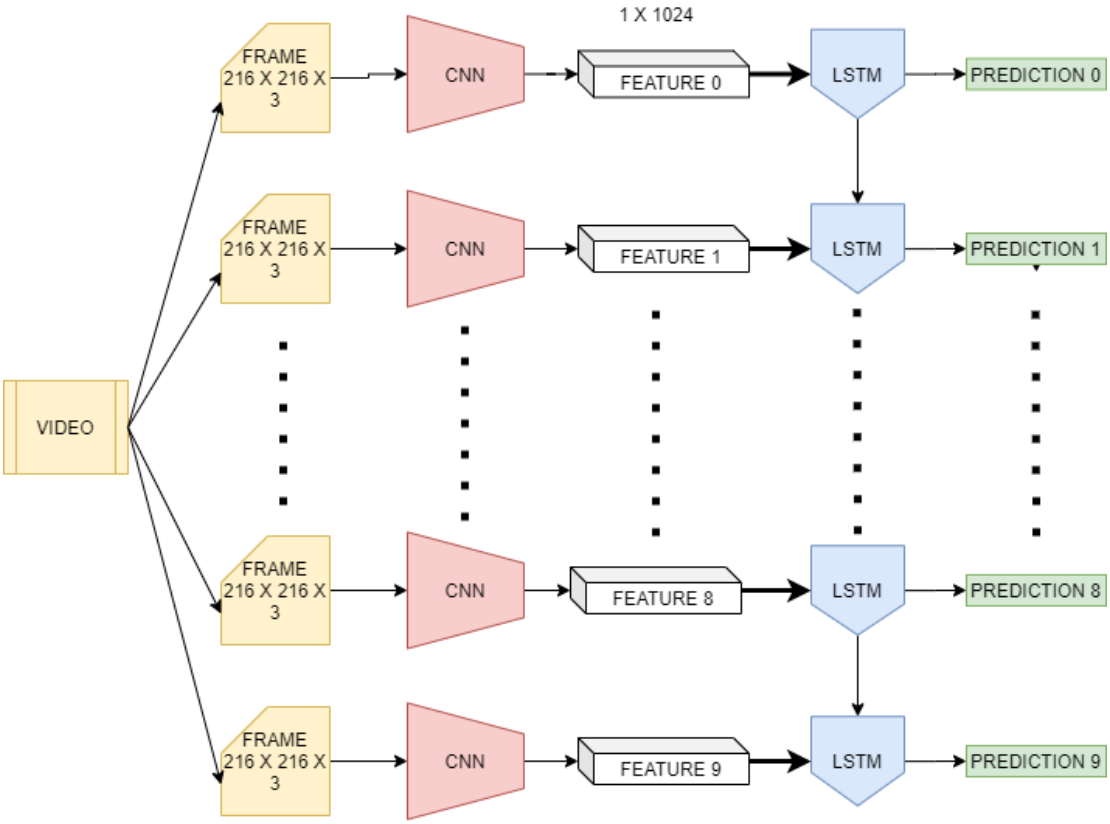


Figure 14: CNN-LSTM based action recognition architecture.

### 3.3.5 3D CNN Based Action Recognition

The 3D CNN architecture is used in this method for action recognition is described following. Primarily a 3D CNN follows similar architecture as 2D convolution neural networks, but all the 2D convolution and 2D pooling are replaced with 3D convolution and 3D pooling. The assumption for this model is that the 3D CNN is more suitable to extract the spatial features and temporal features. A homogeneous 3D CNN architecture with 3 x 3 x 3 kernel could perform better than the previous model. From the features learned, it would be easy for the model to classify the actions with a simple linear classifier [73].

3D CNN is another architecture that befits action classification applications. Unlike other models, the 3D CNN can extract both spatial and temporal features simul-

taneously from the input. In 2D CNN, the convolution and pooling layers learn only spatial features, whereas, in 3D CNN, based convolution and pooling layers can capture spatio-temporal information. The 3D CNN takes input as a set of frames extracted from video files and processes these frames to obtain the spatio-temporal information using 3D convolutions and 3D pooling methods. The spatio-temporal features extracted will contain information regarding objects, scenes, actions, and background in video, and a simple linear classifier is used on these features to classify actions from videos [12, 74].

The architecture of the 3D CNN used in this thesis work is shown in Figure 15. The set of frames from the video file is the input to CNN and passed to a 3D convolution layer with 32 kernels of each dimension $3 \times 3 \times 3$. The features from these convolution layers are processed through a Relu layer to get the areas with significant information. Again the output from the Relu layer is passed to the convolution layer with the same parameters as before, then continued to a softmax activation layer. Later 3D max-pooling layer and a drop out layer are used next. This entire pipeline is repeated change in convolution parameter number of the kernel as 64 instead 32.

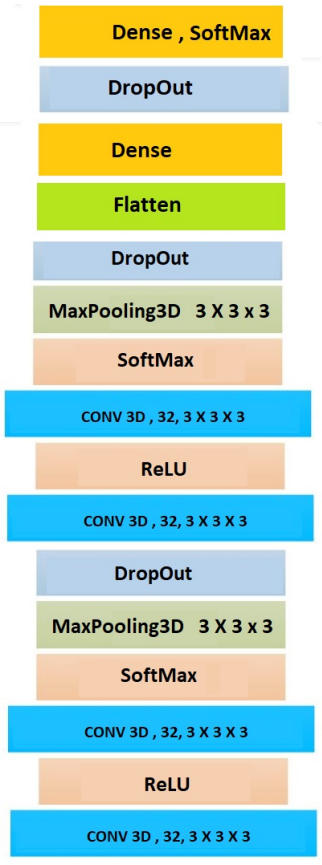


Figure 15: 3D CNN architecture.

This output is passed to Flatten to convert the array of features to a single dimension array then applied to a fully connected layer of neurons with 512 and activation

sigmoid. Finally, a dropout layer is followed by a final fully connected layer with the same number of neurons as the total number of classes and softmax classification.

#### 3.3.5.1 Implementation Details

The model is implemented with Keras with TensorFlow as a backend. The evaluation of this method is conducted with the UCF 101 dataset. Each video file in the dataset is converted to $32 \times 32 \times 10 \times 3$ where $32 \times 32$ is the image size, 3 is the depth standing for color channels, and 10 stands for 10 frames extracted from each video. There is a total of 13320 video files loaded from the dataset for training, from which a set of 32 videos is used as batch input to the 3D CNN model. All the layers before the first max-pooling layer produce features of dimension $32 \times 32 \times 10 \times 32$ and after max-pooling they get reduced to $11 \times 11 \times 4 \times 32$. The next set of convolutions and activation layers will generate an output of $11 \times 11 \times 4 \times 64$ and the second set of max-pooling and drop out will generate output features of dimensions of $4 \times 4 \times 2 \times 64$. The flatten layer converts features to an array of 2048 dimensions, and later fully connected layers of 512 and 101 are used. The dense layer with 101 neurons are the final layer with neurons same as the number of classes and equipped with a linear classification using softmax.

The train and test of the model were conducted with loss function categorical cross-entropy and with optimizer Adam with learning rate 0.001. The experiment was conducted 10 times, with 100 epochs for each time to verify the accuracy.

# 4 Results

The previous chapter discussed the implementation details of different baseline models and our proposed hybrid model for the action recognition task. In this chapter, we report the evaluation results of these models in terms of their accuracy, losses, confusion matrix and class-wise accuracy. Finally, we conduct a comparison study of the proposed model's performance against the performance of the baseline models and other state-of-the-art models.

## 4.1 Performance Evaluation of the Baseline Models

As discussed in Chapter 3, we consider five baseline models for our study; namely, *spatial CNN based action recognition, temporal CNN based action recognition, two-stream CNN based action recognition, CNN-LSTM based action recognition and 3D CNN based action recognition*. We now present the results of the performance evaluation of these models.

### 4.1.1 Spatial CNN Based Action Recognition

As described in Section 3.3.1, the spatial CNN model extracts the spatial features from RGB video frames and predicts the action class from these features. To minimize the time required for training, we used the weights of already trained Spatial CNN for action recognition and did not train the model from scratch.

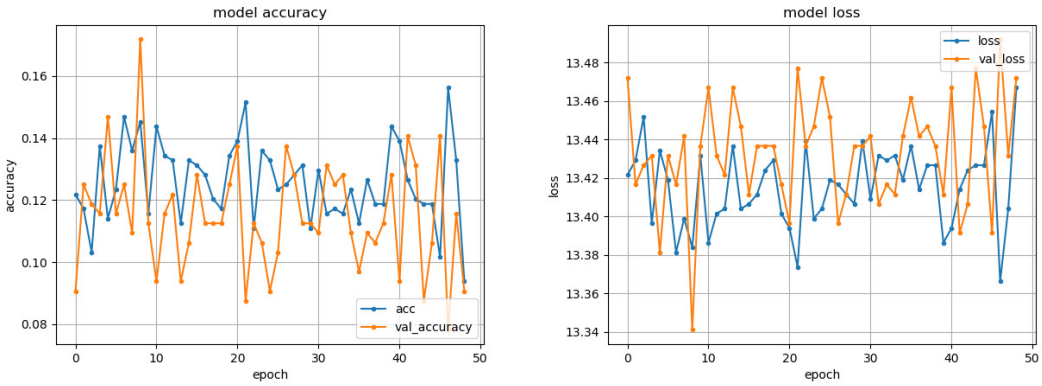


Figure 16: The accuracy and loss plot of the spatial CNN model.

Figure 16 represents the accuracy and loss plot of the spatial CNN model when trained for 50 epochs. We repeated the training for 200 epochs on the *UCF*101 dataset. We observe that the average validation accuracy is around 11.47%, which is not sufficient for an efficient classification model. Since this model identifies the actions from an associated object alone, the chances of incorrect classification are potentially high.

This is mainly because multiple actions can be linked to the same object. This proves that spatial data extraction alone is not sufficient to classify human actions accurately.

### 4.1.2 Temporal CNN Based Action Recognition

The temporal CNN model extracts the optical flow data from RGB video frames, and using this, it determines the action class. Figure 17 depicts the accuracy and loss plot of this model for 50 epochs. Similar to spatial CNN, we used trained weights for this network and trained the model for 200 epochs, and the average validation accuracy obtained was 38.45%. We observe no significant improvements in the validation accuracy on training. The model result provides the evidence that the motion information from frames is an influencing constituent for action classification.

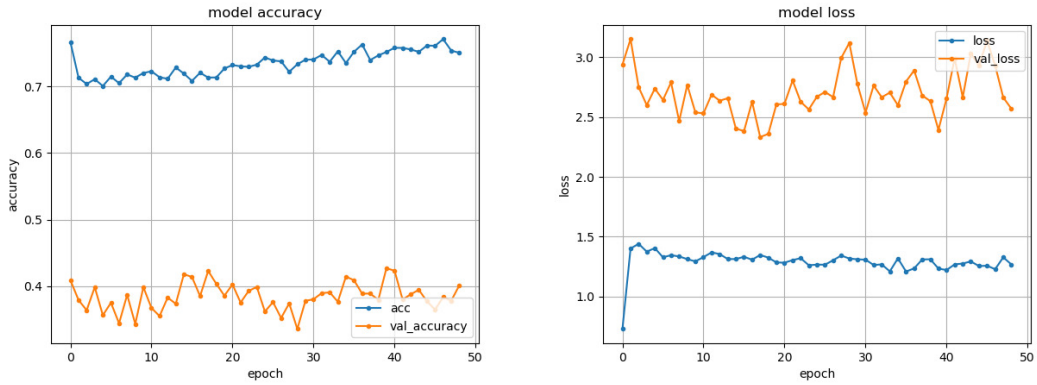


Figure 17: The accuracy and loss plot of the temporal CNN model.

### 4.1.3 Two-Stream CNN Based Action Recognition

This model is a combination of the spatial and temporal CNN models. It takes processed video frames as the input and processes spatial and temporal information in separate CNN to classify the actions. The softmax classifier of both the CNNs provides an output action class score that is fused to form the final class score, which is the final prediction, and this model is schematically is shown in Figure 13. The *average fusion* and *Maximum fusion* are the two fusion methods used to fuse the action class scores.

Table 1: Accuracy of Two-Stream models for Average and Maximum fusion.

| sample# | Accuracy: Average Fusion | Accuracy: Maximum Fusion |
|---|---|---|
| 5000 samples | 45.16% | 42.48% |
| 10000 samples | 45.07% | 42.98% |

We performed two sets of experiments; first with 5000 video samples and the second with 10000 video samples. In the first set of experiments, we observe that the

model with the *average fusion* method was able to classify 2258 videos correctly, and that makes the average accuracy of the model to be 45.16%. The *maximum fusion* model was able to make 2124 predictions accurately (with an average of 42.48%). In the second set of experiments, the *average fusion* method was able to classify 4507 of these samples correctly. The evaluation accuracy of the *average fusion* model is approximately around 45.07%. The *maximum fusion* was able to accurately classify 4298, i.e., an average of 42.98%. These results are tabulated in Table 1 and the results affirm that the *average fusion* method is better for two-stream models.

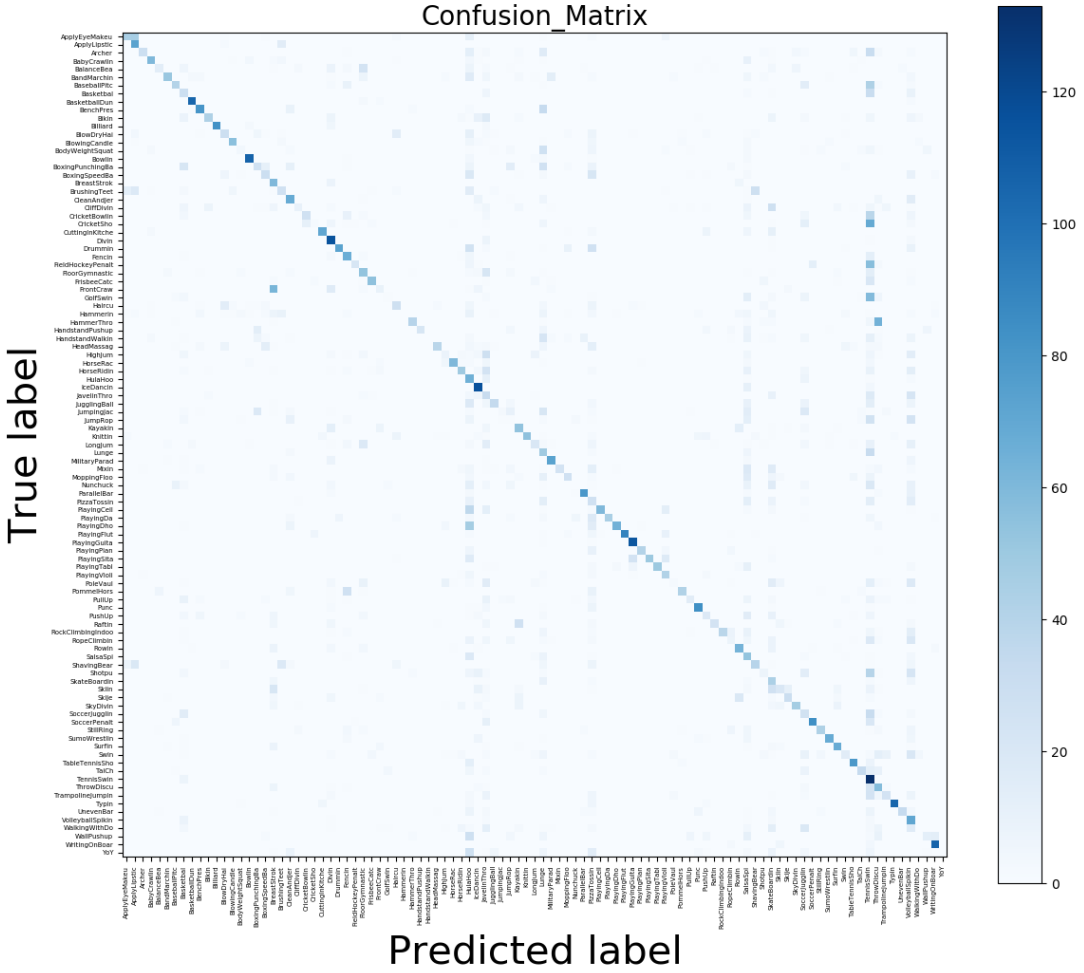


Figure 18: The Confusion matrix of the average fusion method with 10000 samples.

The confusion matrix and the class-wise report of the *average fusion* method for 10000 video samples are plotted in Figures 18 and 19. The results show that the maximum accuracy is obtained for actions that have a strong association with the objects. The lowest accuracy is found for action classes that show similar appearances, for instance, walking and jogging.

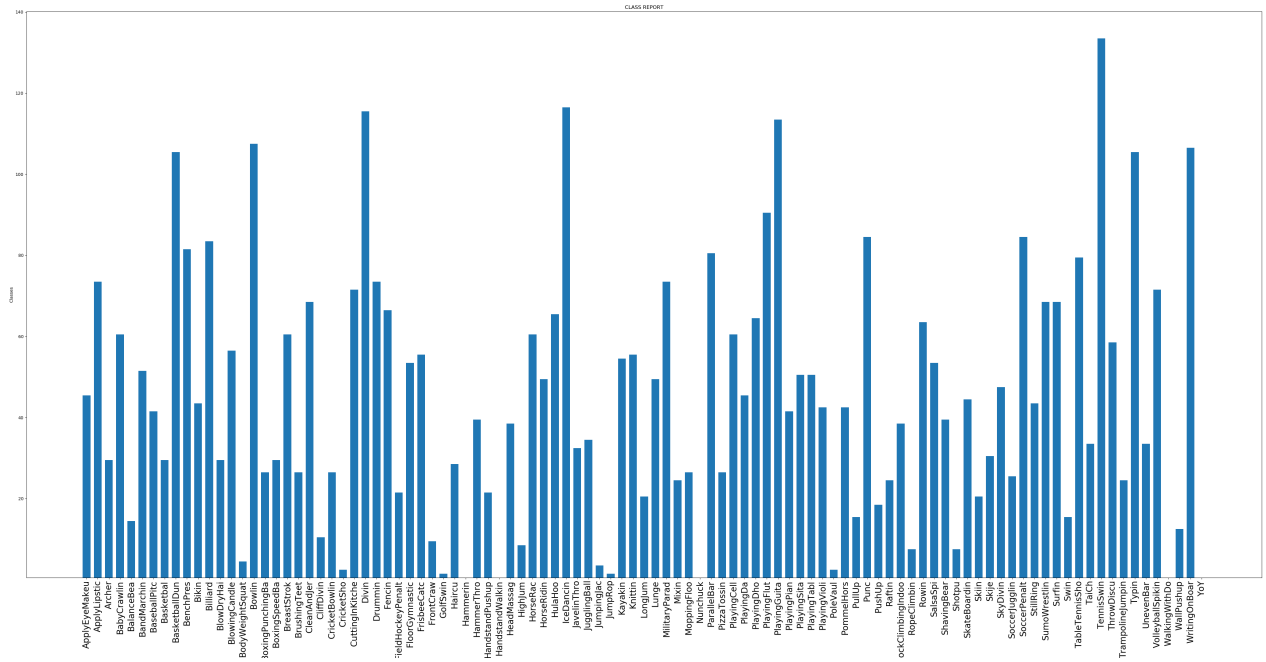

Figure 19: The class-wise accuracy report of the average fusion method with 10000 samples.

### 4.1.4 CNN-LSTM Based Action Recognition

We now evaluate the performance of the CNN-LSTM based model, described in Section 3.3.4. This model was trained for 100 epochs. The training was conducted on 9324 video samples and 2664 video samples for validation. We repeated this training and testing for 5 times to confirm the consistency in the accuracy and loss of the model. The average validation accuracy obtained was 62.04%. The accuracy and loss plot of this model is as shown in Figure 20.

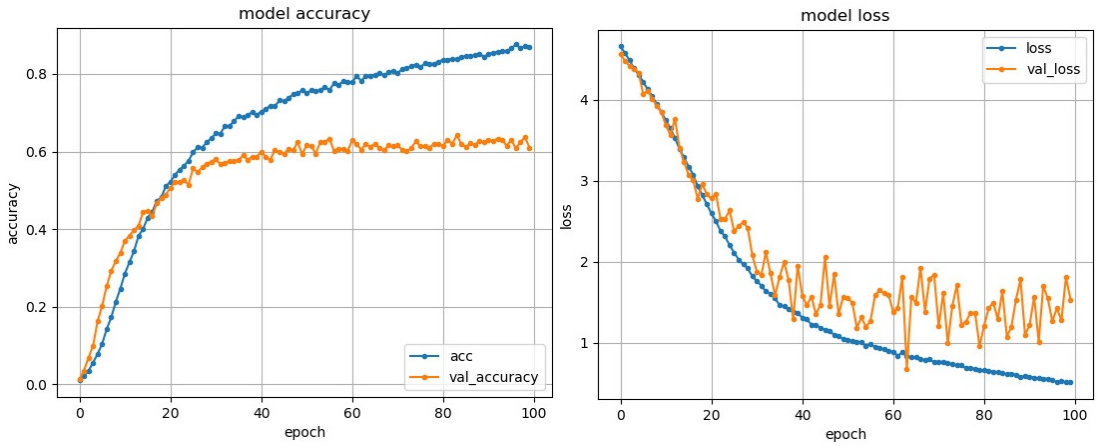


Figure 20: The accuracy and loss plot of the CNN-LSTM based model.

Table 2 reports the accuracy and loss values during the 5 rounds of prediction task after the training. The average accuracy obtained from 5 rounds of tests is 60.876%, and the average loss is 7.426.

Table 2: Accuracy of CNN-LSTM model for 5 rounds of test.

| Test | Accuracy | Loss |
|---|---|---|
| 1 | 62.12% | 11.36 |
| 2 | 62.57% | 5.77 |
| 3 | 60.54% | 9.11 |
| 4 | 57.86% | 5.58 |
| 5 | 61.29% | 5.31 |
| Average | 60.876% | 7.426 |

### 4.1.5 3D CNN Based Action Recognition

The last baseline model that we have considered is the 3D CNN model. We have performed a rigorous evaluation on this model. To begin with, we trained the model from scratch and performed a classification test with video samples that do not appear in both training and validation video samples. This evaluation was repeated for 10 times, to verify the performance of the model. In each cycle, we trained the model 50 epochs. The accuracy and loss plot for one of the training is depicted in Figure 21. The maximum average accuracy obtained during the training is 87.95%, and that for validation is 90.66%. The average accuracy obtained for testing data is 89.41%. Table 3 presents the accuracy obtained during the training, validation, and testing phase. This model has shown better performance than all the other baseline models that we considered for this study.

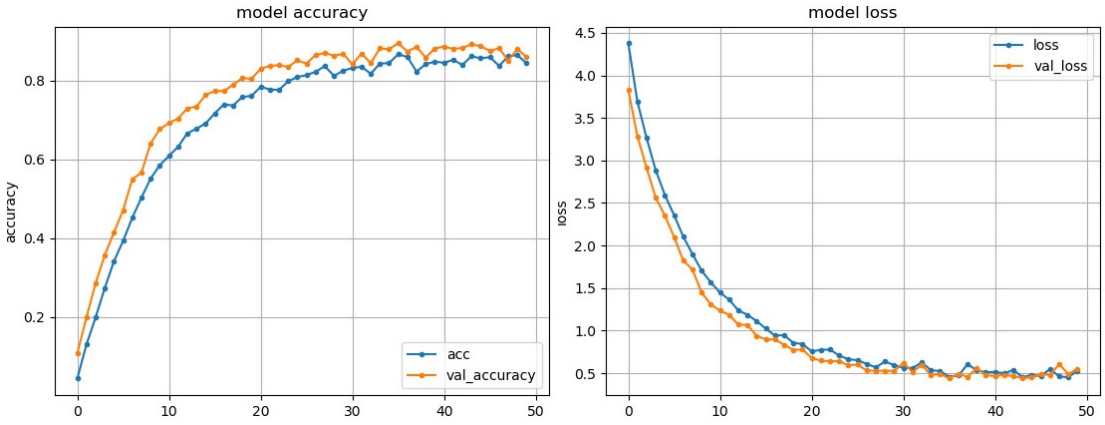


Figure 21: The accuracy and loss plot of the 3D CNN based action recognition training on UCF101 dataset.

The confusion matrix and class-wise accuracy report is depicted in Figure 22 and

23 respectively. The 12 action classes were predicted with an accuracy above 90%. The action classes such as haircut, skiing and walking with a dog, have the lowest accuracy.

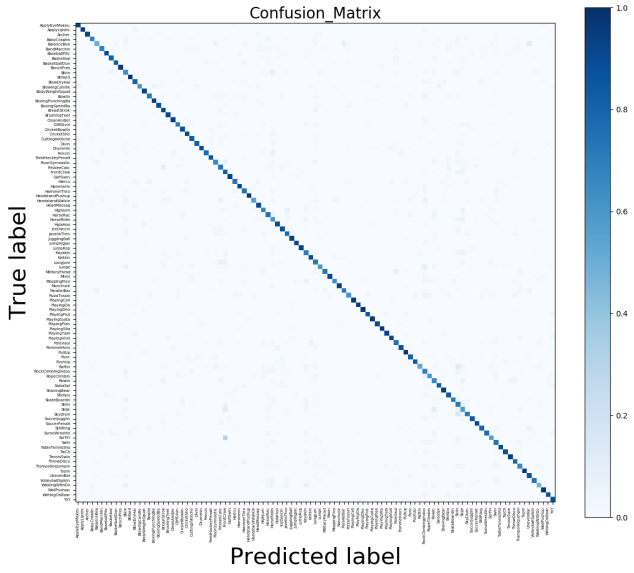


Figure 22: Confusion matrix of the 3D CNN based action recognition.

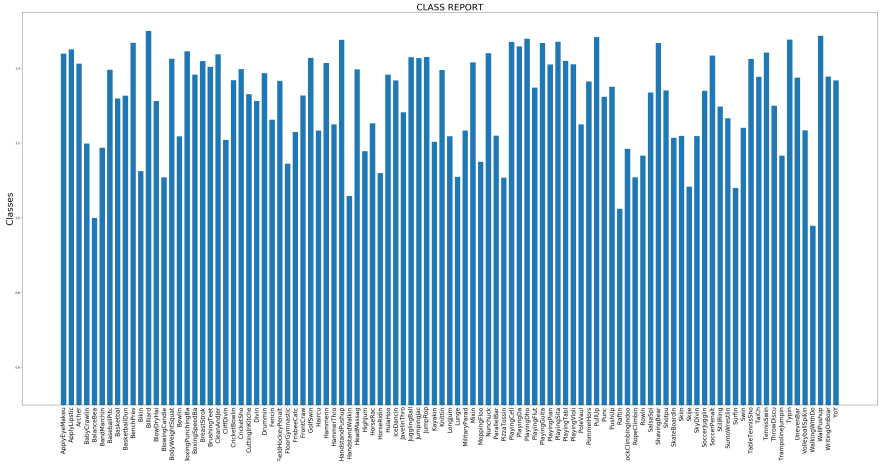


Figure 23: The class-wise accuracy report of the 3D CNN based action recognition.

Table 3: Accuracy of 3D CNN based action recognition of 10 test.

| Trial | epochs | time | acc | validation acc | Evaluation acc |
|---|---|---|---|---|---|
| 1 | 50 | 49 | 89.0% | 91.0% | 91.0% |
| 2 | 50 | 49 | 86.5% | 89.6% | 88.7 % |
| 3 | 50 | 49 | 87.6% | 91.0% | 90.9% |
| 4 | 50 | 48 | 89.4% | 91.6% | 90.4% |
| 5 | 50 | 40 | 89.6% | 91.0% | 87.0% |
| 6 | 50 | 39 | 85.2% | 90.0% | 90.0% |
| 7 | 50 | 39 | 88.0% | 91.5% | 89.4% |
| 8 | 50 | 39 | 89.2% | 91.3% | 90.3% |
| 9 | 50 | 39 | 88.2% | 90.0% | 90.2% |
| 10 | 50 | 39 | 86.4% | 89.3% | 85.8% |
| Average | | | 87.9% | 90.6% | 89.4 % |

## 4.2 Results: Proposed Hybrid Model

We present the results of our proposed hybrid model that combines both the Bi-LSTM and the 3D CNN. For our experiment, we use three official train/test splits of the UCF101 dataset to keep the consistency of reporting performance on this dataset [2]. In each split, train samples are used to train and validate the model, and testing samples are used to perform prediction tasks and calculate the accuracy. Each training cycle consists of 100 epochs, and we conducted 10 rounds of this training cycle. Models can be stochastic, and therefore, the model performance cannot be concluded with one test.

Table 4: Accuracy of the proposed model for 3 splits of UCF101.

| Trial | epochs | UCF101 SPLIT 1 | UCF101 SPLIT 2 | UCF101 SPLIT 3 |
|---|---|---|---|---|
| 1 | 100 | 95.9% | 95.7% | 97.3% |
| 2 | 100 | 95.8% | 95.9% | 97.0% |
| 3 | 100 | 96.3% | 95.0% | 96.9% |
| 4 | 100 | 95.9% | 95.9% | 97.0% |
| 5 | 100 | 96.3% | 95.3% | 97.8% |
| 6 | 100 | 96.3% | 95.2% | 97.6% |
| 7 | 100 | 95.6% | 95.8% | 97.2% |
| 8 | 100 | 96.1% | 95.4% | 97.7% |
| 9 | 100 | 96.2% | 96.6% | 97.3% |
| 10 | 100 | 97.3% | 95.5% | 97.1% |
| Average | | 96.2% | 95.6% | 97.2% |

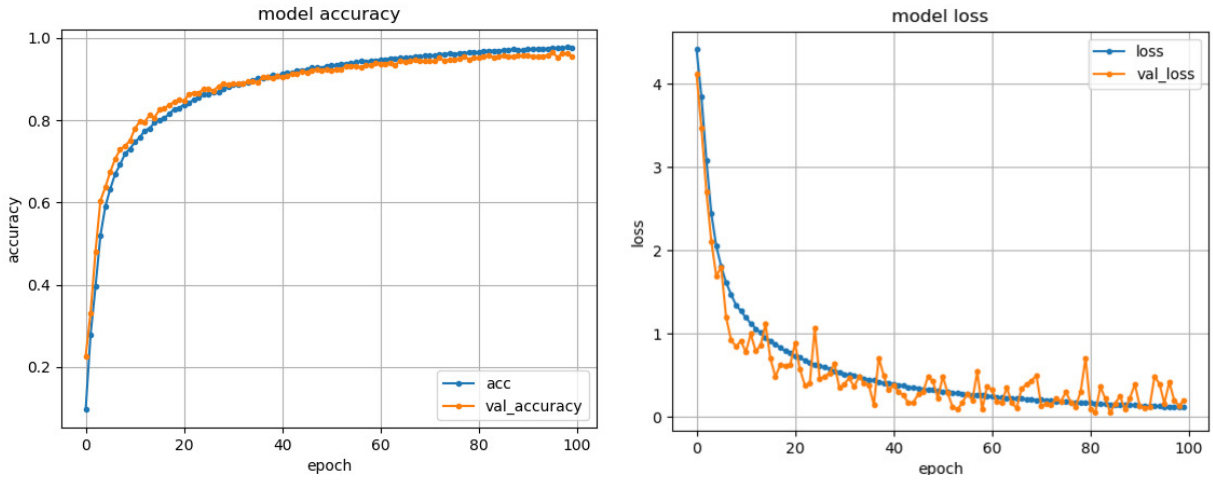


Figure 24: The accuracy and loss plot of the 3D CNN and Bi-LSTM based action recognition.

The experimental evaluation is performed by training and predicting all three train/test splits of the UCF101 dataset. The average accuracy of prediction task on all three train/test splits is 96.4%. The experiment showed consistent result for all tests performed, and the summary of all tests are tabulated in Table 4. The accuracy and loss plot for the first round training of the second train/test split of the dataset is plotted in Figure 24. Both accuracy plots showed a steady and progressive rise with an increase in epochs. The accuracy becomes stable after 70 epochs, and validation accuracy shows some fluctuations. The plot of loss against epochs also shows a gradual decrease in loss with an increase in epochs, and training loss gets saturated towards the end. Noticeably, the decrease in validation loss has shown many fluctuations.

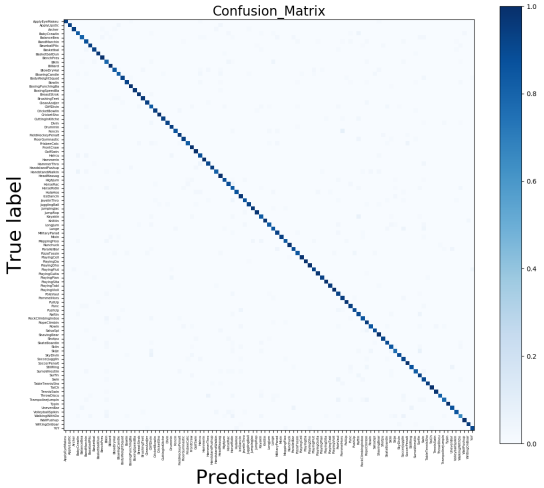


Figure 25: Confusion matrix of the 3D CNN and Bi-LSTM based action recognition.

Next, we try to analyze the model with the confusion matrix and class-wise accuracy report. The actions such as Basketball, Playing Dhol, Table Tennis Shot, Biking, Cricket Shot, Golf Swing and Hammering performed the best in classification. These actions are strongly associated with an object related to it. The appearance details and motion of these objects permit ease less identification of actions from videos. 87 Actions were classified with more than 95% accuracy, and 100 actions perform with class-wise accuracy of more than 90%. Walking with a dog, Balance Beam, Band March, and Handstand Walking are the 4 classes that showed the lowest classification accuracy; moreover, these are the worst-performing ones. All these actions are very similar to walking, which creates confusion for the model in identifying them and resulting in low accuracy for all these actions.

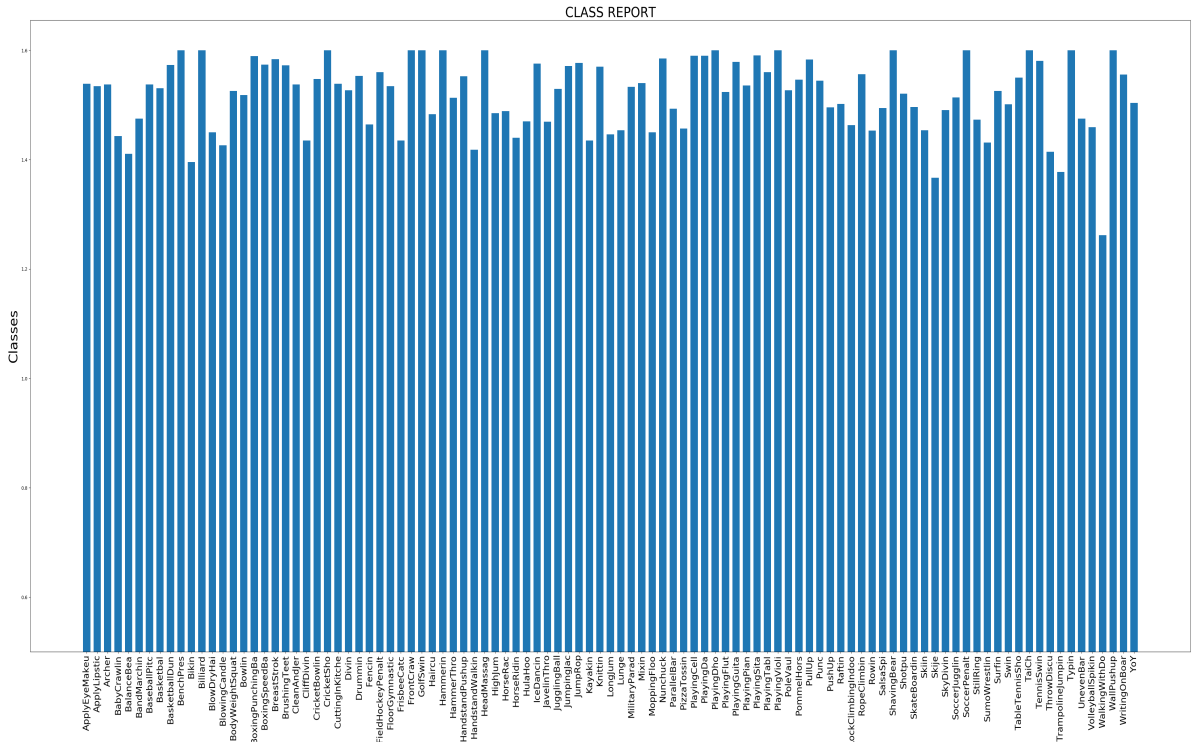


Figure 26: The class-wise accuracy report of the 3D CNN and Bi-LSTM based action recognition.

Lastly, we try to visualize the prediction of the model. For which we have to take a sample frame from each video sample and predict the action class. Both ground truth and predicted action class are inscribed on the frames. A total of 80 different video samples are displayed, in which 58 of them are correctly classified. The actions like Shaving beard, Skiing, Rowing, Diving and Walking with dog have been misclassified in more than once prediction.

Figure 27: The frames and predicted action classes of the action recognition model based on 3D CNN and Bi-LSTM based action recognition.

## 4.3 Evaluation of the Proposed Model

In this section, we evaluate the proposed model for action recognition against the performance of the baseline models. The first two baseline models, i.e., the spatial CNN based and the temporal CNN based approaches, show very low accuracy in classifying actions, which highlights the influence of both spatial and temporal information in an action classification task. Since these models extract either the spatial information or temporal information from video, they cannot accurately classify the actions. We argue that models cannot showcase a good performance without extracting information from both the spatial and the temporal dimensions. Therefore, these two models are not used for further evaluation of the proposed model.

The two-stream model that used average fusion to calculate class scores performed with 45.16% accuracy on video samples. This model had a different approach from all the other models in that they are not trained directly, but the model tries to classify actions by combining class scores of two separate trained model weights. Although the model had achieved an accuracy better than the two previous models, this accuracy does not meet the adequate performance for a good classifier. Comparing the complexity, the two-stream model is more complex than the proposed model in training the model, preprocessing the videos for classification and performing classification on videos. Notably, the factor that resulted in low accuracy is that the spatial and temporal features are processed in two different networks that don't examine the relationship between them.

Table 5: Evaluation of the proposed model against baseline models.

| Models | Accuracy | Time for Each Epochs | Complexity |
|---|---|---|---|
| Spatial CNN | 10.47% | 290 Sec | HIGH |
| Temporal CNN | 38.45% | 422 Sec | HIGH |
| Two-Stream CNN | 45.16% | 950 Sec | HIGH |
| CNN-LSTM Model | 60.87% | 10 Sec | LOW |
| 3D CNN | 89.41% | 42 Sec | LOW |
| Proposed Model | 96.4% | 30 sec | MEDIUM |

The CNN-LSTM based model was able to classify the actions with an accuracy of 60.87%. This accuracy is almost 36% less compared to the accuracy of our proposed model. This model has a similar algorithm as our proposed model, but here CNN can only extract the only spatial features, while LSTM computes the temporal information. This model proves that the spatial information and temporal information obtained from a few video frames are insufficient to classify the actions efficiently. In this case, the classification has been affected by the poor calculation of temporal information from intermittent frames, and the local temporal feature has not been calculated in this case. This also affirms the relevance of local spatial-temporal features and global temporal features to classify the actions used in our proposed method.

The 3D CNN-based model is the only baseline model that has shown good efficiency in classifying the actions from videos with an accuracy of 89.4%. This accuracy is 7% less than the proposed model. The complexity of the 3D CNN is less compared with the complexity of the proposed model. The 3D CNN-based model is a non-hybrid model that consists of only one deep learning algorithm, which is 3D CNN, and this reduces complexity.

The proposed model, a combination of the 3D CNN and Bi-LSTM, has showcased the best accuracy (96.4%) compared to all other baseline models. The proposed model executes in less time than all other models except CNN-LSTM based model. The model is slightly more complex than the complexity of CNN-LSTM based model and 3D CNN based model. From all these comparisons tabulated in Table 5, we presume that the

proposed model will be the most eligible to use in all applications that require action recognition.

## 4.4 Comparison of Proposed Model with State-Of-The-Art Results

This section compares the results of the proposed model with different action recognition state-of-the-art approaches for the UCF101 dataset. We have proved that the proposed model performs better than all evaluation models for a deeper investigation; we try to compare the results of the proposed model with various state-of-the-art approaches. Table 6 shows the accuracy of different state-of-the-art approaches.

Table 6: The accuracies of current state-of-the-art for action recognition.

| Rank | Model | Accuracy | Year |
|---|---|---|---|
| 1 | R2+1D-BERT | 98.69% | 2020 |
| 2 | OmniSource | 98.6% | 2020 |
| 3 | LGD-3D Two-stream | 98.2% | 2019 |
| 4 | Two-stream I3D | 98.0% | 2017 |
| 5 | MARS+RGB+Flow | 97.8% | 2019 |
| 6 | HATNet | 97.8% | 2019 |
| 7 | CCS + TSN | 97.4% | 2019 |
| 8 | Multi-stream I3D | 97.2% | 2019 |
| 9 | Hidden Two-Stream | 97.1% | 2017 |
| 10 | CMA iter1-S | 96.5% | 2019 |
| 11 | STM | 96.2% | 2019 |
| 12 | MF-Net, RGB only | 96% | 2018 |
| 13 | Optical Flow Guided Feature | 96% | 2017 |
| 14 | MARS+RGB+Flow | 95.8% | 2019 |
| 15 | Prob-Distill | 95.7% | 2019 |
| | Proposed model | 96.4% | 2020 |

The R2+1D-BERT based approach has the highest accuracy with 98.69% and OmniSource approach achieves 98.6% accuracy. Our proposed model has an 2.2% lower accuracy than the best performing models. The other action recognition models that achieved better accuracy than proposed model are LGD-3D Two-stream (98.2%), Two-stream I3D (98.0%), MARS+RGB+Flow (97.8%), HATNet (97.8%), CCS + TSN (97.4%), Multi-stream I3D (97.2%), Hidden Two-Stream (97.1%), CMA iter1-S (96.5%). The proposed model has showcased 96.4% accuracy, which is better than the eleventh highest performance among all the models above.

# 5 Discussion and Conclusion

We recall the research questions stated in Chapter 1 and discuss how they have been addressed in this thesis. We finally conclude this thesis with discussion about future work.

## 5.1 Answers to the Research Questions

***1. How can we develop a model that combines 3D-CNN with Bi-LSTM for action recognition?***
In Chapter 3, we addressed this research question by proposing a hybrid model that combines 3D-CNN and Bi-LSTM. The model was constructed with three parts. The first part employs a 3D CNN model that extracts local spatio-temporal features from 25 small splits from the input video. The second part uses Bi-LSTM to extract the hidden global temporal information from 25 local spatio-temporal features. In the last part, a softmax classifier is used to identify the actions from the global temporal features. This work makes the first step towards a novel and sophisticated action classification model.

***2. Can the spatio-temporal representation learned by the proposed model predict action from videos accurately and efficiently.?***

Our answer to this question has been provided in Chapter 3, where we demonstrated how our proposed model classifies action using both local spatio-temporal features and global temporal features. The 3D CNN network is used to extract the spatial-temporal information for a given set of frames. The local spatio-temporal features will represent objects and motion information not from a single frame but a set of 10 frames. The Bi-LSTM model estimates the global temporal information by observing the pattern in 25 different sets of local spatio-temporal features from the same video. Finally, a simple softmax classifier was used to classify the actions from the global features with 96.4% average accuracy, and this denotes that the feature set is an accurate representation of the entire dataset. Every action class was classified with more than 90% accuracy, and almost 87 actions were classified with 95% accuracy. From this, we can infer that the boundaries of all action classes were well learned by the model. Finally, these results and performances substantiate that we have accurately estimated the spatio-temporal features, and the features were efficient enough to classify all actions with good performance.

***3. How can we experimentally evaluate the proposed model?***

The answer to this research question is addressed in detail in Section 4.3. The experimental evaluation included a comparison of the proposed model with baseline models in terms of their accuracy. We also considered the complexity, execution time and class-wise accuracy of these models. Comparing the accuracy of the models, the proposed model achieves around 96.4% average accuracy, whereas the two best performing baseline models, the 3D CNN approach, performs at the accuracy of 89.41% and the CNN-LSTM based model, achieves 60.87% of accuracy. Comparing the complexity of models, the proposed model has medium complexity, whereas the 3D CNN-

based and CNN-LSTM based approach has low complexity. The time to train one epoch is another metric to evaluate the performance of models. The final classification task alone takes 30 seconds to execute each epoch in the proposed model. The 3D CNN model takes around 45 seconds to test each epoch, and the CNN-LSTM model takes around 10 seconds for the final classification task. We have also already seen that the class-wise accuracy of our proposed model is better than the 3D CNN model. From all the evidence above, it is illustrated that the proposed model performs better than all the other evaluation models.

***4.How can we analyse the performance of the proposed model in comparison to other state-of-the-art models?***

In Chapter 4, Section 4.4, we compared the accuracy of the proposed model with the accuracy of the state-of-the-art models used in action recognition. The state-of-the-art models with maximum accuracy were $R2+1D-BERT$ (98.69%), *OmniSource* (98.6%), $LGD-3D\ Two-stream$ (98.2%) and $Two-stream\ I3D$ (98.0%). We have seen from our extensive experiments that the average accuracy of our proposed model is 96.4%. From our analysis, we conclude that the proposed model is the eleventh best performing model in available state-of-the-art models.

## 5.2 Conclusion and Future Work

In this thesis, we have proposed a novel action recognition model utilizing two prominent deep learning algorithms. The 3D CNN was used to obtain the local spatio-temporal features from the input videos, and the Bi-LSTM was used to learn the global temporal features from local features extracted from the 3D-CNN. From the output features of Bi-LSTM, a softmax classifier was used to accurately classify the actions. Our model, along with five other baseline models, have been implemented and validated. The experimental evaluation demonstrates that our model has an average accuracy of 96.4%, thereby outperforming all the other baseline models. Moreover, our model provides the eleventh highest average accuracy in comparison to the available state-of-the-art models for action recognition.

*Future Work:* Currently, we have evaluated our models with the UCF101 dataset. In the future, we plan to extend the performance evaluation of our model against real-world datasets of varying sizes. This would help in gaining a better understanding of the robustness of our model. Moreover, There are boundless amounts of action that human beings perform, but in this thesis, we have only considered the 101 human performed actions. We also plan to extend our model to support more action classes without compromising the performance, and also to support other data modalities such as kinetic data and skeleton-based data. Finally, in this work, due to the limitation of time, the proposed model could only be evaluated against five baseline models. In future, we plan to deploy our model on a robot to study the real-time performance and shortcomings (if any) of our proposed model.

# References


[1] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. http://www.deeplearningbook.org.

[2] K. Soomro, A. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *ArXiv*, abs/1212.0402, 2012.

[3] Yu Kong and Yun Fu. Human action recognition and prediction: A survey. 2018.

[4] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

[5] J.R. Quinlan. Induction of decision trees. *Journal of Machine Learning*, 1:81–106, 1986.

[6] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Trans. Inf. Theory*, 13:21–27, 1967.

[7] Bo Meng, X. Liu, and X. Wang. Human action recognition based on quaternion spatial-temporal convolutional neural network and lstm in rgb videos. *Multimedia Tools and Applications*, 77:26901–26918, 2018.

[8] Minghuang Ma, Haoqi Fan, and Kris M. Kitani. Going deeper into first-person activity recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1894–1903, 2016.

[9] K. Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014.

[10] Christoph Feichtenhofer, A. Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1933–1941, 2016.

[11] Yonghao Dang, Fuxing Yang, and Jianqin Yin. Dwnet: Deep-wide network for 3d action recognition. *ArXiv*, abs/1908.11036, 2020.

[12] S. Ji, W. Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35:221–231, 2013.

[13] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[14] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278–2324, 1998.

[15] Yoshua Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5 2:157–66, 1994.

[16] S. Hochreiter. Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München, 1991.

[17] D. Tran, Lubomir D. Bourdev, R. Fergus, L. Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4489–4497, 2015.

[18] Gopika Rajendran, Ojus Thomas Lee, A. Gopi, Jais Jose, and Neha Gautham. Study on machine learning and deep learning methods for human action recognition. 2020.

[19] RamezaniMohsen and YaghmaeeFarzin. A review on human action analysis in videos for retrieval applications. *Artificial Intelligence Review*, 2016.

[20] Xiaohua Zhai, Y. Peng, and J. Xiao. Cross-media retrieval by intra-media and inter-media correlation mining. *Multimedia Systems*, 19:395–406, 2012.

[21] Mikhail Bortnikov, Adil Khan, Asad Masood Khattak, and Muhammad Ahmad. Accident recognition via 3d cnns for automated traffic monitoring in smart cities. In *Science and Information Conference*, pages 256–264. Springer, 2019.

[22] Piotr Bilinski and François Brémond. New results - human action recognition in videos. 2014.

[23] A. Ciptadi, Matthew S. Goodwin, and James M. Rehg. Movement pattern histogram for action recognition and retrieval. In *ECCV*, 2014.

[24] C. Schüldt, I. Laptev, and B. Caputo. Recognizing human actions: a local svm approach. *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, 3:32–36 Vol.3, 2004.

[25] Imen Jegham, Anouar Ben Khalifa, Ihsen Alouani, and M. Mahjoub. Vision-based human action recognition: An overview and real world challenges. 2020.

[26] A. Bobick and J. Davis. The recognition of human movement using temporal templates. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23:257–267, 2001.

[27] Lena Gorelick, Moshe Blank, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29:2247–2253, 2007.

[28] B. Horn and B. Schunck. Determining optical flow. *Artif. Intell.*, 17:185–203, 1981.

[29] Deqing Sun, S. Roth, and Michael J. Black. Secrets of optical flow estimation and their principles. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2432–2439, 2010.

[30] I. Laptev and T. Lindeberg. Space-time interest points. *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 432–439 vol.1, 2003.

[31] Chris Harris and Mike Stephens. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pages 147–151, 1988.

[32] Ivan Laptev. On space-time interest points. *Int. J. Comput. Vision*, 64(2–3):107–123, September 2005.

[33] P. Dollár, V. Rabaud, G. Cottrell, and Serge J. Belongie. Behavior recognition via sparse spatio-temporal features. *2005 IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pages 65–72, 2005.

[34] I. Laptev, M. Marszalek, C. Schmid, and Benjamin Rozenfeld. Learning realistic human actions from movies. *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[35] G. Willems, T. Tuytelaars, and L. Gool. An efficient dense and scale-invariant spatio-temporal interest point detector. In *ECCV*, 2008.

[36] S. Wong and R. Cipolla. Extracting spatiotemporal interest points using global information. *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8, 2007.

[37] P. Scovanner, Saad Ali, and M. Shah. A 3-dimensional sift descriptor and its application to action recognition. In *ACM Multimedia*, 2007.

[38] Lahav Yeffet and Lior Wolf. Local trinary patterns for human action recognition. *2009 IEEE 12th International Conference on Computer Vision*, pages 492–497, 2009.

[39] H. Wang, M. M. Ullah, Alexander Kläser, I. Laptev, and C. Schmid. Evaluation of local spatio-temporal features for action recognition. In *BMVC*, 2009.

[40] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI*, 1981.

[41] Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. Technical report, International Journal of Computer Vision, 1991.

[42] J. Shi and Carlo Tomasi. Good features to track. *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[43] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt. Sequential deep learning for human action recognition. In *HBU*, 2011.

[44] S. Haykin. 1 feedforward neural networks : An introduction. 2004.

[45] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65 6:386–408, 1958.

[46] Y. LeCun, P. Haffner, L. Bottou, and Yoshua Bengio. Object recognition with gradient-based learning. In *Shape, Contour and Grouping in Computer Vision*, 1999.

[47] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. *Learning Representations by Back-Propagating Errors*, page 696–699. MIT Press, Cambridge, MA, USA, 1988.

[48] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

[49] Y. LeCun and F. F. Soulié. Modeles connexionnistes de l'apprentissage. *Intellectica*, 1987.

[50] H. Bourlard and Y. Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, 59:291–294, 2004.

[51] D. Broomhead and D. Lowe. Radial basis functions, multi-variable functional interpolation and adaptive networks. 1988.

[52] Larry R Medsker and LC Jain. Recurrent neural networks. *Design and Applications*, 5, 2001.

[53] Sepp Hochreiter and Yoshua Bengio. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. 2001.

[54] F. Gers, J. Schmidhuber, and Fred A. Cummins. Learning to forget: Continual prediction with lstm. *Neural Computation*, 12:2451–2471, 2000.

[55] V. Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

[56] B. Xu, Naiyan Wang, T. Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *ArXiv*, abs/1505.00853, 2015.

[57] Y. Sun, X. Wang, and X. Tang. Deeply learned face representations are sparse, selective, and robust. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2892–2900, 2015.

[58] Jun Han and Claudio Moraga. The influence of the sigmoid function parameters on the speed of backpropagation learning. In José Mira and Francisco Sandoval Hernández, editors, *IWANN*, volume 930 of *Lecture Notes in Computer Science*, pages 195–201. Springer, 1995.

[59] Chigozie Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall. Activation functions: Comparison of trends in practice and research for deep learning. *ArXiv*, abs/1811.03378, 2018.

[60] Y. Guo, L. Sun, Z. Zhang, and H. He. Algorithm research on improving activation function of convolutional neural networks. pages 3582–3586, 2019.

[61] Li Hui-fang. Research on quantum neural network and its applications based on tanh activation function. *Computer and Digital Engineering*, 2012.

[62] Xi Ouyang, Shuangjie Xu, Chaoyun Zhang, Pan Zhou, Y. Yang, G. Liu, and X. Li. A 3d-cnn and lstm based multi-task learning architecture for action recognition. *IEEE Access*, 7:40757–40770, 2019.

[63] L. Zhang, G. Zhu, Peiyi Shen, and Juan Song. Learning spatiotemporal features using 3dcnn and convolutional lstm for gesture recognition. *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 3120–3128, 2017.

[64] Sheeraz Arif, Jing Wang, T. Hassan, and Z. Fei. 3d-cnn-based fused feature maps with lstm applied to action recognition. *Future Internet*, 11:42, 2019.

[65] Wentao Zhu, Cuiling Lan, Junliang Xing, Wenjun Zeng, Yanghao Li, Li Shen, and X. Xie. Co-occurrence feature learning for skeleton based action recognition using regularized deep lstm networks. *ArXiv*, abs/1603.07772, 2016.

[66] A. Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *CACM*, 2017.

[67] D. Silver and K. Bennett. Guest editor's introduction: special issue on inductive transfer learning. *Machine Learning*, 73:215–220, 2008.

[68] Jun-Hwa Kim and Chee Sun Won. Action recognition in videos using pre-trained 2d convolutional neural networks. *IEEE Access*, 8:60179–60188, 2020.

[69] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[70] Laura Sevilla-Lara, Yiyi Liao, Fatma Güney, V. Jampani, A. Geiger, and Michael J. Black. On the integration of optical flow and action recognition. *ArXiv*, abs/1712.08416, 2018.

[71] Florence Simon and Ujwal Harode. Optical flow for action recognition. *International Journal for Research in Applied Science and Engineering Technology*, pages 1503–1511, 2017.

[72] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. volume 2749, pages 363–370, 06 2003.

[73] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4489–4497, 2015.

[74] Y. Jia, Evan Shelhamer, J. Donahue, S. Karayev, J. Long, Ross B. Girshick, S. Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *ArXiv*, abs/1408.5093, 2014.