Comparative study of Multimodal Human Activity Recognition (HAR) models on the UTD-MHAD dataset

Project report submitted as a part of

Summer Internship Program 2021

Submitted by:

Daksh Singh (IIIT Naya Raipur)

Ajit Singh (IIIT Guwahati)

Ashmik Harinkhede (IIIT Guwahati)

Atharva Karande (IIIT Pune)

Indian Institute of Information Technology, Pune

E-Mail: tanmoy@iiitp.ac.in, Phone: 9975727016, Website: iiitp.ac.in

Acknowledgment

First and foremost, we would like to express our wholehearted and sincere gratitude to our faculty mentor, **Dr. Tanmoy Hazra, Department of Computer Science and Engineering, Indian Institute of Information Technology, Pune** for his kind direction, assistance, and constant motivation from time to time, throughout the time period of our project completion. We thank him for providing us with an opportunity to work on such a relevant topic. We greatly admire his approach to research, problem-solving, sincerity, and hard work. We are immensely grateful to him for reviewing all our work and separate versions of our report with patience. His constant inputs throughout the project greatly shaped the present work. This project would not have been possible without his efforts and kind supervision. We owe him our profound gratitude for his support and kind supervision. Lastly, we would also like to thank the relevant authorities at Indian Institute of Information Technology, Pune for providing us with the opportunity to pursue the Summer Internship Program.

Abstract

Human Activity or Action Recognition (HAR) from video has been an important area of research in Computer Vision (CV). It has applications in surveillance systems, human-computer interactions and various other real-world applications involving human beings. As the major focus of our exploration has been skeletal and inertial data, we faced the issues which one runs into while working with and collecting sensor data. Classical approaches to the problem involve hand crafting features from the time series data based on fixed-sized windows and training machine learning models, such as ensembles of decision trees. The difficulty is that this feature engineering requires deep expertise in the field. Recently, deep learning methods such as recurrent neural networks and one-dimensional convolutional neural networks, or CNNs, have been shown to provide state-of-the-art results on challenging activity recognition tasks with little or no data feature engineering, instead using feature learning on raw data. We have performed a comparative analysis of Machine Learning classifiers and Deep Learning models on all the data modalities available in the University of Texas at Dallas Multimodal Human Action Dataset (UTD-MHAD) and, through understanding the classification task and the features in the dataset, the team has performed relevant feature engineering and successfully made use of different modelling techniques namely Random Forest, Gradient Boosted Trees, CNN, VGG16-CNN based transfer learning and ConvLSTM for performing Multimodal HAR. Best performance was given by ConvLSTM and it's architecture has been explained in section 6.5 . The findings of our study have been summarised in section 7.

Contents

Chapter 1

Introduction to Human Activity Recognition

1.1 Introduction

Human Action Recognition (HAR) from video has been an important area of research in Computer Vision (CV). It has applications in surveillance systems, human-computer interactions and various other real-world applications involving human beings. Human Action Recognition, or HAR basically encompasses analysing and then identifying different types of human activity from unknown video sequences.

The different types of human activities, as discussed in this paper[10] can be broadly classified into 4 different categories:

1. **Gestures**: A collection of movements made with hands/legs/face, etc. Eg: Raising an arm, Stretching a leg, nodding head, etc.
2. **Actions**: Collection of multiple gestures. Eg: Walking, running, etc.
3. **Interactions**: It is a collection of human actions of maximum two actors. One actor has to be a human being and the other one may be a person or an object. Eg: Hand shaking between two persons, Swinging a cricket bat, etc.
4. **Group Activities**: It is a collection of gestures, actions and interactions where the number of actors is more than two and there may be single or multiple interactive objects. Eg: People playing a football match, group meeting, etc.

1.2 Understanding of HAR

Human activity recognition (HAR) is a vast topic of study that focuses on recognizing a person's particular movement or action based on some of the sensor data. Indoors, common activities such as walking, chatting, standing, and sitting are examples of movements. Human Activity Recognition (also known as HAR) aims to understand the action that the Human performs with the surrounding environment. With the recent developments in the Deep learning field, HAR gains popularity among the community. The prediction of human action can be done with different types of data like human motions,

some sensor-based single human activity data, with the help of vision datasets, etc. Some of the Human Activity Recognition applications are Surveillance Systems, Health monitoring systems, Human-computer interaction, and many more.

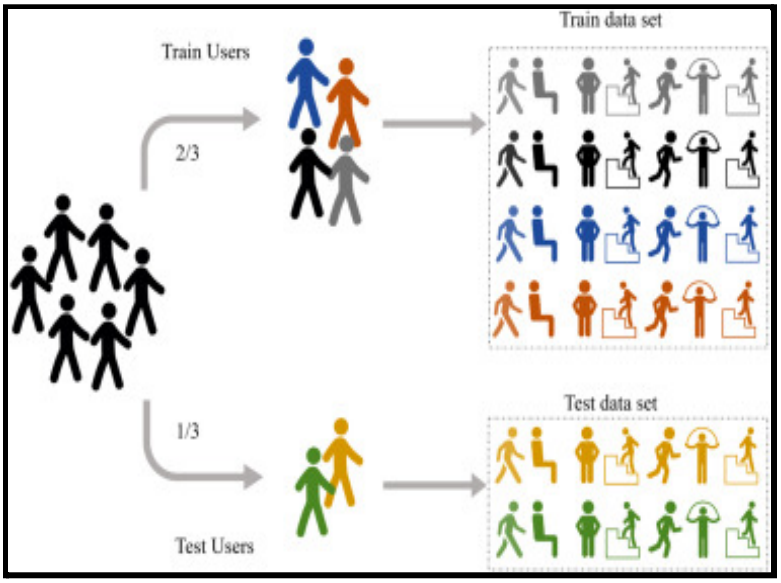

Fig 1.1: Segregation of data for Training and Testing

Source: Google AI blogs

They might also be more concentrated tasks, such as those carried out in a kitchen or on a manufacturing floor. We can record the data from the sensors with the help of some wireless technology, radar or in the form of video. Not only limited to this, data can also be calculated with the help of some hardware associated sensors such as accelerometers and gyrometers of the mobile devices. Sensor data for activity identification has traditionally been difficult and expensive to acquire, requiring specialized hardware[10]. But nowadays Smartphones and other fitness bands and hardware related to health monitoring gadgets are now inexpensive and widely available. As a result, sensor data from these devices is less expensive to acquire and more frequent, making it a more widely investigated variant of general activity recognition.

The challenge is to anticipate activity based on a snapshot which can be in the form of video, image, raw points or any other type of data points of sensor data. HAR is usually considered as a Multivariate or Univariate classification

task. It's a difficult challenge since there are no apparent or straightforward ways to link recorded sensor data that is given as input for the task to specific human actions, and each subject may execute an activity differently, resulting in differences in recorded sensor data[2].

We have implemented the vision-based HAR model since we are developing the system for Human Object Interaction for Robot Learning. Since the activity recognition task would be done on the input videos so here we have to take care of Spatio-temporal data, that's why 3D CNNs are used in all the models that are designed for the activity recognition. The computation requirements of these types of models are very high. But still, there are a lot of issues like variation and similarity between the two classes, the interaction between humans and objects are not marked, only the activity is identified, inability to predict the long contextual meaning of the human interactions, and so on. A more advanced problem that the deep learning community is now tackling is Human Object Interaction (HOI). Some of the standard datasets which are primarily used for activity recognition are HMDB51, UCF-101, Kinetics. Models such as Inflated Two-Stream 3D ConvNet (I3D), Temporal 3D ConvNet (T3D), TSN, ResNet50 + LSTM are known to give good results for activity recognition.

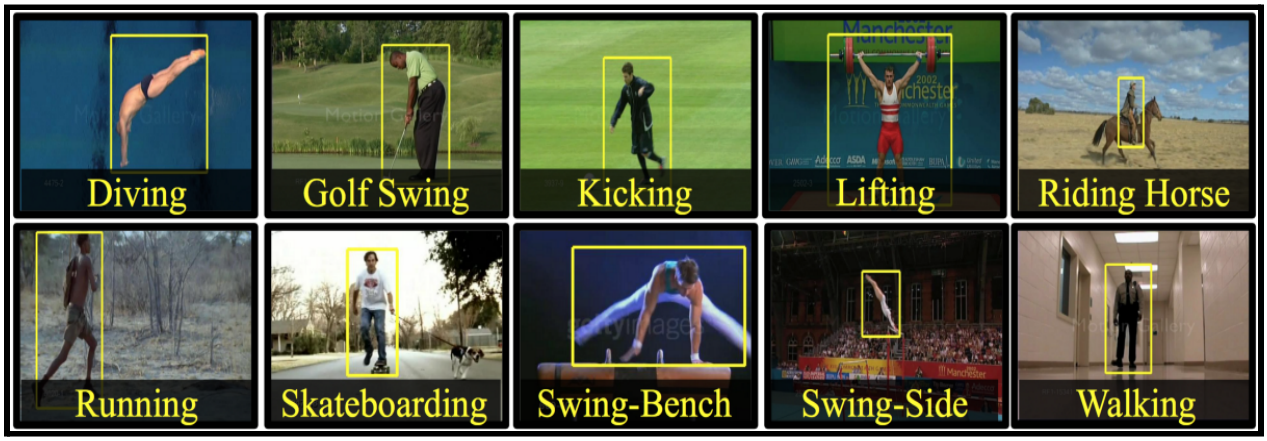

Fig 1.2: Examples of Human Activity Recognition

Source: Medium

1.3 Object Detection

Object Detection is a new developing field that usually works with the identification and location of objects within the image of certain classes. We

can interpret the location of the object in different ways, this involves building a bounding box around the object or labelling each pixel in the object-containing image. It is considered as one of the key technologies behind the Autonomous Driving Car. It is not only limited to Self-driving cars, but they are also used in Surveillance systems, Sports analysis, and many more real-life applications[3].

Two different types of networks are defined for object detection, first one is two-stage networks in which first the region or subset of the input image is identified where the target object might be present and then the second stage of the network is used to classify the object. Some examples of two-stage defined networks are Mask-R CNN, Faster RCNN, and RCNN (Region-based Convolutional Neural Network). Single-stage networks on the other hand use only a Single network for object detection, they use the Anchor box concept for the regional identification of the target object then some Suppression algorithms for the selection of only useful regions. One of the most common and advanced models that use this approach is YOLO (You Only Look Once), it localizes and detects the object in real-time with a single pass of the input image making it an efficient algorithm.

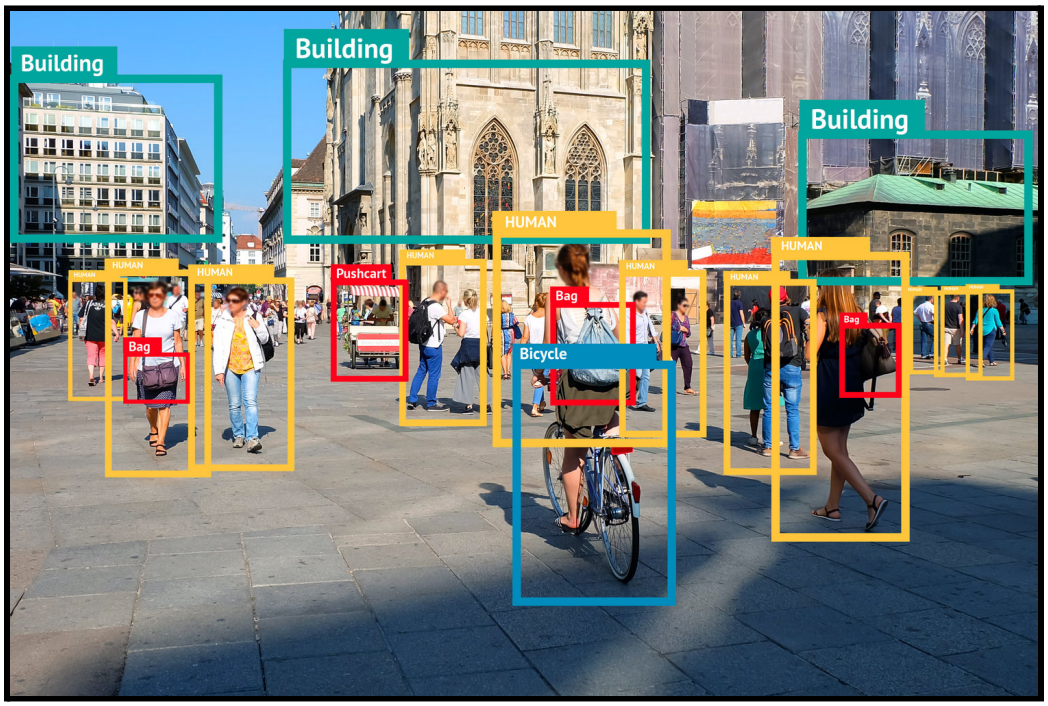

Fig 1.3: Example of Object Detection

Source: Foundation AI

1.4 Pose Estimation

The representation of a person's orientation in graphical format is called a Human Pose Skeleton. It is a set of coordinates that if joined describe the pose of a person. Each coordinate in the graph is called a key point. And a connection between 2 valid coordinates is called a limb or pair. Not all connections give a valid pair[9]. An example of a human skeleton data extracted from a picture is shown in the figure below.

There are 2 different kinds of Deep Learning approaches to Pose Detection:

Top-Down Approach : The simplest approach is using a person detector first, which is followed by estimating the parts. Then calculation of pose for every person is done. This method is called the top down approach. RMPE is a famous top down method of Pose Estimation. Top down methods usually depend on the accuracy of the person detector. This is because pose estimation is performed on the region the person is usually located.

Bottom-Up Approach : The other approach for tackling the pose detection problem is by detecting all parts in the image, followed by grouping up parts belonging to distinct persons. This method is called the bottom up approach. OpenPose is a famous bottom-up approach for tackling the pose detection problem. Like other bottom up approaches, OpenPose primarily detects parts belonging to every person in the image. Then assigns parts to different individuals.

Fig 1.4: Left-keypoint format for human pose skeleton , Right-human pose skeleton computer rendered

Source: OpenPose

Chapter 2

Introduction to Deep Learning

2.1 Introduction

Deep learning is an area of machine learning that is related to the algorithm that tries to replicate or act as a human brain. It's inspired by human brain function and structure today, many applications in the automotive industry to the medical sector are powered by deep learning as many models can achieve a higher precision than the conventional models can even achieve with the aid of deep learning. This is the main technology behind many projects and applications such as self-driving cars, object detection and classification, object position, text recognition and generation, and many more. Lately, deep learning is accomplishing things that are never imagined before, even doing better than human inference for certain tasks. One big explanation for this branch's sudden growth and development is the availability of massive quantities of data. We are using technology in today's world and producing more and more data, but still, deep learning needs a lot more data. Deep learning is trained using a large amount of data and it has a basic architecture which is a neural network.

Deep learning is considered a special case of machine learning. The use of traditional algorithms first requires the relevant features to be extracted manually, then those features are used to train the model while the neural network itself extracts relevant features in deep learning, no manual intervention is required. There is no need for manually removing characteristics in the initial stages. Deep learning performs 'end-to-end learning,' meaning a network has provided raw data and will perform the appropriate task by automatically learning it and producing the desired final output. The major difference between deep learning and conventional algorithms is that deep learning requires a lot of data to perform well, and its performance improves with the increase in data, while machine learning algorithms converge with the increase in data means after achieving some performance that they do not further improve. Deep learning applications range from autonomous cars to medical devices and are used in various industries[10].

Some of the most common fields that have the greatest contribution of deep learning are:

- **Automated driving:**.Many researchers in the field of automobiles use deep learning to automatically recognize objects, such as stop signs and traffic lights, or to locate the object so that after seeing an obstacle, vehicles stop. In addition, it uses deep learning to recognize pedestrians, thus helping to reduce the number of injury cases. Special sensors and small chips only enable the use of profound learning in these tasks.

- **Aerospace and Defense:** Deep learning can be used to classify objects from satellite outputs that could be in terms of photos, heat maps, videos or something else, to locate points of interest on various planets and in space, to classify secure and dangerous places for military personnel to camp for the nation during difficult times and many more.

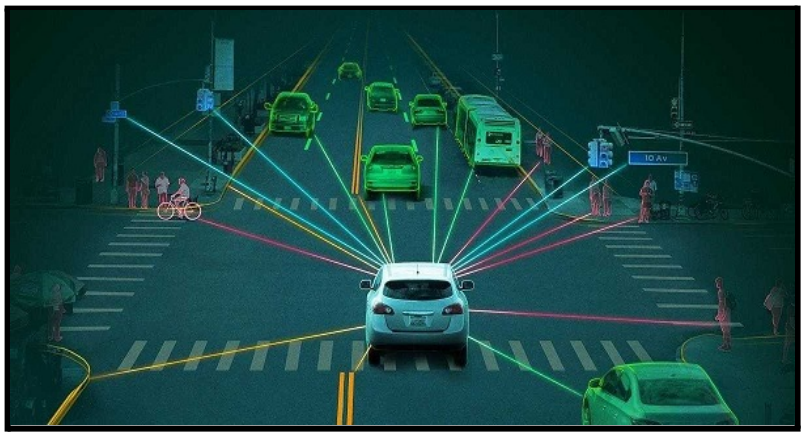

Fig 2.1: Autonomous car localization and detection

Source: Google AI blogs

- **Medical Research:** Deep learning is used to conduct many medical-related studies, nowadays even cancer cell recognition using deep learning is possible. Several teams built multiple models during the Convid-19 that can predict the existence of viruses using lung x-rays with an accuracy above 99%. For all teams, this is a great accomplishment. Deep learning is used in many areas nowadays and many industries, the medical industry is one of those sectors in which deep learning has been of great benefit. In medicine, several classification tasks are now performed using neural networks.

- **Industrial Automation:** Many machinery and other technical devices are now automated to ensure that they are maintained properly and that deep learning is used to verify whether they are functioning properly or not. Some industries have chemical hazards in use, therefore the workers should work with some difference between them and machinery. The task is easily automated with the aid of deep learning. For some goods, object identification and detection of faults is often achieved using neural network models.

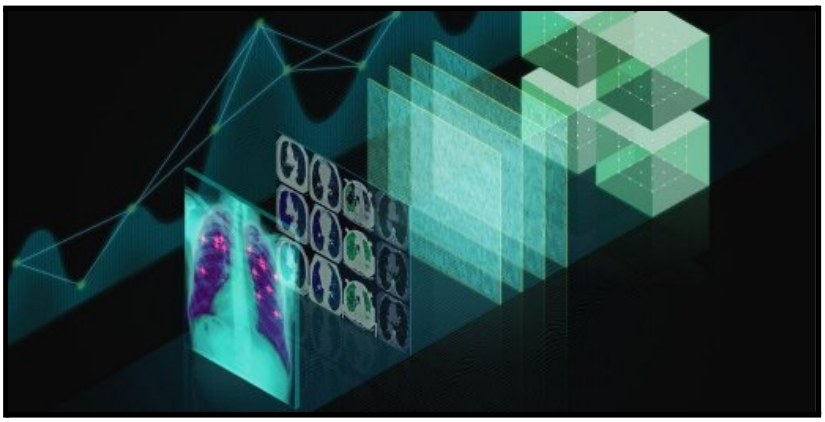

Fig 2.2: Enlargement detection in the lungs using neural networks

Source: The medical futurist

- **Electronics:** Deep learning is used in many applications for automatic auditing and speech translation. Apple's Siri, Amazon Alexa, and echo, for sure, and much more. These devices are trained using the various sequence models which are developed using the neural networks of deep learning. These devices can respond to your voice commands and help you perform certain tasks. Deep learning powers them all. Many face recognition, mobile cameras, and face-unlocking AI are all done using deep learning. Many speech recognition and generation tasks have used a heavy deep learning model, nowadays even with a limited corpus, a learner is able to use various models to create poetry like Shakespeare.

2.2 How Deep Learning works?

Most of the deep learning systems use specific units called a neural network made up of layers of neurons or perceptrons, which are also sometimes referred to as artificial neural networks or deep learning models. The term

deep refers to the model's complexity, in times when there is fewer data and computing resources are typically minimal, the model is normally only 2-3 layers deep, but with the development of computational resources, especially the advent of powerful GPUs, deep nowadays means that the model has around 100-150 layers in the model. Deep learning models are usually trained with a large amount of data which is labeled and deep neural network architectures seek to learn the characteristics from the data so that they will be able to generalize well when the data that the model has never seen before is seen without any human side intervention. Neuron or perceptron is the basic unit of deep learning architecture, which is a single unit within a single layer of neural network. As the size of the neural network increases, so does the complexity of the features the model learns will also increase.

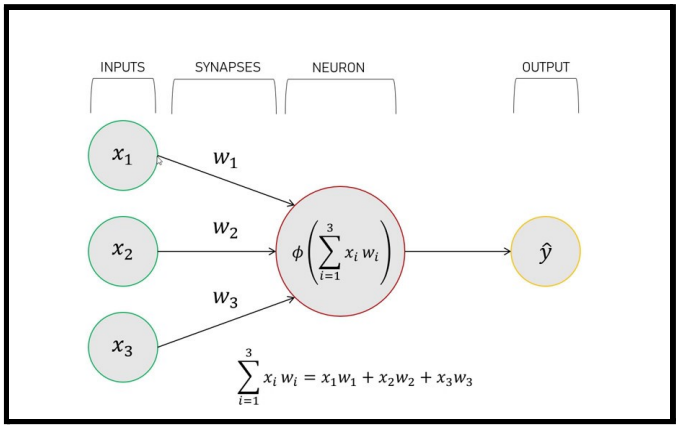

Fig 2.3: Perceptron

Source: Deeplearning.ai

The most common type of neural network is Convolutional Neural Network(CNN). They use filters and convolution to extract the features from the inputs thus removing any human pre-trained interventions. As the depth of CNN increases, they are able to learn more and more specific and special features, thereby improving the neural network 's overall precision. For example, maybe the first hidden layer may be able to detect vertical and horizontal edges and the second hidden layer may be able to detect more complex and different shapes and if we increase more hidden layers in the net then the complexity of features learned at each layer will increase. When the number of hidden layers increases, so does the efficiency and the computing resources needed. With greater depth, it will consume more and more time while training on the data.

Chapter 3

Deep Learning and Machine Learning Models

3.1 CNN (Convolutional Neural Network)

Convolutional Neural Network can be considered as the regularised form of Multilayer perceptron (MLP). CNN is the most commonly utilized neural network with the rise of the deep neural networks. It's become common in the classification and detection of images. Many models of image recognition, localization, and classification used multiple CNNs because they provide good performance when several blocks of CNN are used properly. CNN's may also be used in text recognition and generation, while it is difficult to achieve a state-of-the-art model for them using convolutional neural networks alone, they do very well in performance terms in certain tasks. CNN has been built based on the workings of visual cortex animal neurons. CNN's usual task is to take an input image, extract and learn features from it, and to predict the image under one of the categories being provided while testing using those learned features[5].

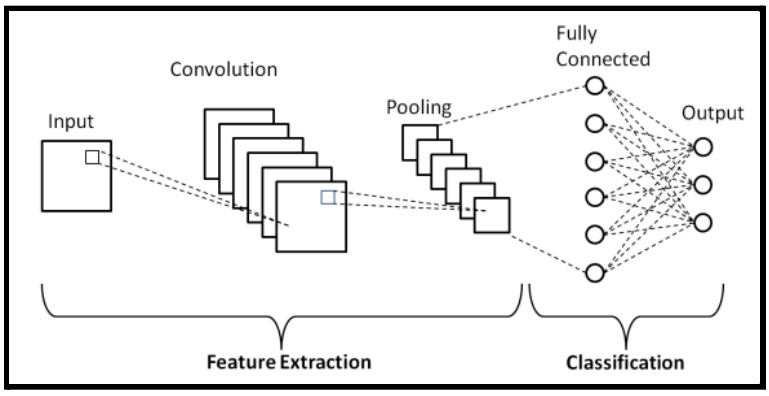

Fig 3.1: Basic Convolutional Neural Network

Source: researchgate.net

Instead of transmitting the image that is passed to the neural network as a single image, it is passed as a matrix or array that contains pixel values from 0 to 255 depending on the intensity color, and a specified number of channels, the most common being 3 that constitute red, green, and blue. The basic

concept behind the convolutional neural network is that it will be passed by an input array with the numbers, and it will generate the output in the form of probabilities defining the desired class is certain. For example, if we take an input image of a 64x64x3 dimension cat containing all three RGB channels and the neural network is trained to distinguish between horse, cat and dog, then the output may be generated like this cat with 0.80 probability, dog with 0.15 probability and 0.05 horse probability.

Since we want to output as one class only, we choose the class with the highest likelihood as our output. Similar to the human brain this works. For starters, when we look at a dog, we can quickly identify it as our brain has developed and learned from childhood that the dog is recognized by four paws, this type of face and other characteristics.

3.2 Classic Convolutional Neural Network

The common contents of a classic Convolution Neural Network are:

- Convolutional Layer
- Activation operation following every Convolutional layer.
- A Pooling layer, usually Max Pool layer is used but sometimes even the Average Pool layer is also used.
- Sometimes a Flatten layer is used but most of the time it won't be used in basic CNN.
- Finally a Fully Connected output layer.

Input Layer

An image is fed to the network before resizing it to an optimal size. One of CNN's key properties is that they take a fixed-size input image, so if the dataset on which training is to take place includes images of various sizes then we need to reshape them until moving them to the convolutional neural network.

Convolutional Layer

Convolutional is the first layer in the network to extract the functions from the input image. Convolution learns the features by using different filters of the same size to establish a pixel relation.

Such convolutional layers have filters or kernels lying over the image and extracting the properties from the images. Normally the kernel slides over the

input image, and the kernel values are multiplied element-wise with the input image's original pixel values. The kernel values will determine the features that will be extracted from the image. When doing the multiplication of the element-wise operation, usually the filter size of 3 or 5 is used. But it can be updated to match the need. The multiplications are summarized to produce a single value for that portion of the kernel and the input image, then the kernel slides over to the next portion of the image data. How the kernel will slide depends on the strides used in the convolutional layer, if a stride of 1 is chosen then the kernel will slide to the adjacent pixel or if the stride of two is chosen the kernel will slide skipping the alternate pixels. Thus the output of the convolutional layer which is also called a feature map will have a size that is less than the input image size.

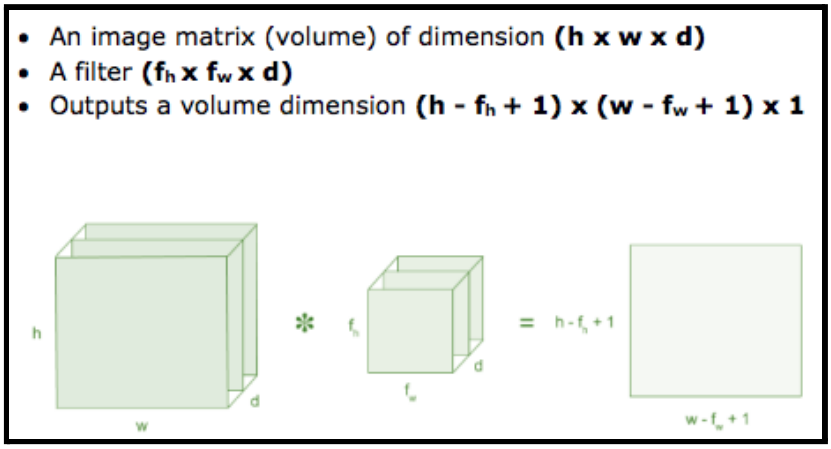

Fig 3.2: Working of convolutional filters over the input image

Source: medium blog

The performance dimension generated by the convolution layer depends on the number of features in the first layer that we want to evaluate. To obtain the number of features, a number of filters are chosen, so if we use 10 filters or kernels then the output will have a dimension of (l,b,10). These initial stage filters are used to extract some basic features from the image. As the number of convolution layers increases, so does the complexity of the function learned and extracted from that. The output is transferred to activation functions after the convolution to introduce the nonlinearity in the Convnet. This helps the neural network work on negative or any different type of values. Some common activation functions are tanh, sigmoid, and ReLU. ReLU stands for the Rectified Linear Unit for a non-linear operation. The output is ***f(x) = max(0,x).***

Pooling layer

Pooling layers help to reduce the number of parameters in case the image input is too high. They are often called upsampling or downsampling, as they can reduce and increase the number of parameters a model needs to learn to predict well. It allows a great deal to reduce training time and computational complexity even though the essential features remain. Max and Average Pooling are some of the common pooling techniques deployed in Convnet. The largest value of filter that overlaps from the rectified feature map is selected in the Max pooling algorithm, so if we use a 2x2 filter then the output size will be half the input size, whereas if we use the Average pooling, the output will be considered to be the average of all the four-pixel values in that region.

Fully Connected Layer

FC layer is the final layer before the output, after flattening the output received from the previous layers, fully connected layers take a single vector as the input and it will combine all the features that are extracted and learn by the previous layers to provide the output a single vector containing the probability of the classes that are desired in the output. The output vector size depends on how many classes are required for classification.

Output Layer

The network's final layer that is fed with the output vector of completely connected layers typically uses activation functions such as sigmoid or softmax to produce the final output of the desired class. These activations can only generate a single class output which is expected to be the standard after training.

3.3 LSTM (Long Short Term Memory)

Recurrent Neural Network is a special type of neural network which is very strong and robust in nature due to the important property of having memory. It is one of the most promising algorithms that is used today. Recurrent neural networks, like many other deep learning techniques, are relatively new. They were first developed in the 1980s, but it wasn't until recently that we realized their full potential. RNNs have risen to prominence as a result of increased processing power, huge quantities of data we now have to deal with. RNNs can recall key details about the input data network receives

thanks to their internal memory, allowing them to anticipate what will happen next with great accuracy.

That's why they're the ideal algorithm for voice, time series, text, audio, video, weather,financial data, and many other types of sequential data. In comparison to other algorithms, RNN's can build a far deeper knowledge of a sequence and the context it holds. The RNN's are usually used when spatial information of the data is not that much significantly more important than that of the temporal data. In today's life we used the RNN's enabled applications such as Siri, Alexa and Google Translate and Assistant daily. So with the comfortable advancements of the hardware it becomes easy to target and solve high end problems related to the temporal dynamics[7].

3.4 How Recurrent Neural Networks (RNN) work?

To understand the working of RNNs we have to understand what is the difference between RNN and other neural networks. Sequential data is simply ordered data in which related items appear one after the other. Some of the common examples include Financial data or a DNA sequence. Perhaps the most common form of sequential data is time series data, which is just a collection of data points in chronological order.

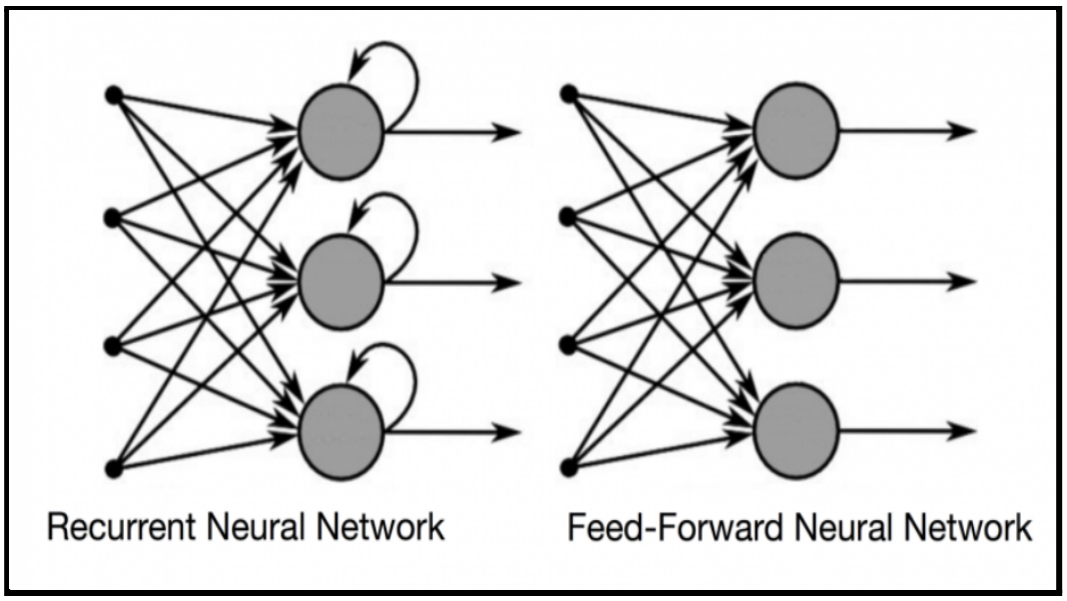

Fig 3.3: RNN vs Feed-Forward Neural Network

Source: builtin blogs

The information about the features of the input data moves from the initial input layer to the last final output layer by passing through some of the hidden layers in between. One of the important things about feed-forward neural networks is that once the information pases a node it never comes back to that node twice, so the information moves straight from input point to end point in single forward direction as the name suggests. That is why Feedforward nets have no recollection of the information they receive from the input and they are poor predictors of what will happen next with the features. Also it has no concept of time order since it only examines the current input. Except for its training, it has no recollection of what transpired in the past. The information in an RNN cycles via a loop. When it makes a judgment, it takes into account the current input as well as what it has learnt from prior inputs. Because of the memory that the RNNs have they are able to reconcile the characteristics of previous input. It generates output, replicates it and then provides it back to the network. This goes on the loop.

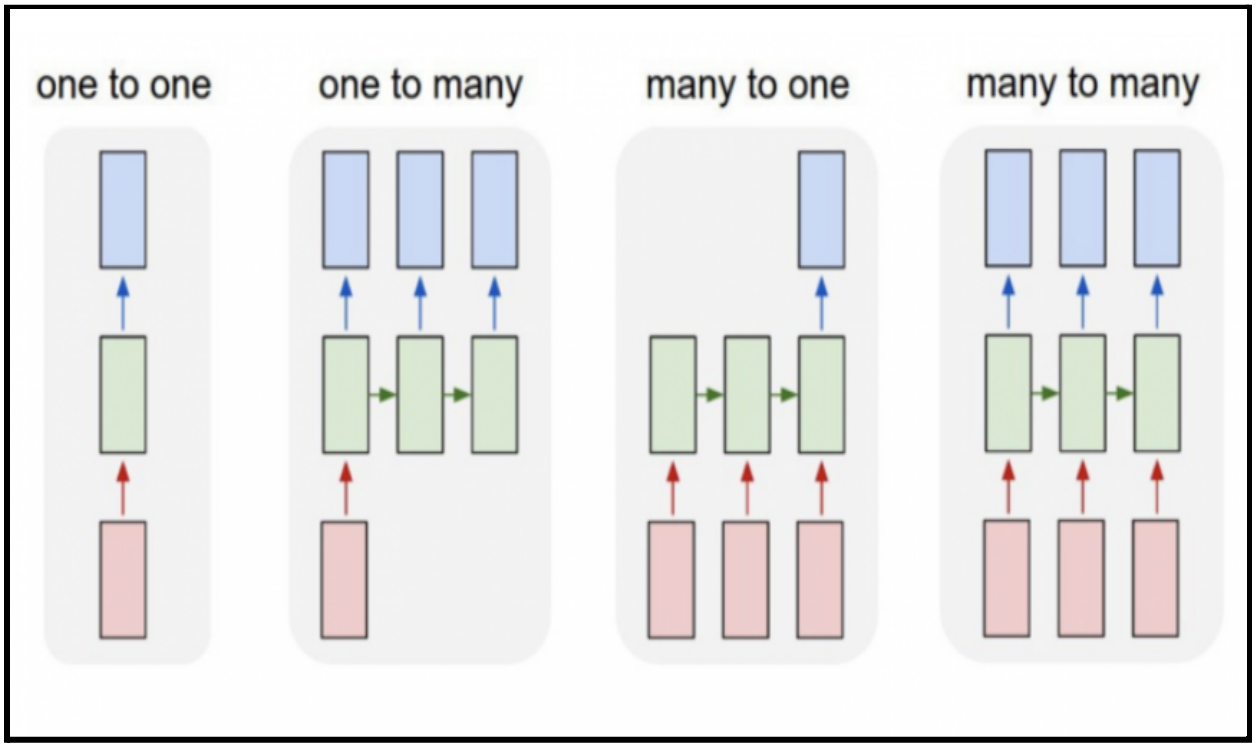

Fig 3.4: Different input and output mappings

Source: builtin blogs

As a result, there are usually two different inputs to an RNN, one is the present input and the other input is the output of the recent past. This is significant because the data sequence provides critical information about what will

happen next, because of this property only the Recurrent Neural Network (RNN) can perform tasks that other algorithms cannot. Weights are applied to both the current and prior inputs by RNNs. In addition, the network will adjust the weights over time via gradient descent and backpropagation through time. RNNs are not limited to one to one mapping as done in the feed neural networks, instead they are capable of different mappings which are one to many, many to one and many to many usually done for translation purposes.

The gradient understanding is important to properly understand the issues with the basic RNN's. We can consider gradient as a partial derivative of the input provided. Another way to present it is as a function slope's. The faster model is learning, steeper slope it has and resulting higher gradient values. If at any point the slope of the function reaches to zero, we can say the model will cease learning. It measures the change that happens in the weights during training to the changes that happen in the error. The RNN's faces problems of vanishing gradients and exploding gradients.

Exploding gradients occur when the algorithm gives the weights an absurdly high priority for no apparent reason. Fortunately, squashing or truncating these gradients is a simple solution to this problem. Vanishing gradient problem generally occurs when the gradient values are too tiny and because of this the model ceases learning further or to produce sufficient results it will take a long time. This issue was very much serious as it was not able to be addressed properly until the introduction of LSTM.

3.5 Long Short Term Memory (LSTM)

Long short-term memory (LSTMs) networks are a type of recurrent neural network that increases the memory capacity. As a result, it is ideally adapted to learning from significant events separated by extended periods of time. The layers of an RNN, sometimes referred to as an LSTM network, are built using the units of an LSTM. RNNs can recall inputs for a long time because of LSTMs. This is due to the fact that LSTMs store information in the memory that is very much similar to that of a computer. The LSTM has the ability to read, write, and erase data from its node memory[7]. This memory may be thought of as a gated cell, where gated indicating that the cell determines whether or not there is a need to store or erase information (i.e., whether or not to open the gates) based on the significance it gives to the data. Weights, which are generally learnt by the algorithm, are used to give importance. This

basically implies that it learns the importance of information over time to decide which information is not important or vice-versa.

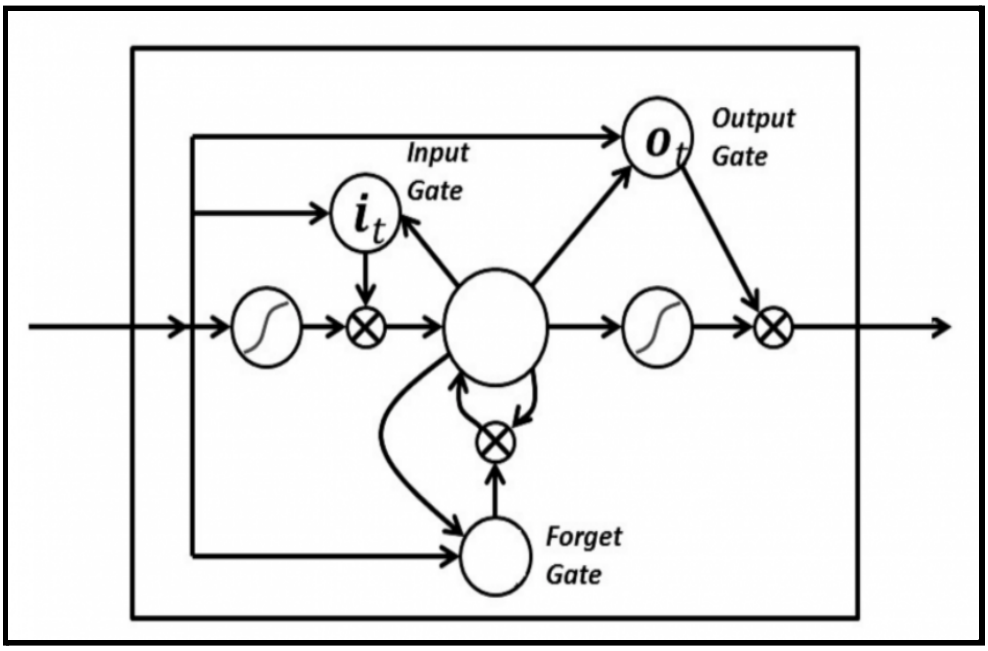

Fig 3.5: LSTM unit cell

Source: builtin blogs

There are generally three gates in an LSTM uint: input gate, forget gate, and output gate or impact gate. These gates decide whether fresh input should be allowed (input gate), whether it should be deleted because it isn't significant (forget gate), or whether it should have an influence on the output according to the current timestep (impact gate) (output gate). These gates are usually analog and generally they are in the form of sigmoids function thus making them range between zero to one. This enables them to do backpropagation. The issue of vanishing gradients of RNN is properly handled by LSTM because it tries to keep the gradients steep enough, resulting in a quick training period and excellent accuracy.

3.6 Random Forest

Random forest is versatile and one of the easy-to-use machine learning methods that, in most cases, delivers excellent results even without hyper-parameter tweaking. Because of its simplicity and versatility. It is often used a lot when compared to other Machine Learning algorithms because it can be used for both types of problems that are classification and regression.

Random forest comes under supervised learning algorithms. It creates a "forest" out of an ensemble of decision trees, which are generally trained using the "bagging" approach. The bagging method's basic premise is that combining several learning models improves the final outcome. The hyperparameters of a random forest are quite similar to those of a decision tree or a bagging classifier. This makes it possible that there is no need to mix some decision trees with the braggin classifier, instead we can use a class of the random forest classifier[9].

During the growth of the trees, the random forest adds more unpredictability to the model. When splitting a node, it looks for the best feature from a random group of characteristics rather than the most significant feature. As a result, there is a lot of variety, which leads to a better model. As a result, in a random forest, the method for splitting a node only considers a random subset of the characteristics. The random properties of trees can be increased by applying some random thresholds for every feature instead of looking for the best feasible thresholds.

3.7 How Random Forests work?

Because the random forest mixes many trees to forecast the dataset's class, some decision trees may correctly predict the output class whereas some of the trees predict the incorrect output class. However, when all of the trees are combined, the right result is predicted. There are two basic assumptions while using the Random Forest algorithm. The features variables that are going to be extracted from the dataset should hold some real values, since only then a classifier can predict the correct results instead of some guesses. And additionally the prediction of each tree should have low correlations only. Random forest has two main phases, in the first phase to construct a random forest we merge N decision trees and in the last phase we make predictions around each decision tree which later combines using some method to form the final prediction.

In a random forest it is very easy to measure the importance of a feature in the prediction. This helps a lot in the feature selection and engineering process since we can pick which feature will be important for the prediction. This impacts significantly, since with greater number of features there's a chance of overfitting always which can be avoided with this. The main distinction is that "deep" decision trees may be prone to overfitting. Random

forest usually prevents this by generating random subsets of the characteristics or the different important features and utilizing those selections to create smaller trees. It then joins the subtrees together. It's essential to note that this doesn't always work, and it also slows down the calculation depending mainly on the number of trees the random forest generates.

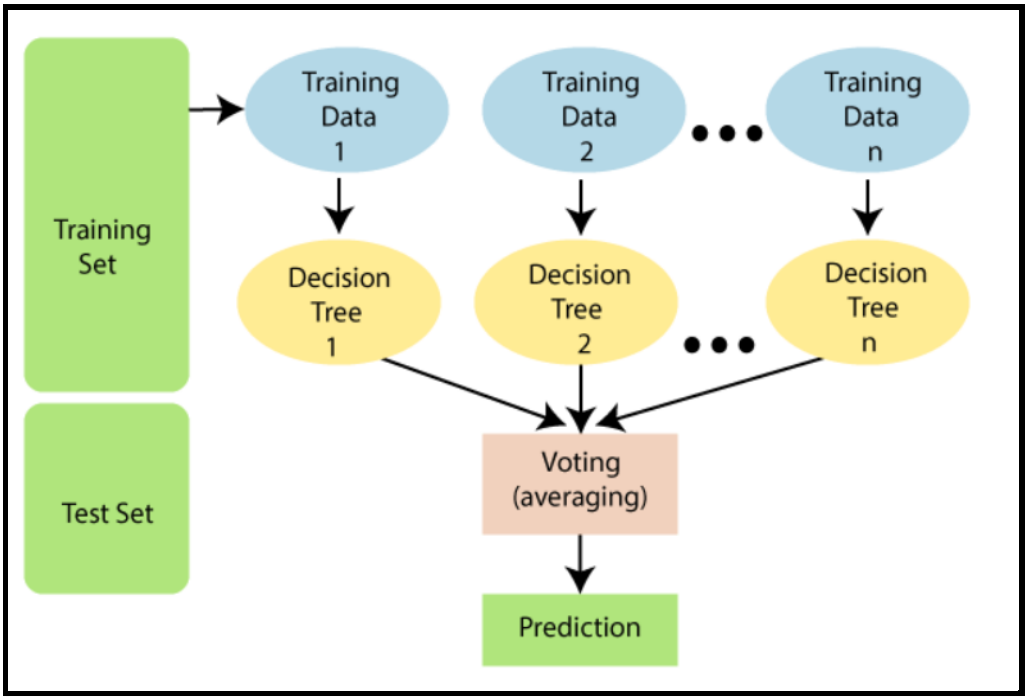

Fig 3.6: Working of Random Forest

Source: Tutorialspoint

Random forest's flexibility is one of its most appealing features.It can easily be used for different kinds of tasks specially regression and classification, and the relative significance it assigns to the input characteristics can be easily viewed. Random forest is also a useful method since with the default hyperparameters only it employs frequently yield accurate predictions. Understanding these hyperparameters is simple, and there aren't many of them to begin with. Overfitting is one of the most common difficulties in machine learning, however owing to the random forest classifier, this is seldom a problem. Since with the large datasets or large number of features increases the chances of overfitting it is suggested to avoid a large number of features, thus with the increase in number of decision trees and understanding of the importance features, random forests models usually

don’t overfit. The primary drawback of this is that it might become too sluggish and useless for real-time forecasts if there are too many trees. In general, these algorithms are quick to learn yet take a long time to make predictions after they've been trained. More trees are required for a more accurate forecast, resulting in a slower model.
The random forest technique is generally fast enough in most of the real-world applications, however there are times when run-time performance is critical and other approaches are preferable. And more importantly if we want to understand the descriptive nature of the data then other methods might be preferable since random forests are not descriptive they are used for predictive modeling. The random forest method is utilized in a variety of applications, including finance, stock trading, medical, and e-commerce. It is used in the healthcare industry to find the right mix of components in the testing medicine and to evaluate a medical history of a patient to find illnesses. In e-commerce, a random forest is used to assess whether or not a customer would enjoy the product. It can also be used like a Recommendation System.

3.8 XGBoost

XGBoost is a method of ensemble learning. It may not always be enough to depend on the findings of a single machine learning model. Ensemble learning is a method for combining the predictive capacity of several learners in a systematic way. The end result of this is one single model that combines the outputs of many models. The foundation learners, or models that make up the ensemble, might be from the same learning algorithm or from separate learning algorithms. Bagging and boosting are two types of ensemble learners that are commonly employed. Though these two approaches may be used to a variety of statistical models, decision trees have been the most popular. XGBoost is a high-speed and high-performance implementation of gradient boosted decision trees. The full form of XGBoost is eXtreme Gradient Boosting[12].

Extreme gradient boosting, often known as XGBoost, is a well-known gradient boosting technique(ensemble) that improves the performance and speed of tree-based (sequential decision trees) machine learning algorithms. Tianqi Chen invented XGBoost, which was first maintained by the Distributed (Deep) Machine Learning Community (DMLC). Nowadays it gains a lot of

popularity because the winning solutions are provided with the help of this algorithm in structured and tabular data.

3.9 How does XGBoost work?

Decision trees are built sequentially in this technique. In XGBoost, weights are very essential. All of the independent variables are given weights, which are subsequently put into the decision tree so that the decision tree can predict outcomes. The weight of variables that the tree predicted incorrectly is raised, and the variables are then put into the second decision tree and so on. This process keeps going on. These separate classifiers/predictors are then combined to create a more powerful and precise model. It may be used to solve issues including user defined issues of predictions, regression and classification and some predefined ranking.

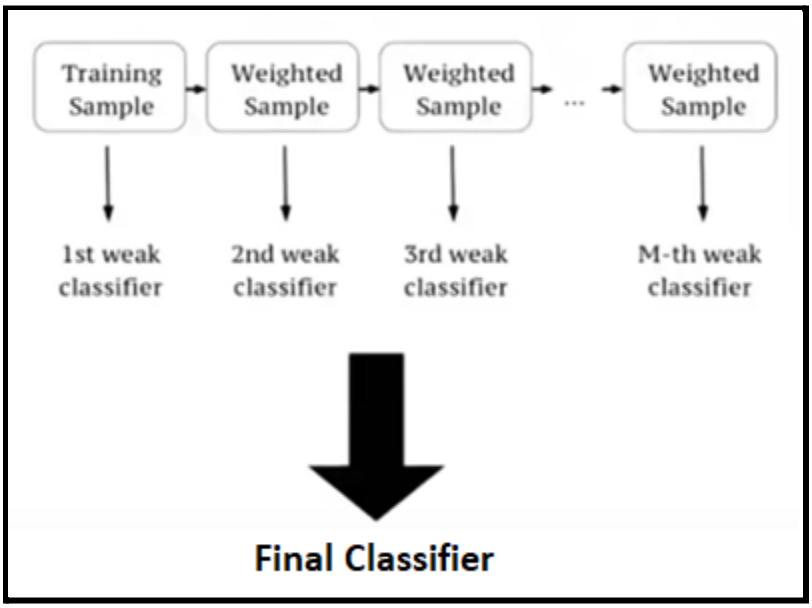

Fig 3.7: Working of XGBoost

Source: GeeksForGeeks

Bagging or boosting aggregation can assist any learner to minimize variation. The basic learners of the bagging approach are many decision trees that are created simultaneously. These learners are taught using data that has been sampled using replacement. The aggregate output from all of the learners is used to make the final forecast. The trees are created progressively in boosting, with each succeeding tree aiming to minimize the preceding tree's mistakes. Each tree builds on the knowledge of its ancestors and corrects any remaining mistakes. So this will result in the following tree that is present in

the sequence learning from an updated set of residuals. One of the major disadvantages of the boosting is that the base learners in the boosting are considerably weak learners having a larger bias and the predictive powers of them are slightly better than the random guessing. The boosting approach is successful only because these weak learners offer some important crucial information for prediction, so combining those results will generate a good result equivalent to the strong learner. This will help to reduce the bias and variance of the final strong learner.

The most effective Scalable Tree Boosting Method has been proved to be XGboost. It has demonstrated remarkable results in a variety of applications, including detection of motion, stock sales forecasting, virus classification, consumer behavior analysis, and a lot more. With efficient handling of the data and memory, the system operates far quicker on a single computer than any other approach. The optimization approaches used by the algorithm increase performance and hence offer speed while utilizing the fewest resources possible.

3.10 VGG16

VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper “Very Deep Convolutional Networks for Large-Scale Image Recognition”. It was used to win ILSVR (ImageNet Large Scale Visual Recognition Challenge) competition in 2014. This model makes the improvement over AlexNet. AlexNet have large kernel-sized filters (11 in the first convolutional layer and 5 in a second convolutional layer). Improvement is done by replacing large kernel-sized filters with multiple 3×3 kernel-sized filters one after another. It was one of the famous models submitted to ILSVRC-2014. In this challenge, each team competes on two tasks. The first is to detect objects within an image coming from 200 classes (Object Localization task).

The second is to classify images, each labeled with one of 1000 categories (Image classification task). The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. It was trained for weeks and was using NVIDIA Titan Black GPUs. It is considered to be one of the excellent vision model architectures till date. The ImageNet dataset contains images of the fixed size of 224*224 and has RGB channels. Thus, the model has a tensor of (224, 224, 3) as input.

Architecture of VGG16 is as follows, The input to the network is an image of dimension (224, 224, 3). The first two layers have 64 channels of 3*3 filter size with the same padding. Then max pool layer of stride of (2,2), two convolutional layers with filter size 256 and 3x3. The same layering of the max pool, convolutional, convolutional is repeated. Then the model has 2 sets of three convolutional layers and one max pool layer. Each layer has 512 filters of 3x3 filter size with the same padding. The obtained image is then passed through 2 convolutional layers. The most unique thing about this model is instead of using a large number of hyperparameters (7x7 in ZF-Net and 11x11 in AlexNet), it uses a convolutional layer of 3x3 filter with stride 1 and 2x2 filter for max pool layer. Some layers also use a 1x1 filter for the manipulation of the number of input channels.

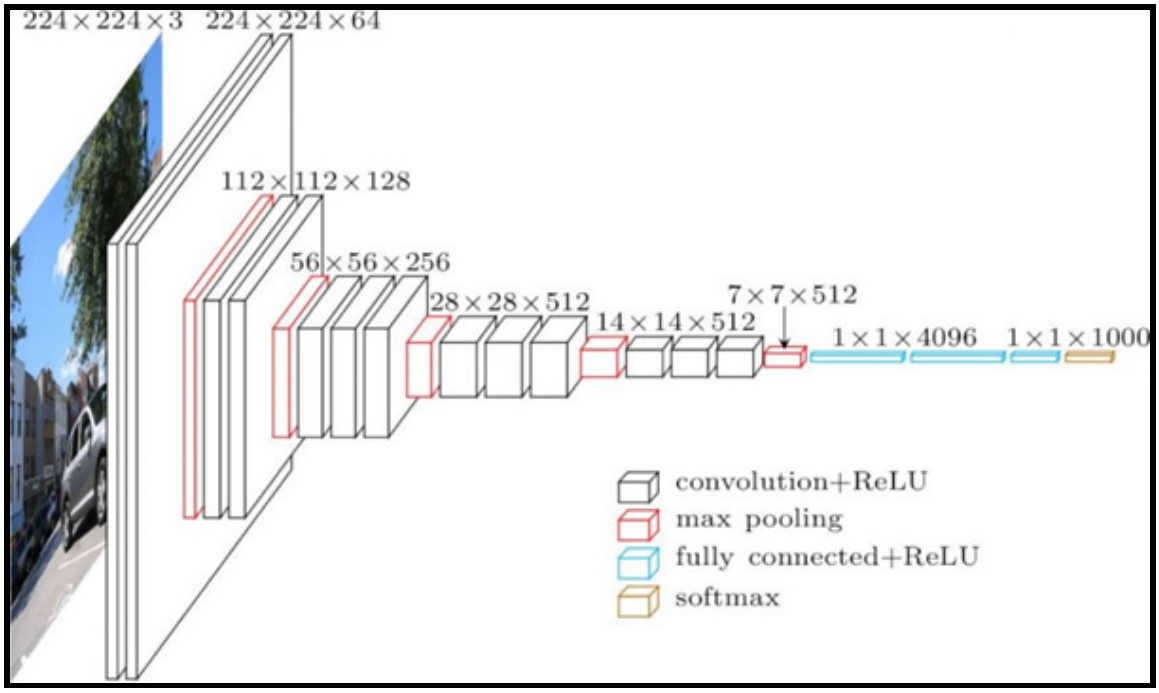

Fig 3.8: VGG16 Architecture

Source: GeeksForGeeks

To prevent spatial features of the image, the padding of 1 – pixel is added after each convolutional layer. Then a feature map of size (7,7,512) is obtained which is flattened to make it (1, 25088) feature vector. Then there are 3 fully connected layers containing, the first layer which takes input from the last feature vector and outputs a vector of size (1,4096), the second layer takes this as an input and produces the output of same size (1, 4096). But the third layer outputs 1000 channels (1 channel for each class from 1000 classes if ILSVRC challenge). This output is passed to the softmax layer which normalizes the

classification vector. From the classification vector, the top-5 categories are taken for evaluation.

All hidden layers use the ReLU activation function. It is computationally efficient because it results faster learning and decreases the likelihood of vanishing gradient problem. The challenges of VGG 16 include its slow training. The original model was trained for 2-3 weeks. Also, the size of trained ImageNet weights is about 530 MB. Due to this size, it takes quite a lot of disk space and bandwidth which makes this model inefficient sometimes in terms of complexity.

Chapter 4

Literature survey and Related work

4.1 Skeleton-based action recognition with hierarchical spatial reasoning and temporal stack learning network[4]

In this paper they have proposed a hierarchical spatial reasoning and temporal stack learning also known as HSR-TSL to perform human activity recognition over 4-5 different datasets. They tried to explore discriminative and distinct temporal and spatial features for the same. They used a residual hierarchical graph neural net, so that they can capture basically two different spatial features which are structural body level information between the each part and intra spatial points of each part. To get the skeleton sequences in detailed temporal dynamics they have used multiple skip-clips Long Short Term Memory Network (LSTMs). For the loss function they developed a clip-based incrementing loss which is also known as CIloss function so that they can measure it over to effectively optimize the model.

They stated that the Graph Neural Network (GNN) is a sophisticated model that was proposed as an extension of recursive neural networks to cope with a more general class of graphs. They focused on skeleton-based action recognition. They achieved a 94.4% accuracy on the dataset UTD-MHAD. They suggested that the efficient coupling of spatial information into temporal representations is an essential topic that should be investigated further in future research. They suggested that in contrast to RGB data, 3D skeleton data may describe the body structure using a collection of 3D coordinate locations that are unaffected by backdrop clutter, lighting changes, or appearance fluctuation. As a result, 3D skeleton data may readily be used to acquire more effective and discriminative representations of human activity.

4.2 An efficient end-to-end deep learning architecture for activity classification[1]

In this paper they have proposed an optimized approach which is end-to-end that can easily classify and describe the human action in the input videos. They first sample the RGB videos to the frame sequences then from these frame sequences they extract convolutional features with the help of

common pre-trained model Inception-v3. They claimed that their proposed model will be an efficient end-to-end approach and it will provide an improvement in accuracy with low computational cost. They downsampled the actions that are present in the RGB video to 229*229 frames of sequences which they have used to fine tune the pre-trained Inception-v3 model. To extract the features from these frame sequences they have removed the ending fully-connected layer of the Inception-v3 neural network. This provided them the features that they wanted for the classification.

So to get the classification result they have passed these features to four-layer Long Term Short Memory (LSTM) recurrent neural network (RNN) with each layer containing 1000 hidden units. According to them, to get network parameters, pretrained CNN networks are trained on a difficult dataset. To save computation time, transfer learning uses these common pre-trained parameters as an initialization or a fixed feature extractor for the representation and classification job. They stated that they have achieved an accuracy of 98.4% and 95.5% on public datasets UTD-MHAD HS and UTD-MHAD SS respectively.

4.3 Human action recognition based on kinematic similarity in real time[8]

In this paper they have proposed a simple, fast and powerful method to recognize the activity based on human kinematic similarity. They have used a very high level, mathematical and complex approach based on the action descriptors. Usually three action descriptors are used which consist of angular velocity, joint position and angular acceleration, with the help of these they try to eliminate the normalization problem which is usually complex in the activity recognition since these descriptors are unique for unique action and individual. To account for each pose in the sequence of human action they used the angular joint descriptors with the help of a sliding time window of short duration only, approximately 5 frames around the current frame.

Now they have used three different modified K-nearest neighbor algorithms so that from one they can get the confidence score for each training frame, another one to provide the label to each frame descriptor and last one for the classification of the human action. They suggested that we can address the single frames that are given as input by estimating the frame's time label. This approach can be very efficient in the real time. This method will provide

good performance on the datasets that are segmented irregularly. They also provided a preprocessing method with the help of which it is easy to detect each and every stage of human action with the help of basic statistics. They put each frame in the sequence which is extracted from the action as independent to provide it to a classifier to estimate the frame's time label, thus providing real-time action recognition.

4.4 An enhanced method for human action recognition[11]

In this paper they provided a simple, fast and efficient method for human recognition with the help of Support Vector Machine (SVM). They designed the model which works on detecting the interest points only with the help of Scale Invariant Feature Transform (SIFT) from each and every frame of the input video. They stated that the usual flow based approach where optical flow computation is generally used to assign a motion of the human activity, it is sensitive to the external interferences and noise, thus real motion cannot be known. On the other hand, spatio-temporal methods treat the HAR problem in 3D, somewhat similar to object recognition problems. In these methods features are extracted from 3D volume thus increasing the overall cost of computation for the whole model which makes them undesirable for real world applications which usually require real-time processing.

They have described a fine-tuning step which is used to control the number of interesting points among the amount of details. They have used Bag of Video Words along with some modified normalization techniques which they state that helped in improving the performance of the model. At last for the classification purpose they have used a multi-class Linear Support Vector Machine. The suggested method is divided into four stages: finding interesting points among the details and containing them, descripting features of these points, codebook construction, and in the last classification. The KTH and Weizmann datasets were used in the experiments. The findings show that their technique surpasses most current approaches, with a KTH accuracy of 97.89 percent and a Weizmann accuracy of 96.66 percent.

4.5 A comprehensive survey of human action recognition with spatiotemporal interest point (STIP) detector[17]

This paper presents a comprehensive review on STIP-based methods for human action recognition. The paper also summarises related public datasets useful for comparing performance of various techniques. Unlike classical

methods for motion analysis (optic flow and feature tracking), STIP based detectors are extensively discussed in this review work. A STIP-based detector captures 'interest points' from a video in the spatiotemporal domain. In the context of HAR, A corner/isolated point of an image (video frame) is an interest point (point of maximum/minimum gradients/intensities).

Methodologies of an STIP-based detector encompass providing input video, detection of STIPs (via Dense feature detectors like Hessian detector, V-FAST detector and Sparse feature detectors like Harris3D, Cuboid detector), feature extraction (via Local descriptors like Cuboid descriptor, N-jet and Global descriptors like HOG3D, Histogram of Oriented Gradient - HOG/ Histogram of Optical Flow - HOF), vocabulary building (via Bag-Of-Words based model or State Space Based Model (Dilated Block Adversarial Network - DBAN, Hidden Markov Model - HMM)) and then classification (via classifiers such as the K-nearest neighbours - KNN, Support Vector Machines - SVM). The paper discusses performance of the STIP-based detectors on benchmark datasets such as KTH, Weizmann, UCF action, etc. which in essence are examples of controlled environments - with no or very less influence of illumination changes, occlusions, etc. Hence it is a major drawback when using these results to gain an exact insight into the performance of STIP-based detectors in the real world.

The paper then summarises various STIP-based detectors in usage and compares their performance on 2 benchmark datasets, Weizmann and KTH. Also, it discussed about 9 benchmark datasets for motion analysis and concluded their advantages and drawbacks. Performance of 10 STIP-based detectors onthe KTH dataset was thoroughly discussed. Out of which, the most accurate was the detector proposed by Chakraborty et Al (2012). It achieved a recognition accuracy of 96.35%. It provided 100% recognition results on the Weizmann dataset. Their STIP detector used a Sparse detector for detecting Interest points, a local feature descriptor (N-jet), Bag-Of-Words based vocabulary builder and used a Supervised classification method. They basically followed a greedy approach to motion analysis and their proposed system is robust against occlusion and a jumbled background.

4.6 Human action recognition using STIP-based detector on RGB data[17]

This paper presents a method to perform HAR using Spatiotemporal interest points for detection of important changes in the video frames. The UTD-MHAD complex dataset is used for model training and evaluation. Then, extraction of appearance and motion features of these interest points is done using the Histogram of Oriented Gradient (HOG) and Histogram of Optical Flow (HOF) descriptors. Finally, classification is performed by matching SVM by the Bag-Of-Word (BOW) of the space-time interest point descriptor to give the label of each video sequence. The training process can be summarised as providing action representation with a spatiotemporal bag of features and then training an SVM on it.

Then, model and SVM parameters are reformed gradually. For testing, again we perform action representation with a spatiotemporal bag of features. Then we run our trained SVM classifier and calculate the recognition accuracy. In the algorithm, they have used the RGB colour sequence video from the multimodal UTD-MHAD dataset to perform HAR. A train-test split of 60%-40% is done. The recognition accuracy varied from one action to another. The overall recognition accuracy of this method on the UTD-MHAD dataset is 70.37% and the recognition rate of this method (STIP-BOW-SVM using RGB Data) is 67.37%, an increase from a previously described method, in which the authors performed DMM-CRC (Depth Motion Maps used to extract features from the kinetic camera data of the UTD-MHAD dataset) using depth data and attained a recognition rate of 66.1%.

4.7 Combining CNN streams of RGB-D and skeletal data for human activity recognition[18]

This paper is an attempt to combine individual CNN streams of RGB, Depth and skeletal data at the decision level. It also proposes a novel skeleton data processing method, which can act as an individual classifier to recognize human activity. Proposed work is a 5 stream CNN in which the first stream is based on RGB data(creating Motion History Image(MHI)), the second, third and fourth stream are based on Depth data(created Depth Motion Maps(DMM) using front, side and top views), and the fifth stream is based on skeletal data by using a new approach of creating skeleton images from joint skeleton sequences.

A comparative analysis of pretrained models of large architecture(VGG-16) over pretrained models of small architecture(VGG-F) is also included.

Unsupervised methods for combining individual stream classifiers such as majority voting, Borda Count, average method and weighted product method have also been explored. Their main contribution most relevant to our work is the experimental justification that combining individual classifiers at the decision level improves accuracy. An accuracy of 95.11% and 94.60% has been obtained using VGG-16 and VGG-F respectively on the UTD-MHAD dataset. They have also implemented proposed architecture on CAD-60 and SBU Kinect dataset and attained accuracy of 92.5% and 96.67% respectively.

4.8 Interpretable 3D Human Action Analysis with Temporal CNN[19]

RGB data leads to loss of spatio temporal information as to perform HAR using RGB videos, complex human motion in 3D euclidean space is projected onto a series of 2D images. This paper focuses on the fact that encoding humans as a 3D skeleton yields both a discriminative and a robust representation for the purpose of human action recognition. They have leveraged the recent progress of powerful human pose estimation methods from RGB/RGBD data and incorporated it into their architecture. Original Temporal Convolutional Network(TCN) follows encoder decoder design. Proposed work has adapted the encoded portion of the TCN for action recognition.

The network is built from stacked units of 1-dimensional convolution followed by ReLU activation function. The 1-dimensional convolution is across the temporal domain. The model takes as input frame-wise skeleton features concatenated temporally across the entire video sequence. In this Residual TCN architecture except the first convolution layer, the model consists of stacked residual units. While making decisions they are using global average pooling followed by softmax scores. However, this point-based description scheme may lead to information loss, secondly, the existing human body skeleton extraction methods are still sensitive to pose and imaging viewpoint variation and thus they may fail in certain cases.

4.9 Real time human action recognition using triggered frame extraction and a typical CNN heuristic[20]

In this paper a two fold framework has been proposed for the frame extraction and recognition of human action from a video input. The two important phases in this architecture are 1) Frame extraction phase and 2)

CNN heuristic phase. In the frame extraction phase, instead of taking into account the entire range of frames in a video they have applied an approach based on the Fuzzy inference system (FIS) that extracted frames only when there are likely indications of action events. This makes the process a lightweight one and leads to the reduction of computational overhead. In the CNN heuristic phase, the extracted set of frames obtained from level-1 are fed to a typical CNN.

The algorithm employs the standard 2D convolution operations throughout the convolution layers. There are three convolution layers employed in this algorithm. Each convolution set of operations in a specific layer is followed by a pooling strategy. In this modified pooling strategy, the feature matrices which contain more than 75% of NULL sparsity are omitted from the pooling process. The main reason behind this customization is that when they are supplying the frames, due to the method followed for frame extraction the feature vectors contain a huge density of NULL values. Evaluation results for the proposed scheme on the HMDB51 dataset(one of the benchmark datasets for Human Action recognition) indicates overall rate of accuracies of 96.5% and 98% for frame extraction and action recognition respectively.

4.10 Region-sequence based six-stream CNN features for general and fine-grained human action recognition in videos[21]

This paper targets the problems of General Human Action Recognition and Fine-grained human action recognition. General human actions include right arm swipe, boxing, raising hand, waving hand. Examples of fine grained human actions include eating from a packet of chips, squeezing an object, washing hands etc. Their work is mainly focused on improving fine-grained action discrimination but it can be extended to primitive human actions too. Their method first estimates human pose and human parts positions in video sequences. Then they utilize a Convolutional Neural Network (CNN) to process each such image patch. Instead of using the output one-dimension feature from the full-connection layer, they utilize the outputs of the pooling layer of CNN structure, which is of a higher dimension and hence contains more spatial information.

Then the high dimension of the pooling features is reduced by encoding, to generate the final human action descriptors for classification. This method reduces feature dimension while also effectively combining appearance and

motion information into a single framework. They carried out empirical experiments using two publicly available datasets sub-JHMDB, MPII Cooking dataset. However, in video sequences human movement is typically continuous. Through observation it has been found out that some adjacent frames are similar while others have significantly different human position and pose. This is, different frames contribute different amounts of information for human fine-grained action recognition in videos. In this case, the calculation of similar frames in video may lead to redundant information and increase the amount of calculation.

4.11 Action recognition for depth video using multi-view dynamic images[23]

This paper uses Dynamic images for action recognition. Dynamic images compress a video or video clip into a single image thus maintaining rich motion information. A dynamic image encodes temporal data such as RGB or optical flow videos by using the concept of 'rank pooling'. When a linear ranking machine is used, the resulting representation is in the form of an image, which is called dynamic because it summarizes the video dynamics in addition to appearance. This is a powerful idea because it allows to convert any video to an image so that existing CNN models pre-trained for the analysis of still images can be immediately extended to videos. Due to the success of dynamic imaging in RGB video, this study aims to extend it to the depth domain.

By rotating the virtual camera within the 3D space the raw depth is densely projected with respect to different virtual imaging viewpoints. Subsequently, dynamic images are extracted from the obtained multi-view depth videos and multi-view dynamic images are thus constructed from these images. A novel CNN model is then proposed to perform feature learning on multi-view dynamic images. Particularly, the dynamic images from different views share the same convolutional layers but correspond to different fully connected layers. Accordingly, it is proposed that the raw depth video be densely projected with respect to different virtual imaging viewpoints by rotating a virtual camera around specific instances within the 3D observation space.

This procedure is derived from the intrinsic 3D imaging characteristics of depth data, which cannot be achieved by RGB data. Subsequently, dynamic images are extracted from the obtained multi-view depth videos, and thus

multi-view dynamic images are constructed for action characterization. During the training phase, the fully connected layers of different viewpoints will be iteratively tuned, whereas the shared convolutional layers are consistently trained. In particular, in this learning model, each imaging viewpoint corresponds to a sub-CNN model. All the sub-CNN models with the same structure can be trained independently in an end-to-end manner, but share the same convolutional layers. State of the art results were obtained on the NTU RGB-D and Northwestern - UCLA datasets.

4.12 3D Behaviour Recognition Based on Multi-Modal Deep Space-Time Learning[24]

In this paper, they proposed a dual-stream 3D space-time convolutional neural network action recognition framework. Their framework mainly consists of three parts: feature learning based on 3D spatial-temporal convolutional neural networks, human skeleton representation, and the integration of classification results. During the learning phase, a CNN is used to extract the 3D space-time features from the original depth video sequence using the soft-max function. They used a stochastic gradient descent algorithm to train the neural network. These features and SVM classification results are fused to achieve behaviour recognition. They train an SVM classifier with the features from the depth image sequence, the space-time convolution feature of the depth motion map, and the Fourier feature of the 3D human skeleton. During the test phase, corresponding features extracted from each action in the video are fed to the trained model, and the classification probability of each action category is estimated. Finally, the probabilities of the corresponding actions obtained with three models are fused according to the linear combination.

To fully exploit and utilize the discriminatory space-time features from different perspectives, they proposed a fusion-based action recognition framework which automatically learns advanced features directly from the original depth sequence and depth motion maps, and seamlessly integrates them with the human skeleton manifold representation. Their fusion-based framework combines global structure information, motion information, and detailed information of the action together and computational time for fusion step is in the order of tens of milliseconds, which is relatively low. For their framework, they used the UTD-MHAD dataset, UTKinect-Action3D dataset

and MSR-Action 3D dataset. They categorized data from these 3 datasets into 3 action sets (AS1, AS2, AS3) depending on the movement of body parts.

Using only CNN results obtained from depth map sequence the classification accuracy obtained on UTD-MHAD, UTKinect-Action 3d, MSR-Action 3D was 85.7%, 68.89%, 82.3% respectively. But while using their proposed fusion framework the accuracy improves to 95.34%, 97.29%, 94.15% respectively. This gives an idea about the efficacy of their framework. The drawback of their proposed method is that their proposed method could only work in batch mode, which means the depth video sequence has to be collected before sending it to the network. In the future, they are planning to extend their work for online action recognition applications.

4.13 Deep Convolutional Neural Networks for Human Action Recognition Using Depth Maps and Postures[25]

In this paper, they proposed a method for human action recognition from depth images and posture data using convolutional neural networks (CNN). They used two input descriptors for action representation. One is Depth Motion Image (DMI) for accumulating depth images of action and the other is Moving Join Descriptor (MJD) for the motion of body joints over time. For recognition from depth map and posture, they used CNN with 3 input channels and trained each channel with different input. In their CNN model, for each convolutional layer they used “Network in Network'' structure which is based on convolutional filters with 1x1 size with a larger number of filters than the previous layer. During the training phase using 1x1 convolution without increasing the depth size improves the accuracy without a noticeable influence on the computation time. This leads to good accuracy. For the training of the model, they used a Multinomial Logistic Loss function with a stochastic gradient descent algorithm to update the weights during the training process.

They noticed that for the output vector obtained from the soft-max layer, in most cases, the maximum value of the vector corresponds to the correct action. However, for some test samples, the maximum value doesn’t represent the correct action. The correct action may correspond to a probability value lower than the maximum value. To improve the prediction accuracy of data samples that generate the wrong classification, they fused the output of all 3 channels of CNN. During testing, they found that the ‘Prod’

operation is performing better in the fusion of output. Since their approach uses 3 channel CNN, the computational power required is high, but it produces good accuracy. They claim that the posture data descriptor (MJD) influences greatly on the whole recognition process, providing features to support the front views of the depth maps representation. Their proposed approach offers many possibilities on how to use the data and the model with the fusion operations. For their model, they used MSRAction3D, UTD-MHAD, MAD datasets and achieved an accuracy of 94.51%, 88.14%, 91.86% respectively.

4.14 CNN based and DTW (Dynamic Time Wrapping) features for human activity recognition on depth maps[26]

In this paper, they presented a new algorithm for human action recognition on raw depth maps. Their model makes use of the “One Against All'' method. This means, for each class, they train a separate CNN to extract class-specific features representing person shapes in raw depth maps. Then for each person shape in each depth map, the model will determine Dynamic Time Wrapping (DTW) features. Finally, each class-specific feature vector is concatenated with the DTW feature vector. For each action, they train a multiclass classifier that predicts the probability distribution of class labels. From the ensemble of classifiers, the model will select a number of classifiers with the largest number of correctly predicted classes. Action recognition is performed by a soft voting ensemble, which averages class distributions determined by selected best classifiers. To embed action features, they used Siamese Neural Network. It is a special kind of network that weights while working in tandem on two different input vectors to compute the similarity of the inputs by extracting and comparing feature vectors. (Also called a twin neural network).

SNN performs well for one-shot image recognition and can produce good results in a relatively short time. They used Nesterov Accelerated Gradient (Nesterov momentum) to train the network in 1000 iterations, with momentum set to 0.9, dropout equal to 0.5, learning rate equal to 0.001, L1 parameter set to 0.001. They experimentally demonstrated that the proposed class-specific features and DTW-based features have considerable discriminative power. Their method of concatenation of action features led to a remarkable gain in classification performance. Their algorithm outperformed the many available skeleton-based algorithms. In this paper,

they experimentally concluded that despite the limited number of training data, i.e., action sequences, it is possible to diminish overfitting/ underfitting and to learn features with highly discriminative power. Their model also has less computation power need and performs well for given data. For their model, they used MSR-Action3D and UTD-MHAD dataset and achieved an accuracy of 88.14%.

4.15 Deep Learning-based Action Recognition using Skeleton Joints Mapping[27]

In this paper, they proposed an end-to-end skeleton joints mapping of action for generating Spatio-temporal image (dynamic image) Which is fed to the deep convolutional neural network which is developed to perform the classification among the action classes. For classification, they introduced a modified version of the AlexNet. First, the mapping of the skeleton joints is done along the temporal direction and then discriminated using the DCNN for deciding the final class. Initially, the model generates dynamic images for all actions by using data of mapping between different joints from the neighboring frames. This generated Spatio-temporal images along with three different views (XY, YZ, ZX) are fed to CNN to extract meaningful features, which are then fused together and action is determined. This fusion is done by joining the line between the joints in neighboring frames.

Then modified AlexNet is used to discriminate among the action classes. The modification of AlexNet is done by integrating one additional block containing convolution, ReLU, pooling, and adjusting the size of input and output layers. They used skeleton data from the UTD-MHAD dataset and achieved an accuracy of 95.76% on the XY plane, 92.48% on the YZ plane, 94.10% ZX plane. And since the final CNN is using all 3 planes and working in 3D, it achieved an accuracy of 97.45%. They also concluded that Most of the human actions are more classifiable when mapping is done along XY and ZX planes rather than YZ planes.

Chapter 5

Understanding the UTD-MHAD Dataset

5.1 Introduction

University of Texas at Dallas Multimodal Human Action Dataset (UTD-MHAD)[22] is a publicly available multimodal human action dataset employing a Kinect depth camera and a wearable inertial sensor. It consists of four temporally synchronized data modalities. These modalities include RGB videos, depth videos, skeleton positions and inertial signals for a comprehensive set of 27 human actions[22]. This database can be used to study fusion approaches that involve using both depth camera and inertial sensor data. It has several advantages over existing Multimodal Human Action datasets.

Wearable inertial sensor on right wrist		
1	*right arm swipe to the left*	*(swipe_left)*
2	*right arm swipe to the right*	*(swipe_right)*
3	*right hand wave*	*(wave)*
4	*two hand front clap*	*(clap)*
5	*right arm throw*	*(throw)*
6	*cross arms in the chest*	*(arm_cross)*
7	*basketball shoot*	*(basketball_shoot)*
8	*right hand draw x*	*(draw_x)*
9	*right hand draw circle (clockwise)*	*(draw_circle_CW)*
10	*right hand draw circle (counter clockwise)*	*(draw_circle_CCW)*
11	*draw triangle*	*(draw_triangle)*
12	*bowling (right hand)*	*(bowling)*
13	*front boxing*	*(boxing)*
14	*baseball swing from right*	*(baseball_swing)*
15	*tennis right hand forehand swing*	*(tennis_swing)*
16	*arm curl (two arms)*	*(arm_curl)*
17	*tennis serve*	*(tennis_serve)*
18	*two hand push*	*(push)*
19	*right hand knock on door*	*(knock)*
20	*right hand catch an object*	*(catch)*
21	*right hand pick up and throw*	*(pickup_throw)*
Wearable inertial sensor on right thigh		
22	*jogging in place*	*(jog)*
23	*walking in place*	*(walk)*
24	*sit to stand*	*(sit2stand)*
25	*stand to sit*	*(stand2sit)*
26	*forward lunge (left foot forward)*	*(lunge)*
27	*squat (two arms stretch out)*	*(squat)*

Fig 5.1: Human Actions in dataset

Source: UTD-MHAD

In the Berkeley MHAD dataset the issue of practicality or intrusiveness associated with wearing multiple inertial sensors or accelerometers in a real world setting arises. More importantly, the dataset includes 11 human actions which are rather distinct and thus distinguishable from each other, whereas UTD MHAD includes 27 human actions, some of which are similar. The UR fall detection dataset focuses on human fall detection. Their limitation is that only two types of actions (falling and non-falling) are included in the dataset thereby limiting it's applicability for evaluating recognition algorithms for other actions. The dataset contains 27 actions performed by 8 subjects (4 females and 4 males). Each subject repeated each action 4 times[22].

In UTD-MHAD only one Microsoft Kinect camera and one wearable inertial sensor were used. This was done due to the practicality aspect of using these two differing modality sensors. The advantage of using these sensors is that they are widely available, have a low cost, are easy to operate and do not require much computational power for the real-time manipulation of data generated by them. The dataset possesses large intra-class variations. For data synchronization, a time stamp for each sample was recorded. Since the frame rate of the Kinect camera and the sampling rate of the wearable inertial sensor were different, the start and the end of the action were synchronized by using the timestamps of the depth images to serve as references. Such a dataset is of benefit to researchers working in different fields such as image processing, computer vision, wearable computing and sensor fusion[22].

Chapter 6

Multimodal Human Activity Recognition

6.1 Data Preprocessing and formatting

The 27 different different actions were performed by 8 subjects (4 females and 4 males). Each subject repeated each action 4 times producing 32 samples per action. After removing three corrupted sequences, the dataset contains 861 data sequences. In our implementation, we are using the inertial and skeletal data modalities only. We split the train-validation data in the same fashion as the original paper [#reference], where subjects 1, 3, 5, 7 would be used for training and subjects 2, 4, 6, 8 for validation in order to carry out a baseline comparison. Splitting a dataset in this manner makes training a Machine Learning classifier very challenging as there are only 16 samples per action for the model to learn from and the validation is done against actions performed by subjects unseen by the model. It follows that, different subjects will perform the same action in a slightly different manner and the model has to be able to generalise based on just learning from 16 sample data points on 27 actions with varying degrees of similarity[13].

It also needs to be noted that the 861 data sequences have different frame sizes due to the difference in duration of the subjects that have carried out the various actions. Feature engineering was carried out with the main aim to extract information from the dataset that acts as a discriminator between the actions and to remove variance that could potentially disrupt the classification task. After performing feature engineering, we proceeded to apply several machine learning and deep learning methods for comparison in search of a good classifying architecture. These explorations range from ensembles to deep neural networks.

6.2 Random Forest Modelling

An ensemble method for decision trees, for the purpose of improving prediction accuracy, is random forest. Random Forest is a form of bagging (bootstrap aggregation), which involves sampling with replacement from the original population for the purpose of reducing variance (at the expense of an increase in bias, increased computational cost, and decreased interpretability of the trees). For a random forest, a large number of decision trees are

generated, and the bias is further reduced (by decorrelating the trees) by only considering a subset of the total number of features at each split in the decision tree. While constructing the decision trees of random forests, the two most important parameters that need to be determined are the number of trees(n_estimatores) in the forest and the number of features considered for splitting at each node(max_features). Increasing n_estimatores makes our predictions stronger and more stable but needs a lot of processing power.

	precision	recall	f1-score	support
0	0.46	1.00	0.63	16
1	0.24	0.25	0.24	16
2	0.64	1.00	0.78	16
3	1.00	0.94	0.97	16
4	0.75	0.75	0.75	16
5	0.92	0.75	0.83	16
6	0.79	0.94	0.86	16
7	1.00	0.62	0.77	16
8	0.67	0.25	0.36	16
9	0.47	0.44	0.45	16
10	0.71	0.62	0.67	16
11	0.84	1.00	0.91	16
12	0.81	0.81	0.81	16
13	0.88	0.94	0.91	16
14	0.92	0.75	0.83	16
15	0.92	0.75	0.83	16
16	0.94	0.94	0.94	16
17	0.83	0.94	0.88	16
18	0.70	0.44	0.54	16
19	0.77	0.62	0.69	16
20	1.00	1.00	1.00	16
21	1.00	0.88	0.93	16
22	0.88	1.00	0.94	15
23	0.94	1.00	0.97	16
24	1.00	0.94	0.97	16
25	1.00	1.00	1.00	16
26	1.00	1.00	1.00	15
accuracy			0.80	430
macro avg	0.82	0.80	0.79	430
weighted avg	0.82	0.80	0.79	430

0.7976744186046512

Fig 6.1: Classification Results of Random Forest Model

Increasing max_features generally improves the performance of the model as at each node we will have a higher number of options to be considered for making the optimal split. However, this is not necessarily true as it leads to a

decrease in the diversity of an individual tree which is the unique selling proposition of random forest. When random forest was applied to the above dataset, an accuracy of 79.77% was achieved. It was observed that class 1 has the lowest f1 score of 0.24 and the confusion matrix showed that class 1 is most confused with class 0 (0.75 vs 0.25).

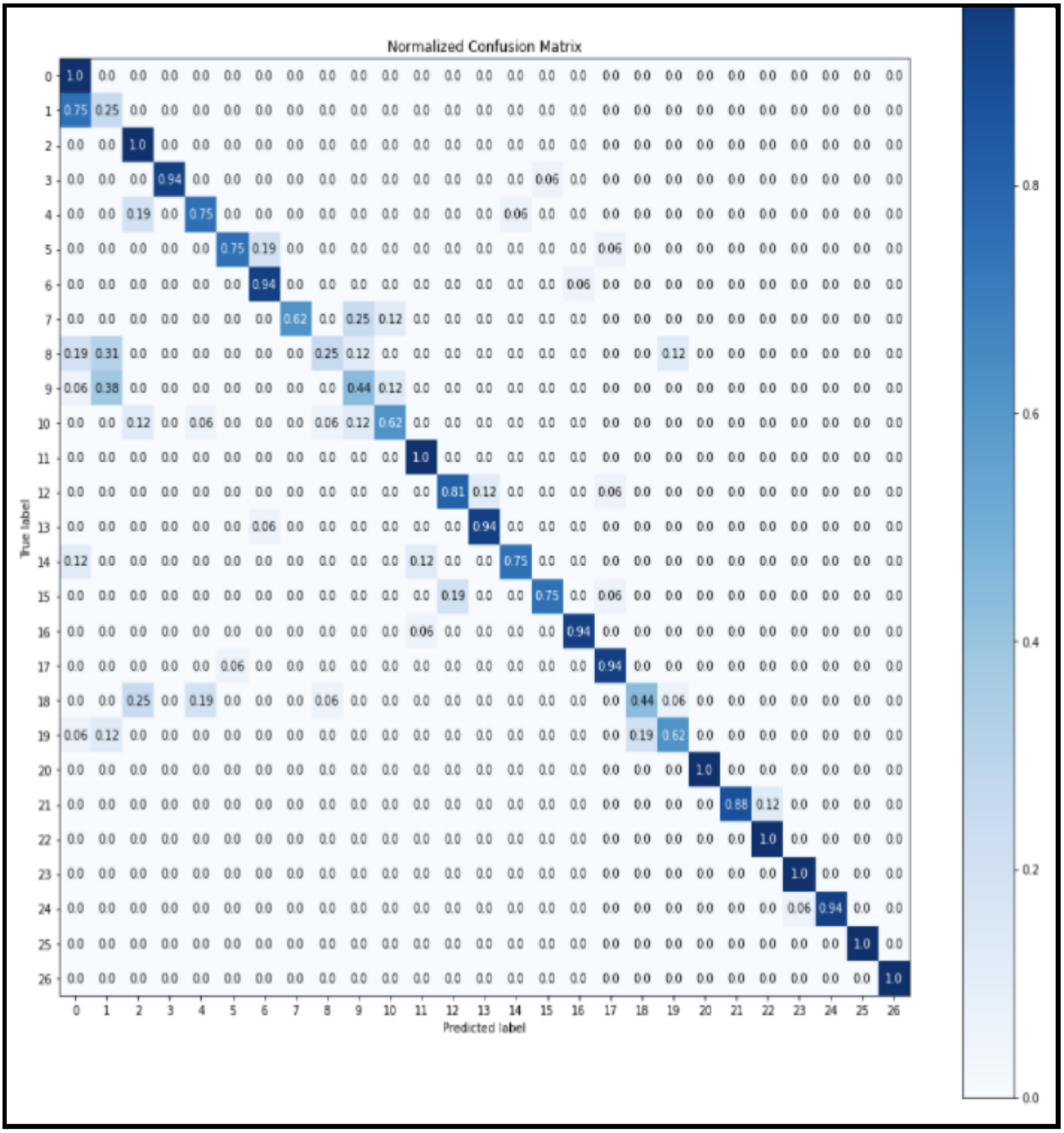

Fig 6.2: Confusion Matrix of Random Forest Model

6.3 Gradient Boosted Trees with XGBoost Modelling

The gradient boosted trees method is an ensemble learning method that combines a large number is a forest of decision trees. “Boosted” means that the model is built using a boosting process. Boosting is build on the principle that a collection of “weak learners” can be combined to produce a “strong learner,” where a weak learner is defined as a hypothesis function that can produce results only slightly better than chance and a “strong learner” is a hypothesis with an ”arbitrarily high accuracy”. When Xgboost was applied to the same dataset, an accuracy of 67.67 percent was achieved.

	precision	recall	f1-score	support
0	0.50	0.81	0.62	16
1	0.29	0.31	0.30	16
2	0.61	0.88	0.72	16
3	1.00	0.62	0.77	16
4	0.69	0.56	0.62	16
5	0.79	0.94	0.86	16
6	0.78	0.88	0.82	16
7	0.50	0.19	0.27	16
8	0.24	0.31	0.27	16
9	0.31	0.31	0.31	16
10	0.69	0.56	0.62	16
11	0.71	0.62	0.67	16
12	0.65	0.69	0.67	16
13	0.61	0.69	0.65	16
14	0.58	0.69	0.63	16
15	0.91	0.62	0.74	16
16	0.67	0.62	0.65	16
17	0.67	0.62	0.65	16
18	0.46	0.38	0.41	16
19	0.64	0.56	0.60	16
20	0.73	1.00	0.84	16
21	0.87	0.81	0.84	16
22	0.81	0.87	0.84	15
23	1.00	0.94	0.97	16
24	1.00	0.94	0.97	16
25	1.00	1.00	1.00	16
26	1.00	0.87	0.93	15
accuracy			0.68	430
macro avg	0.69	0.68	0.67	430
weighted avg	0.69	0.68	0.67	430

0.6767441860465117

Fig 6.3: Classification Results of XGBoost Model

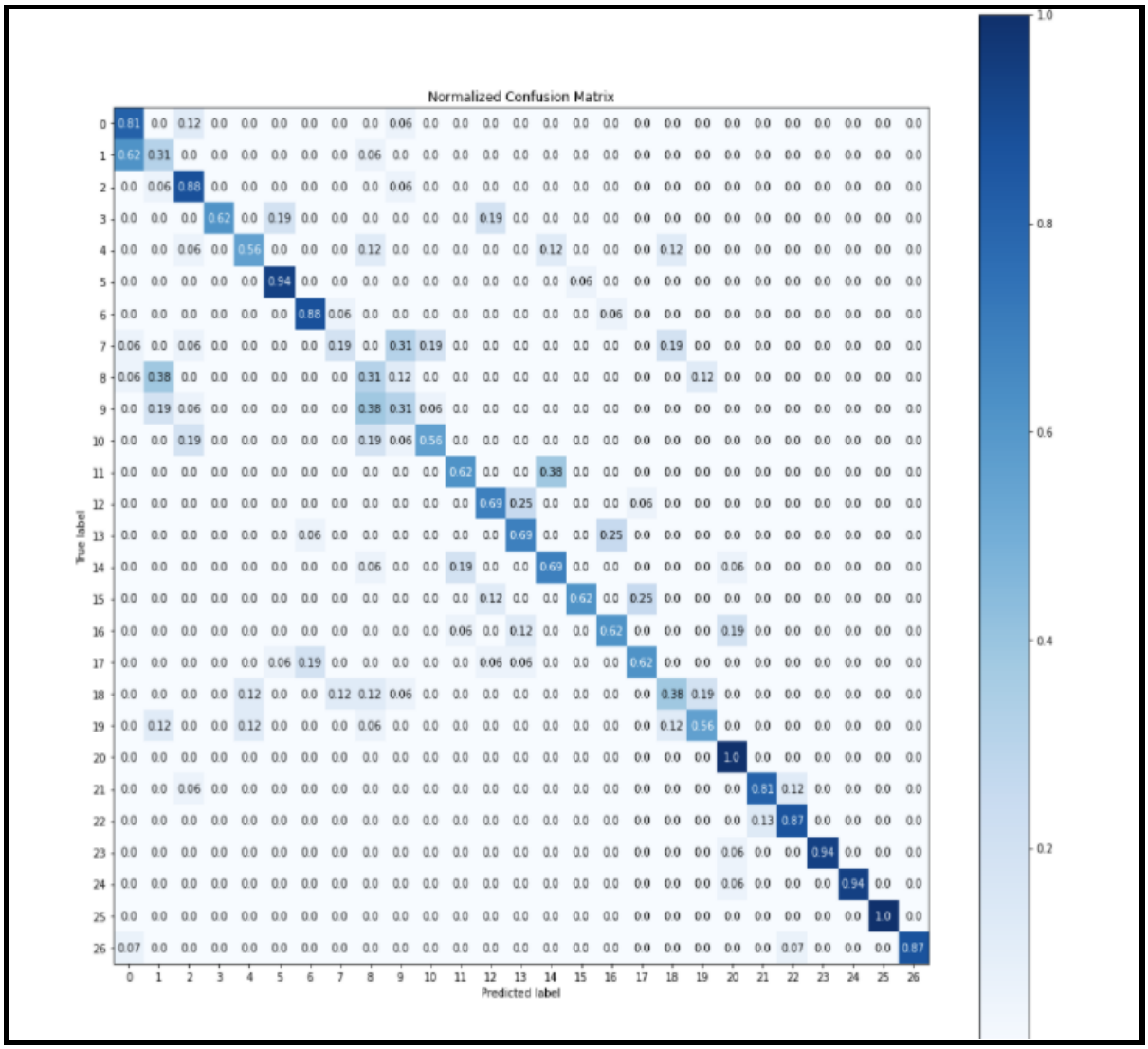

Fig 6.4: Confusion Matrix of XGBoost Model

It was observed that class 7 and class 8 have the lowest f1-score(0.27). Using the confusion matrix we can conclude that class 1 again was easily confused with class 0 (0.31 vs 0.62). Similarly, class 7 and 8 had a large number of mis-classifications but spread over multiple classes. For instance, class 7 was most confused with class 9(0.31), it was also confused with class 10 and class 18 but to a lesser extent(0.19).

6.4 Convolutional Neural Networks (CNN) Modelling

The benefit of using CNNs for sequence classification is that they can learn from the raw time series data directly, and in turn do not require domain expertise to manually engineer input features. The model can learn an

internal representation of the time series data and ideally achieve comparable performance to models fit on a version of the dataset with engineered features. Despite the Convolutional-1D Neural Network achieving a higher accuracy of 69.30 percent when compared to Xgboost, it was observed that class 8 has a f1-score of 0. From the confusion matrix, we can observe that it is misclassified as class 0(0.19), 1(0.31), 10(0.19) and 19(0.19). However, it must be noted that CNN has the highest f1-score for class 1(0.58).

	precision	recall	f1-score	support
0	0.45	0.81	0.58	16
1	0.32	0.38	0.34	16
2	0.70	1.00	0.82	16
3	0.88	0.94	0.91	16
4	0.50	0.31	0.38	16
5	0.76	1.00	0.86	16
6	0.74	0.88	0.80	16
7	0.64	0.56	0.60	16
8	0.00	0.00	0.00	16
9	0.21	0.38	0.27	16
10	0.67	0.12	0.21	16
11	0.75	0.38	0.50	16
12	0.70	1.00	0.82	16
13	0.70	0.88	0.78	16
14	0.86	0.75	0.80	16
15	1.00	0.75	0.86	16
16	1.00	0.88	0.93	16
17	0.70	0.44	0.54	16
18	0.56	0.31	0.40	16
19	0.53	0.50	0.52	16
20	0.82	0.88	0.85	16
21	1.00	0.62	0.77	16
22	0.60	1.00	0.75	15
23	0.94	0.94	0.94	16
24	1.00	0.94	0.97	16
25	0.56	0.88	0.68	16
26	1.00	0.87	0.93	15
accuracy			0.68	430
macro avg	0.69	0.68	0.66	430
weighted avg	0.69	0.68	0.66	430

0.6790697674418604

Fig 6.5: Classification Results of Convolutional Neural Network Model

When the dimensions of data increases, 2Dimensional CNN won't be efficient to handle the features that have to be extracted since it works only on the data points that are stored in a 2D matrix. So to conquer this problem the

concept of 3Dimensional CNN came. It applies 3D filters to the data that helps to extract features in all the 3 dimensions (x, y, z). A 2D CNN takes into consideration only the spatial features of the input data, whereas a 3D CNN also takes temporal characteristics of the input data into account. All the operations are performed in the 3 Dimensional manner, 3D Pooling is performed, 3D convolutions and so on. 2D CNN loses temporal information after every convolution, so all the low level features are lost after that but 3D CNN holds those low level features by moving the filters in all the 3 directions. So no Temporal information is lost while doing the convolution. One of the biggest advantages of the 3D Convolutional Neural Network is that they are not limited to the 3D shapes only, we can apply the 3D CNN to 2D images also.

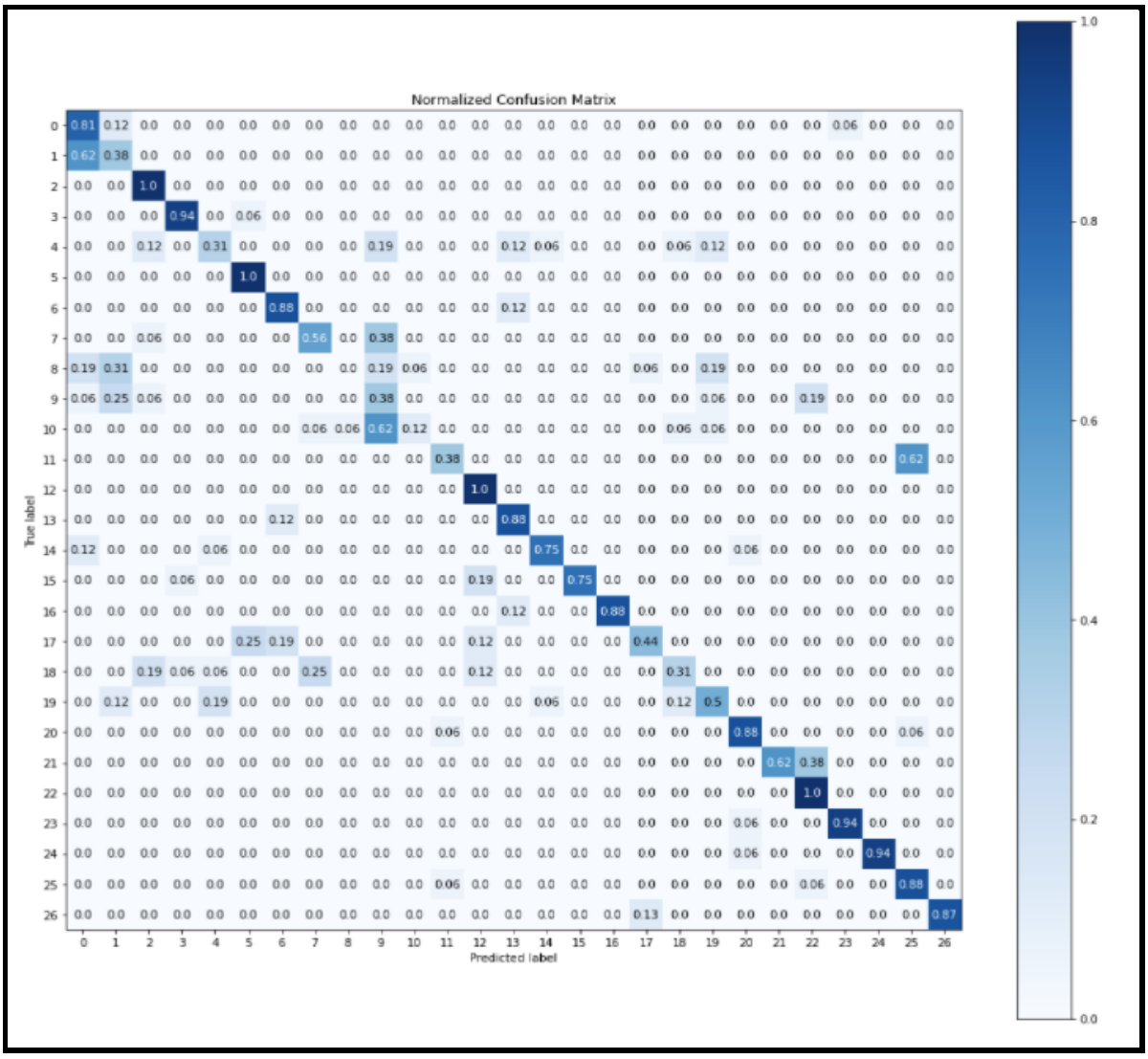

Fig 6.6: Confusion Matrix of Convolutional Neural Network Model

6.5 Convolutional LSTM Network (ConvLSTM) Modelling

ConvLSTM is a type of recurrent neural network for spatio-temporal prediction that has convolutional structures in both the input-to-state and state-to-state transitions. The ConvLSTM determines the future state of a certain cell in the grid by the inputs and past states of its local neighbors. A LSTM model with Conv 1D was built to learn the temporal pattern in the dataset. The model consists of 4 Conv1D layers, 2 LSTM layers and 1 Dense layer with Softmax activation function as shown in the figure below.

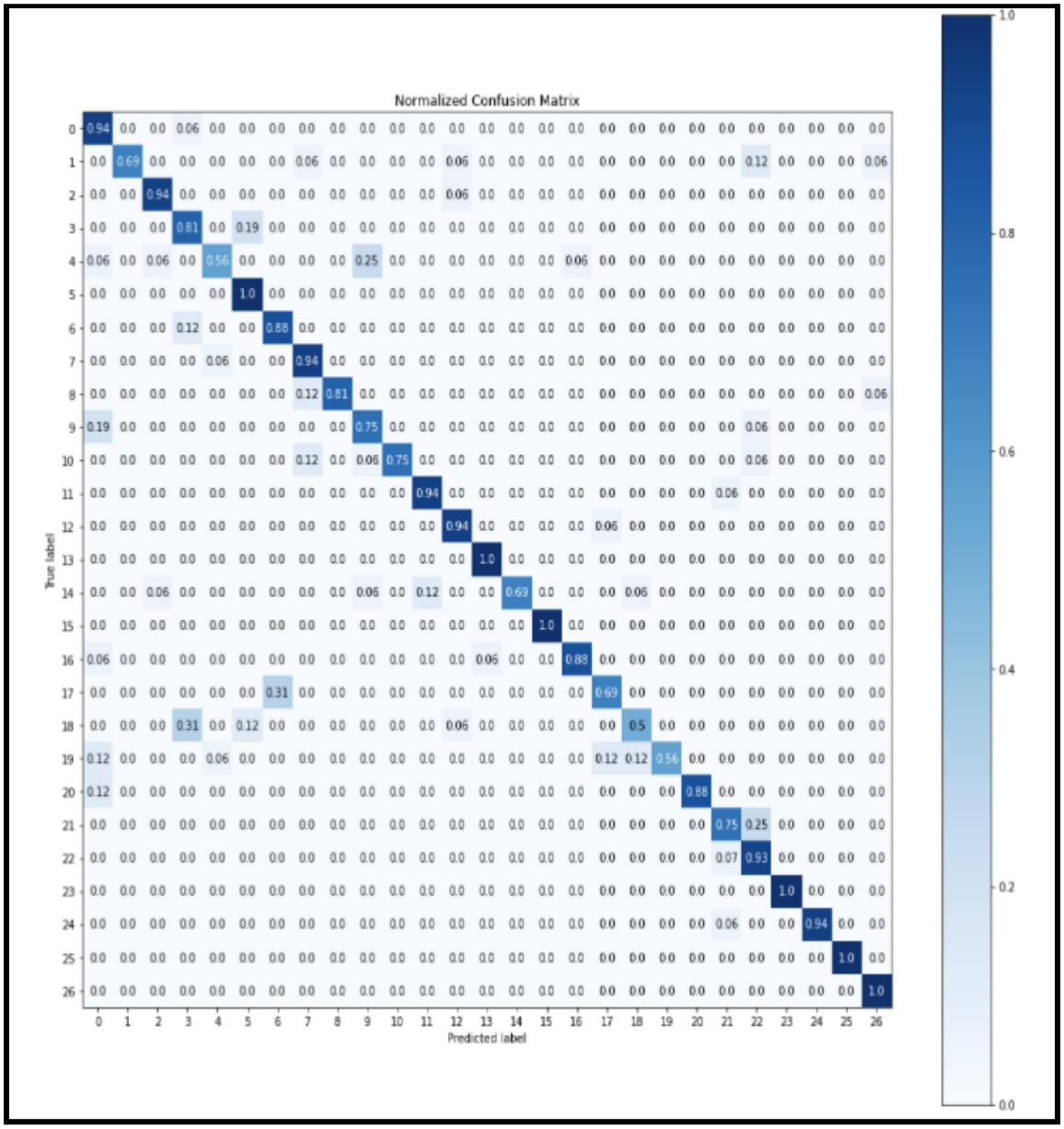

Fig 6.7: Confusion Matrix of ConLSTM Model

The purpose of the Conv1D layers is to perform feature extraction. ReLU activation function was used for all the Conv1D layers. Dropout was added at the LSTM layers to avoid overfitting. Learning rate was set to 1e-4 with a decay of 1e-6 using Adam Optimizer and batch size was set to 4. Input signals were feature engineered, re-sampled to a common length and fed into the Conv1D + LSTM Model. The model was trained for 20 epochs and it achieved an accuracy of 84.18%.

	precision	recall	f1-score	support
0	0.62	0.94	0.75	16
1	1.00	0.69	0.81	16
2	0.88	0.94	0.91	16
3	0.62	0.81	0.70	16
4	0.82	0.56	0.67	16
5	0.76	1.00	0.86	16
6	0.74	0.88	0.80	16
7	0.75	0.94	0.83	16
8	1.00	0.81	0.90	16
9	0.67	0.75	0.71	16
10	1.00	0.75	0.86	16
11	0.88	0.94	0.91	16
12	0.83	0.94	0.88	16
13	0.94	1.00	0.97	16
14	1.00	0.69	0.81	16
15	1.00	1.00	1.00	16
16	0.93	0.88	0.90	16
17	0.79	0.69	0.73	16
18	0.73	0.50	0.59	16
19	1.00	0.56	0.72	16
20	1.00	0.88	0.93	16
21	0.80	0.75	0.77	16
22	0.64	0.93	0.76	15
23	1.00	1.00	1.00	16
24	1.00	0.94	0.97	16
25	1.00	1.00	1.00	16
26	0.88	1.00	0.94	15
accuracy			0.84	430
macro avg	0.86	0.84	0.84	430
weighted avg	0.86	0.84	0.84	430

0.8418604651162791

Fig 6.8: Classification Results (ConvLSTM)

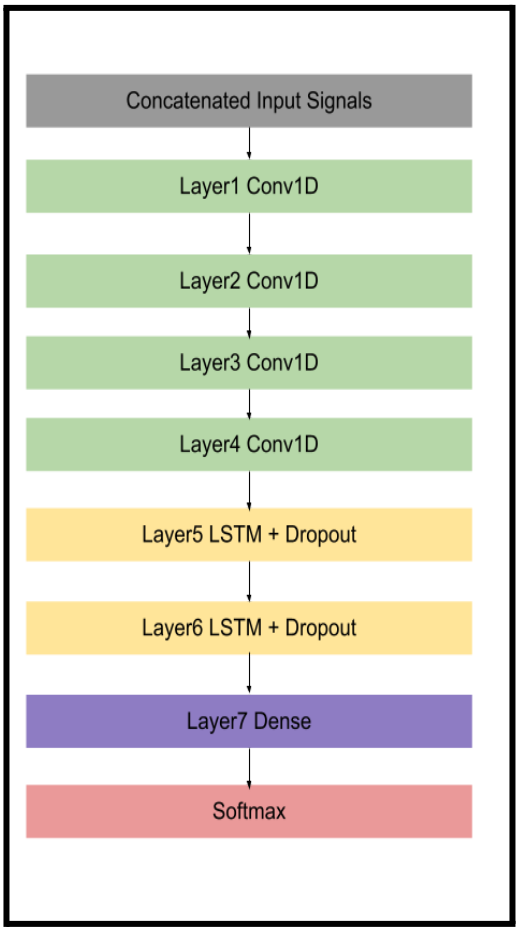

Fig 6.9: Model Architecture

6.6 Transfer Learning using VGG16

This model uses a pre-trained Image classification model to classify the human action from video. The VGG16 model is used for extracting the features from images. These features are then fed to a Neural network to perform an action recognition task. By making some changes in the VGG16 model, this model was created which achieved an accuracy of 80.556%.

This model uses publically available UTD – MHAD (University of Texas at Dallas - Multimodal Human Action Dataset) for the human action recognition task. This dataset contains a set of RGB videos which contains 27 Actions performed by 8 subjects (4 males, 4 females). There are a total of 861 videos. These videos have a duration of 2-3 seconds with a resolution of 480 x 640.

VGG16 is used for the image classification task. Since our model uses a pertained VGG16 model, the convolutional layers and pooling layers used for this model are the same as that of VGG16. But for action classification in the video, the top 3 connected layers of VGG16 are not used. instead of those layers, our model makes use of 3 different layers. The first one is the Dense layer with 2048 units with the activation function 'ReLU'. It is an input layer of this model. The input given to this layer is features extracted from VGG16. The second layer is the Dropout layer with a drop rate of 0.4. This layer is used to avoid overfitting in the model. The third layer which is the output layer makes use of 'Softmax' activation function and has 27 units. Since UTD-MHAD has a total of 27 actions, the output layer has 27 units. Our model uses 'categorical_crossentroy' as a loss function and 'Adam' optimizer with a learning rate of 0.001.

Since the UTD-MHAD has 27 actions in 861 videos, these videos were divided into two sets of 645 (Training set) and 216 (Test set) videos. Initially from all videos, 7-8 frames were extracted from each video and stored in a folder. The name of that frame and the action name of that particular frame is stored in a separate CSV file. The count of frames extracted from the video depends on the framerate of the video. Increasing frames will also increase the accuracy of this model, but the time required for prediction will also increase. Thus frame count is limited on the frame rate and duration of the video.

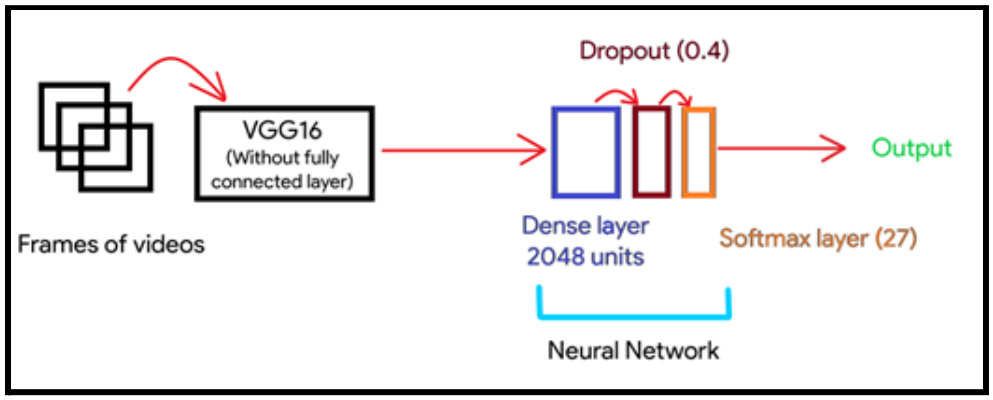

Fig 6.10: Model Workflow

In the pre-processing phase, all data from the CSV file is stored in a NumPy array. This array is passed to the VGG16 model for feature extraction. The size of frame given to the VGG16 model is 224 x 224 x 3. The output obtained (without top fully connected layers of VGG16) is a NumPy array. Each element in the reshaped NumPy array has a size of 7 x 7 x 512. In the training phase, the array of features extracted from frames of video is given to the created network. Training this model for 30 epochs gives an accuracy of 83.18%.

In the testing phase, the same process of creating frames from video, passing the frames to the VGG16 model is done and extracted features of frames are fed to the created network. The data used for testing purposes was from the UTD-MHAD but was not used for training the model. The accuracy obtained in the testing phase is 80.556%

Chapter 7

Findings and Conclusions

7.1 Summary

In our exploration we have used all the data modalities present in the UTD-MHAD dataset. However, our major focus has been on skeletal and inertial sensor data since relevant research contributions have only become recently popular. We have performed a comparative analysis of Machine Learning Classifiers and Deep Learning Models on all the data modalities available and, through understanding the classification task and the features in the dataset, the team has performed relevant feature engineering and successfully built 6 different models. The Random Forest Model has an accuracy of 79.77 percent, Customized VGG16/CNN model has an accuracy of 80.5% and the Conv1D + LSTM model has an accuracy of 84.18 percent.The accuracy of VGG16 architecture in the ILSVRC – 2014 challenge was 92.7% for an image classification task. For this task of video classification, it gave an accuracy of 80.556% by using frames from the video. Along with these models, the team also implemented Gradient Boosted Trees Model and a conventional CNN architecture to achieve an accuracy of 67.67 percent and 67.9 percent respectively. The team first began exploring conventional CNN architectures but, as LSTM Neural Networks are better at predicting sequence of data i.e. extracting the temporal information from skeletal and inertial data, a decision was made to explore the ConvLSTM architecture which helped in increasing the testing performance to 84.18%. Thus, empirically we can conclude that Random Forest is the best Machine Learning Classifier by far and ConvLSTM outperforms conventional CNN architectures which are popularly used for such tasks.

7.2 Challenges Faced

One of the major challenges the team faced was that in order to perform a baseline comparison, the train-validation data needs to be split in the same fashion as the original paper[22], that is using subjects 1, 3, 5, 7 for training and subjects 2, 4, 6, 8 for validation. Splitting the dataset in this manner makes training a machine learning model very challenging as there are only 16 samples per action for the model to learn from and the validation is done against actions performed by subjects unseen by the model. Intuitively,

different subjects will carry out the same action in a slightly different manner and the model had to be able to generalise based on just learning from 16 sample data on 27 actions with varying degrees of similarity. It was also noted that the 861 data sequences have different frame sizes due to the difference in duration of the subjects carrying out the various actions. Another issue that plagued the progress of the team were the resource constraints, as Grid Search, training 20 epochs of ConvLSTM and other hyperparameter tuning techniques took a very long time to run on Google Colab due to low CPU availability. As our most accurate classifiers make use of skeletal and inertial signals as input, this results in a drawback. Skeletal and inertial data collection is done through sensors, which runs into challenges like difficulty in choosing the right interval to collect and transmit data in real time, analytical noise created by poor sensor data quality and dangers of neglected data security etc.

7.3 Further Advancements

To further improve the accuracy in the task of classifying human actions, future exploration works could include more advanced feature engineering, incorporate other modifications in the existing ConvLSTM architecture or implement a rule based system to assist in differentiating between very similar actions that machine learning models could not tell apart.

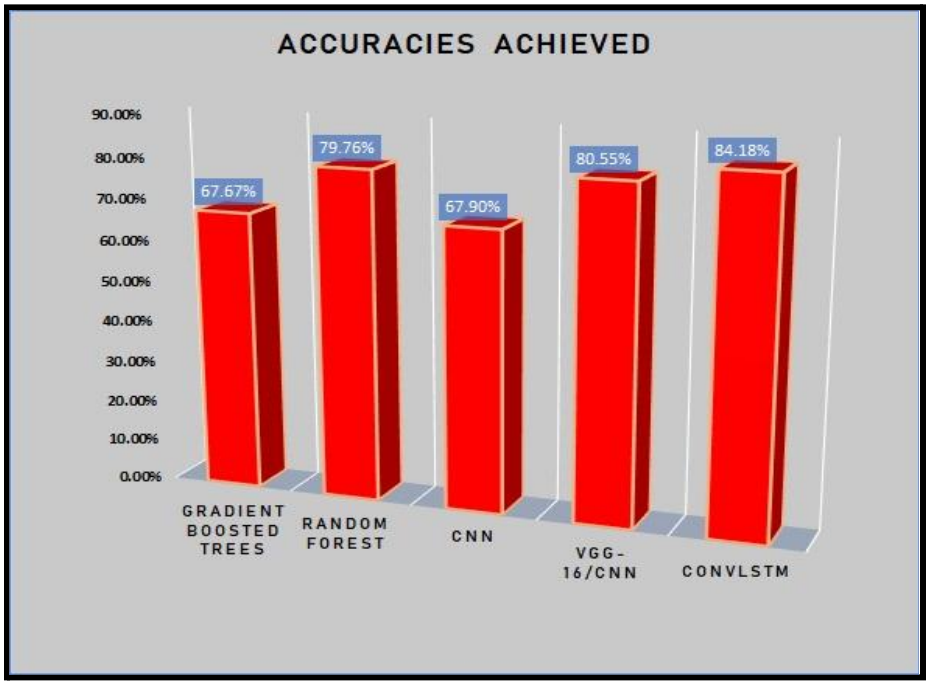

Fig 7.1: Accuracy comparison of all the developed Models

Chapter 8
References

1. Ben Mahjoub, Amel & Atri, Mohamed. (2019). An efficient end-to-end deep learning architecture for activity classification. Analog Integrated Circuits and Signal Processing. 99. 10.1007/s10470-018-1306-2.

2. Wang, P., Liu, L., Shen, C., & Shen, H. T. (2016). Order-aware convolutional pooling for video based action recognition. arXiv preprint arXiv:1602.00224.

3. Asadi-Aghbolaghi, M., Clapes, A., Bellantonio, M., Escalante, H.J., Ponce-Lopez, V., Baro´, X., Guyon, I., Kasaei, S., & Escalera, S. (2017). A survey on deep learning based approaches for action and gesture recognition in image sequences. In 2017 12th IEEE international conference on automatic face & gesture recognition (FG 2017) (pp. 476–483). IEEE.

4. Si, Chenyang & Jing, Ya & Wang, Wei & Wang, Liang & Tan, Tieniu. (2020). Skeleton-Based Action Recognition with Hierarchical Spatial Reasoning and Temporal Stack Learning Network. Pattern Recognition. 107. 107511. 10.1016/j.patcog.2020.107511.

5. Y. Du, Y. Fu and L. Wang, "Skeleton based action recognition with convolutional neural network," 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR), 2015, pp. 579-583, doi: 10.1109/ACPR.2015.7486569.

6. Bo Li, Yuchao Dai, Xuelian Cheng, Huahui Chen, Yi Lin and Mingyi He, "Skeleton based action recognition using translation-scale invariant image mapping and multi-scale deep CNN," 2017 IEEE International Conference on Multimedia & Expo Workshops (ICMEW), 2017, pp. 601-604, doi: 10.1109/ICMEW.2017.8026282.

7. Liu J., Shahroudy A., Xu D., Wang G. (2016) Spatio-Temporal LSTM with Trust Gates for 3D Human Action Recognition. In: Leibe B., Matas J., Sebe N., Welling M. (eds) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science, vol 9907. Springer, Cham. https://doi.org/10.1007/978-3-319-46487-9_50.

8. Wu Q, Xu G, Chen L, Luo A, Zhang S (2017) Human action recognition based on kinematic similarity in real time. PLoS ONE 12(10): e0185719. https://doi.org/10.1371/journal.pone.0185719.

9. Islam, Saiful & Bulbul, Farhad & Islam, Md. (2018). A Comparative Study on Human Action Recognition Using Multiple Skeletal Features and Multiclass Support Vector Machine. Machine Learning and Applications: An International Journal. 5. 01-15. 10.5121/mlaij.2018.5201.

10. Shafaei, Alireza, and James J. Little. “Real-time human motion capture with multiple depth cameras.” In 2016 13th Conference on Computer and Robot Vision (CRV), pp. 24-31. IEEE, 2016.

11. Moussa, Mona & Hemayed, Elsayed & Fayek, M. & Elnemr, Heba. (2013). An enhanced method for human action recognition. Journal of Advanced Research. 10.1016/j.jare.2013.11.007.

12. Komang, Mandira & Nasution, Surya & Ratna, Astuti. (2019). Human activity recognition using skeleton data and support vector machine. Journal of Physics: Conference Series. 1192. 012044. 10.1088/1742-6596/1192/1/012044.

13. Y. Du, W. Wang, and L. Wang, “Hierarchical recurrent neural network for skeleton based action recognition,” in IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2015, pp. 1110–1118.

14. J. Liu, A. Shahroudy, G. Wang, L.-Y. Duan, and A. C. Kot, “Skeleton-based online action prediction using scale selection network,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 42, no. 6, pp. 1453–1467, 2019.

15. Javed Imran, Praveen Kumar, “Human Action Recognition Using RGB-D Sensor and Deep Convolutional Neural Networks”, in Int. Conference on Advances in Computing, Communications and Informatics(ICACCI),(2016).

16. Pushpajit Khaire, Praveen Kumar, Javed Imran, “Combining CNN streams of RGB-D and skeletal data for human activity recognition”, Pattern Recognition Letters (2018), doi: 10.1016/j.patrec.2018.04.035.

17. Das Dawn, Debapratim & Shaikh, Soharab. (2015). A comprehensive survey of human action recognition with spatio-temporal interest point (STIP) detector. The Visual Computer. 32. 1-18. 10.1007/s00371-015-1066-2.

18. Ben Mahjoub, Amel & Atri, Mohamed. (2016). Human action recognition using RGB data. 83-87. 10.1109/IDT.2016.7843019.

19. Kim, Tae Soo & Reiter, Austin. (2017). Interpretable 3D Human Action Analysis with Temporal Convolutional Networks. 1623-1631. 10.1109/CVPRW.2017.207.

20. Mishra, Soumya & Mishra, Tusar Kanti & Sanyal, Prof(Dr.) Goutam & Sarkar, Anirban & Satapathy, Suresh. (2020). Real time human action recognition using triggered frame extraction and a typical CNN heuristic. Pattern Recognition Letters. 135. 329-336. 10.1016/j.patrec.2020.04.031.

21. Ma, Miao & Marturi, Naresh & Li, Yibin & Leonardis, Ales & Stolkin, Rustam. (2017). Region-sequence based six-stream CNN features for general and fine-grained human action recognition in videos. Pattern Recognition. 76. 10.1016/j.patcog.2017.11.026.

22. C. Chen, R. Jafari, and N. Kehtarnavaz, "UTD-MHAD: A Multimodal Dataset for Human Action Recognition Utilizing a Depth Camera and a Wearable Inertial Sensor", *Proceedings of IEEE International Conference on Image Processing*, Canada, September 2015.

23. Xiao, Yang & Chen, Jun & Cao, Zhi-Guo & Zhou, Joey & Bai, Xiang. (2018). Action Recognition for Depth Video using Multi-view Dynamic Images.

24. Zhao, Chong & Chen, Minglin & Zhao, Jinhao & Wang, Qigong & Shen, Yehu. (2019). 3D Behavior Recognition Based on Multi-Modal Deep Space-Time Learning. Applied Sciences. 9. 716. 10.3390/app9040716.

25. Kamel, Aouaidjia & Sheng, Bin & Yang, Po & Li, Ping & Shen, Ruimin. (2018). Deep Convolutional Neural Networks for Human Action Recognition Using Depth Maps and Postures. IEEE Transactions on Systems Man and Cybernetics. PP. 10.1109/TSMC.2018.2850149.

26. Trelinski, J., Kwolek, B. CNN-based and DTW features for human activity recognition on depth maps. *Neural Comput & Applic* (2021). https://doi.org/10.1007/s00521-021-06097-1.

27. Tasnim, Nusrat & Islam, Md & Baek, Joong-Hwan. (2020). Deep Learning-Based Action Recognition Using 3D Skeleton Joints Information. Inventions. 5. 49. 10.3390/inventions5030049.

28. http://cs229.stanford.edu/

29. Machine Learning Models and CNN code link: here

30. ConvLSTM Network code link: here