Sign Language Recognition

A DISSERTATION

Submitted in partial fulfilment of the requirements for the award of the degree of

MASTER OF TECHNOLOGY

in

ELECTRICAL ENGINEERING

(With Specialization in Instrumentation and Signal Processing)

Submitted By:
DEBTANU HAZRA

Under the guidance of:
Prof. R.S. Anand

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE-247667(INDIA)
JUNE 2019

CANDIDATE’S DECLARATION

We hereby declare that the work which is presented in this project, entitled **Sign Language Recognition**, towards the fulfilment of requirements for the award of the degree of **Master of Technology** in **Electrical Engineering**, submitted to the Department of Electrical Engineering **Indian Institute of Technology Roorkee**, is an authentic record of our own work carried out by me during the period of July 2018 to June 2019 under the guidance of **Prof. R.S. Anand**, Department of Electrical Engineering, IIT Roorkee. The matter embodied in this thesis report, to the best of our knowledge, has not been submitted by me for the award of any other degree of this institute or any other institutes.

Date: June 10, 2018
Place: Roorkee

Debtanu Hazra
Enrol. No.: 17528003

CERTIFICATE

This is to certify that the above statement made by candidates is correct to the best of my Knowledge and belief

Dr. R.S. Anand
Professor
Department of Electrical Engineering
IIT Roorkee

ACKNOWLEDGEMENT

I would like to thank my supervisor **Prof. R.S. Anand** to help us select the right project, guiding and helping us and to devote his valuable time to our project. His support and continued motivation helped us to gain insight into various aspects of research and development and to complete the project in time.

We would also like to extend our thanks to global open source communities for their help in providing the implementations for various modules of our project. Our project depended on several open source libraries and publicly available datasets for efficient and effective implementations.

I am very much thankful to **Prof. B. Das**, Head of Department (Electrical Engineering) and the laboratory assistant of the **Biomedical Laboratory** for providing us with the necessary infrastructure and apparatus required to complete this project.

Last but not the least, I would also like to thank our friends who have provided us with constant support and motivation. We also sincerely acknowledge everyone who has directly or indirectly helped us in my project.

ABSTRACT

Sign Language Recognition using machine learning is an image classification project that has an important social motive behind it. The problem statement of this project can be summarized as a problem of image and video classification using deep learning algorithms.

The things that make this problem challenging is to use easily accessible hardware such as simple web cams or smartphone cameras instead of advanced devices such as kinect. This eliminates the element of depth from the data, and compromises the accuracy of the classification task, but this trade-off is necessary in order to make this project usable and scalable. Other challenges include getting good quality labeled data. The data of ISL gestures consists of videos of dynamic hand signs which involves use of one or both hands depending upon gesture complexity. For the CNN-LSTM and CNN-MLP models features were extracted from the video frames using transfer learning on the Inception V3 model with preloaded ImageNet weights.

The models trained displayed good accuracy as well as the demo developed is capable of running in a real time with reasonably good prediction accuracy.

The technologies behind this project is computer vision and deep learning. We used library OpenCV with python that gives the capability to work with images and Keras with Tensorflow for building models.

This project has lot of social and marketable potential. This project was extended using recurrent convolutional network models to identify phrases and words. With the help of existing APIs provided by google or other companies for Natural Language Processing, this project can be developed into full-fledged translator, a product which will have a very low cost of scalability and can affect lives of many people.

ABBREVIATIONS

CNN	Convolutional Neural Networks
LRCN	Long-Term Recurrent Convolutional Networks
LSTM	Long Short-Term Memory
MLP	Multi-Layer Perceptron
2D	Two-dimensional
3D	Three-dimensional
CV	Computer Vision
GPU	Graphical Processing Unit
IEEE	The Institute of Electrical and Electronics Engineers
ILSVRC	ImageNet Large Scale Visual Recognition Competition
ReLU	Rectified Linear Unit
SGD	Stochastic Gradient Descent

CONTENTS

LIST OF FIGURES:

LIST OF TABLES:

CHAPTER 1

INTRODUCTION AND LITERATURE REVIEW

1.1. INTRODUCTION:

This project is focused on translation of American Sign Language (ASL) using supervised classification. The deaf and dumb people use sign language and find it difficult to communicate with those who do not understand sign language. This impairs the ability of deaf and dumb people to communicate smoothly in the real world. We strive to tackle this problem using image and video classification techniques using deep learning.

Image classification is a widely researched area with many applications in vast number of areas. With the development in the field of machine learning and deep learning, there is a deluge of work, both from academicians as well as from industrialists. This field has many engineering challenges, right from conceiving the idea to making a deployable product. The things that make this problem challenging is to use easily accessible hardware such as simple web cams or smartphone cameras instead of advanced devices such as Kinect. This eliminates the element of depth from the data, and compromises the accuracy of the classification task, but this trade off is necessary in order to make this project usable and scalable. Other challenges include getting good quality labelled data. For this we got ASL fingerspelling dataset, as well as we have created our own data. To train a deep learning algorithm, that contains millions of parameters to optimize, data is required in large quantity. To tackle this we used a technique called transfer learning as well as we have tried to create simple and small models over a simplified dataset created by us. Even after the model is trained with good accuracy, there is a challenge of making it available to personal devices, as these devices are not generally optimized to perform large quantities of matrix operations in the absence of special hardware such as GPUs.

With the internet reaching millions of people in the last decade there has been an explosion in flow of information. The easy availability of multimedia devices like smart phones and cameras in an abundance has led to data being shared every minute in the form of images and videos. Hence it is of utmost importance to analyze these data of images and videos for various needs like search, recommendation, recognition and ranking. The computer vision community has been working for decades on these problems to achieve state of the art results of various real-life problems like video classification, activity recognition, video description and detection of abnormalities in videos. The key reasons for getting such results has been availability of large labelled datasets and techniques that scale up learning models to millions and millions of parameters which support the learning process for performing any specified task. Under such conditions CNNs have shown to perform very well especially on image data by learning robust and interpretable features for various image based applications. Inspired by the positive results of deep learning performance on images it has been extended to large scale video classification. In case of images deep learning networks have to learn spatial features only whereas in case of video classification both spatial and temporal features come into play. Hence the problems become more complex in case of videos compared to images and practically there are no video classification benchmarks that can match the diversity and scale of image datasets since it is more difficult to collect and store videos. The IIITA ROBITA ISL gesture dataset was collected which consist of videos pertaining to 11 classes of data. the videos contain mostly hand gestures which uses one or both hands to perform the dynamic signs of the phrases that are to be classified.

For the past decade there has been massive progress in the image domain with respect to feature learning using convolutional neural network architectures. Such pre trained networks or ConvNet models are made available for extracting features which are very useful in transfer learning tasks. But in case of video classification such image based features are not directly suitable for classification due to lack of motion modelling. So these features are applied to recurrent neural network which can be applied to time series classification problems.

From a computational perspective CNNs require extensively long time to train and optimize the millions and millions of parameters that are integral part of the model. Hence the problem becomes further complex for videos where the architecture has to be extended in time where it has to learn several frames of the video at a time instead of just one image. In this paper we show that convolutional neural networks with recurrent units can be applied to time series modelling and put forward the argument that in case of video related applications where flat temporal models has been previously applied, LSTM style RNNs are able to provide significant amount of improvement given there is availability of ample amount of data for learning. Specifically, we show that CNN-LSTM models provide improved recognition of videos. While the existing ISL dataset may not contain gestures of complex temporal dynamic but nonetheless it shows that by directly connecting a convolutional model to deep LSTM or deep MLP network that capture temporal features gives us significant improvement on the flat temporal models previously applied to the same dataset. These deep learning architectures offer improvement in performance over previous statistical machine learning models which were applied as a generic framework for recurrent models and were realized using popular deep learning framework Keras which is simple and easy to implement as it includes ready to use implementation of RNN and LSTM units.

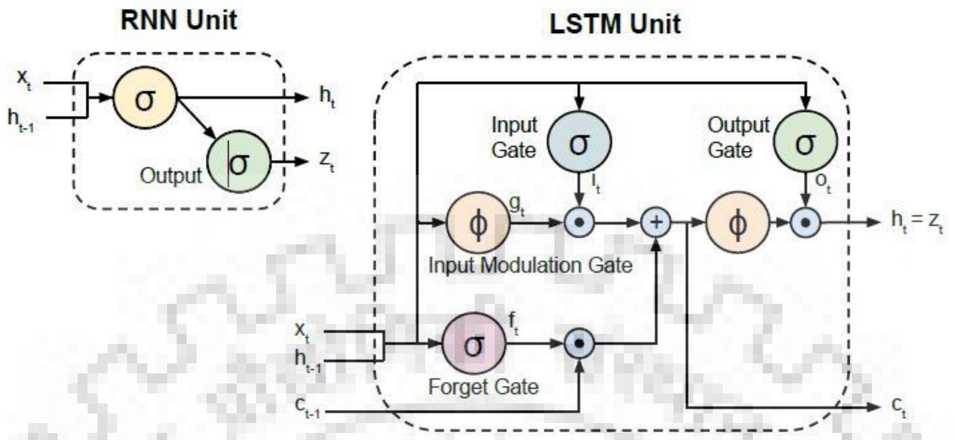


Figure 1.1 Basic RNN(left) and LSTM(right) units [1]

We will introduce a few constraints in order to reduce the computational complexity by simplifying the network architectures for future applications in real time systems.

- Optical flow of images was not used to keep the model simple and reducing a significant number of hyperparameters thus reducing the training time.
- Pre-processing of image frames were not done. The frames were kept pretty raw from starting to the end.
- Every video will be subsampled to 40 frames by fast forwarding videos which may contain more number of frames than 40.
- Each model has to fit into 4GB GPU that was used for training purposes.

1.2. MOTIVATION:

There are many motivations behind taking this task, primary of which includes the needs of the deaf and dumb community. A human need to communicate, for functioning in the society, to express needs, to express emotions, and a language that is limited to a small community of deaf and dumb people puts constraint on this basic need. Even though a real time translation of complete sign language without the need of server communication is a far fetched dream, nonetheless, a small progress in the right direction can elevate the quality of these peoples' life.

Other motivation being chance to work in the area with lots of research interests and potential. Deep learning is one of the research area that has academicians as well as professionals working to produce quality research published every day in peer reviewed journals. The problems as discussed in previous section makes it an interesting field to work and understand. Also working on the state of the art architectures such as Inception-v3 [2] and VGG-16 [3] allowed us to delve deeper in such an interesting and challenging field.

1.3. PROBLEM STATEMENT:

The problem statement is to develop a classifier that classifies alphabets and numbers from the image. Alphanumeric characters forms the building block of a language. There are 24 alphabetic characters (excluding 'j' and 'z' because these characters involve motion) and 10 numeric characters (0-9). The system must be reliable with adequate accuracy of classification. The problem of excessive computation should be handled with a simple simplification of white background in order to develop a real time model without the use of GPUs for the purpose of demonstration.

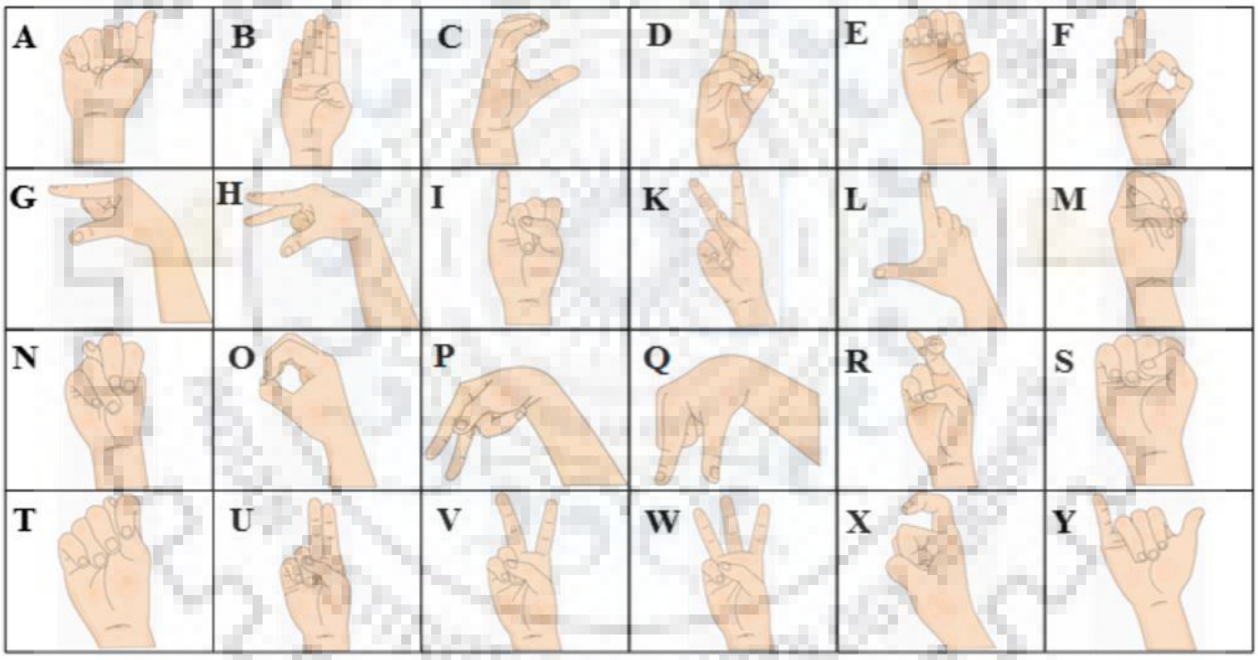


Figure 1.2 ASL Alphabets [4]

Sign language recognition does not only involve the correct recognition of sign language alphabets. The goal has always been to recognize or interpret the gestures performed by a signer into words or actions depending upon the application of the project undertaken. Now these gestures performed are very complex and involves more than one phrases or words to build sentences. So in order to understand the context and recognize sentences it is very important that the model recognizes gestures that translates into words of a particular sign language vocabulary. Since I worked on the ASL fingerspelling dataset to classify images into different alphabets it didn't make sense to work on any other sign language vocabulary for recognition of words. But the unavailability of a workable dataset that consists of words of the American sign language which would fit in with our GPU hardware limitations and also made publicly available for the use of researchers I decided, that since the overall goal is to identify words of a particular sign language vocabulary which can be done and applied similarly for any other vocabulary, to work on the IIITA ROBITA ISL Gesture dataset which contains dynamic gestures that represent words of the ISL vocabulary. The dataset consists of 11 classes of dynamic gestures which would be classified using robust deep learning models to take a step further in the right direction of interpreting sign language in real time.

1.4. LITERATURE SURVEY:

1.4.1. IMAGE CLASSIFICATION:

Previous work has been done in the field of automatic sign language recognition, although, most of the work include special sensors such as kinect or accelerometer. Even when the plain cameras are used then also there is an attempt to segment hand portion out of the image and then process it. The segmentation itself is done using thresholding for skin colour, which in itself poses the problem of robustness, as different lighting condition and skin colour can lead to errors. Even though when we use adaptive thresholding and setting the threshold parameters on the fly, there is a lot of noise that creeps into the system and affects the performance of the system. As a result, we looked at other techniques involving convolutional neural networks [5], [6] to tackle the problem of automated sign language recognition.

Nowadays in pursuit of getting state of the art results in image classification nobody wants to build their model from scratch unless it's concerned with a domain of work where transfer learning models fail. Only then one would train the network end to end. Otherwise its very common to use transfer learning for any type of image recognition problem since the computational cost is much less than end to end trainable networks and also the initial layers of a deep neural network is always responsible for learning general features that can be applied to any domain of image recognition problem. Now the pre trained models most popularly used for transfer learning applications consist of the Inception network which provided state of the art results on the ImageNet dataset which will be discussed in detail later on. The AlexNet [5] paper of 2012 discusses deep convolutional neural networks and their application on ImageNet classification which contains over 15 million images. Hence it is no surprise this model took it 5-6 days to complete training on ImageNet data on two NVIDIA GTX 580 GPUs. It achieved an error rate of 15.4% on the dataset. The ZFNet paper of 2013 achieved an error rate of

11.4% on the same dataset by reducing the filter size from (11,11) to (7,7) as it was observed that the larger filter size discards important pixel information. AlexNet was trained on 15 million images but ZFNet [6] was trained on 1.3 million images and it took 12 days to train on the same GPU as AlexNet training. The VGGNet [3] paper came out in 2014 was even more deeper than both of the above discussed architectures which had an error rate of just 7.3% on the same dataset. The GoogleNet(Inception Module) [2] paper first came out in 2015 which was much deeper convolutional network achieving an error rate of 6.7% on the ImageNet dataset. Also in 2015 Microsoft came up with their paper on ResNet [7] which was a 152 layer deep convolution neural network and it achieved an incredible error rate of 3.6% on the ImageNet challenge where human error rate lies between 5-10%.

1.4.2. VIDEO CLASSIFICATION:

Before the vast advancements of deep learning in the domain of action recognition and video classification, most of the traditional Computer Vision algorithm variants for action recognition can be differentiated into the following three major steps:

- Local high-dimensional visual features that describe a region of the video are extracted either densely [8] or at a sparse set of interest points [9], [10].
- The extracted features get combined into a fixed-sized video level description. One popular variant to the step is to bag of visual words (derived using hierarchical or k-means clustering) for encoding features at video-level.
- A classifier, like SVM or Logistic Regression, is trained on a collection of visual words for final prediction

Among all of these algorithms that involve extraction of shallow handcrafted features the improved Dense Trajectories [11] paper published state-of-the art results. It used densely sampled trajectory features for the purpose of action recognition. Around 2013, 3D convolutions [12] were experimented on for action recognition but it wasn’t of much help. Then came the breakthrough year of 2014 which introduced two pioneering research papers that form the backbone of the methods that are discussed in this report. Among these two papers the basic difference was the architecture surrounding the design choice of combining spatial and temporal features.

1.4.2.1. SINGLE STREAM NETWORKS:

The temporal information between consecutive frames are explored in multiple ways in order to fuse them with the spatial information from the frames by Karpathy et al. [13] in their work. The fusion was carried out by pre-trained two-dimensional convolutions.

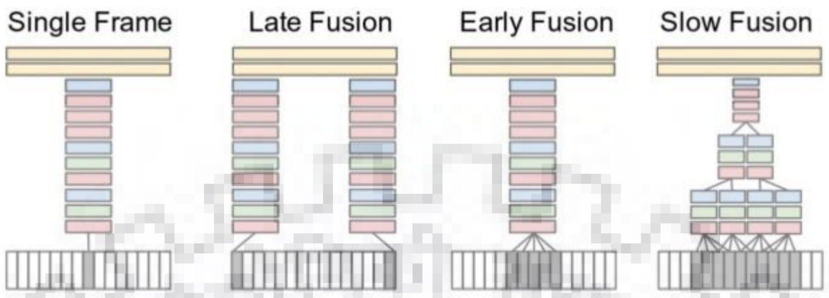


Figure 1.3 Fusion Ideas [13]

It is clearly evident from Fig 1.3 that all the above setups involve consecutive frames serving as input. The single frame architecture fuses information from all the frames at the last stage. The late fusion architecture uses two networks with shared parameters those are spread 15 frames apart and likewise fuses all the information at the end. The early fusion architecture uses convolution over 10 frames and combines all the information in the first layer. The slow fusion architecture offers a balance between the early and late fusion architectures by fusing information at multiple stages. For final prediction multiple clips were sampled from the whole video and the prediction scores obtained from them were averaged for a final prediction. Despite extensive experiments using benchmark action recognition datasets like UCF 101 or Sports-1M the results were not even as good as state-of-the art hand-crafted feature-based algorithms. The possible reasons for such performance can be listed below.

- The spatiotemporal features that were learned did not include the learning of motion features
- It was very tough to learn detailed and deep features if the dataset has less diversity and is a roadblock to achieving state-of-the art results.

1.4.2.2. TWO STREAM NETWORKS:

In this pioneering work [14] by Simmoyan and Zisserman, the authors build on the failures of the previous work by Karpathy et al. [13]. Since it is very tough for deep networks to learn motion features the authors introduced optical flow vectors which was modelled from the motion features. Instead of having one single network for spatial context this architecture constitutes of two separate networks. One, a pre-trained network for spatial context and another one for motion context. A single frame of the video serves as input to the spatial network while authors kept experimenting with the input to the temporal network and found bi-directional optical flow stacked across for 10 successive frames gave the best performance. The spatial network stream and the temporal network stream were trained separately and then fused together using support vector machines [14]. The final prediction was made in the same way by averaging the prediction scores across all the sampled frames. The results obtained after implementing this network did improve by capturing local temporal movement, it had its drawbacks:

- Since the video level predictions were obtained by averaging prediction scores over sampled clips, the long range temporal information which we discussed earlier, was lost from learnt features.

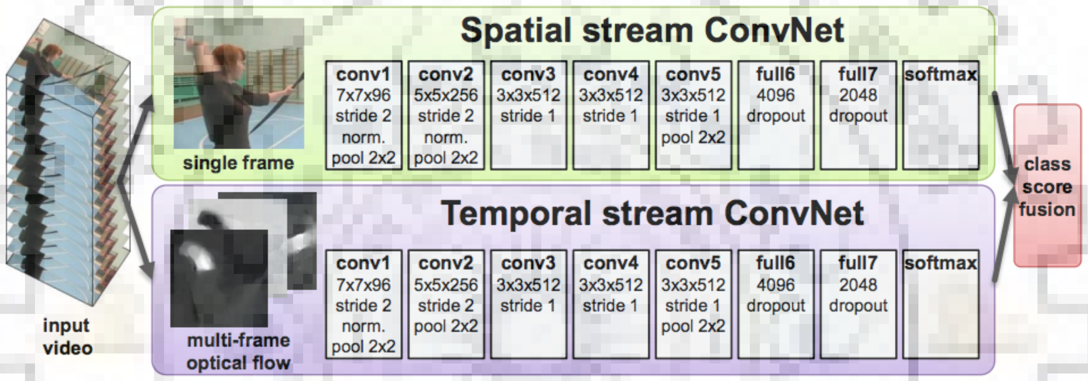


Figure 1.4 Two stream architecture [14]

- Since training clips are sampled uniformly from videos, they suffer from a problem of false label assignment. The ground truth of each of these clips are assumed same as ground truth of the video which may not be the case if the action just happens for a small duration within the entire video.
- The method involved pre-computing optical flow vectors and storing them separately. Since the spatial and temporal networks were trained separately it implies that training the whole network from scratch on-the-go is still a very long road.

The recurrent theme around the papers that were to be published in the future were improvised on the basis of these two ideas. In a previous work by Ng et al. [15] authors had explored the idea of using LSTMs on separately trained feature maps to see if it can capture temporal information from clips. One of the significant drawbacks to this work was the inability to capture long range temporal information which was offered a solution by Varol et al. [16] in their work tried to compensate for the stunted temporal range problem by using lower spatial resolution of video and longer clips (60 frames) which led to significantly better performance. In 2014 in this [17] work authors built upon work by Karpathy et al. However, instead of using 2D convolutions across frames, they used 3D convolutions on video volume. The idea was to train these vast networks on Sports1M and then use them (or an ensemble of nets with different temporal depths) as feature extractors for other datasets. They found out that it was a simple linear classifier like SVM on top of ensemble of extracted features worked better than state-of-the-art algorithms. The model performed even better if hand crafted features like iDT were used additionally. The only drawback of this work was that the computational cost of the model was very high.

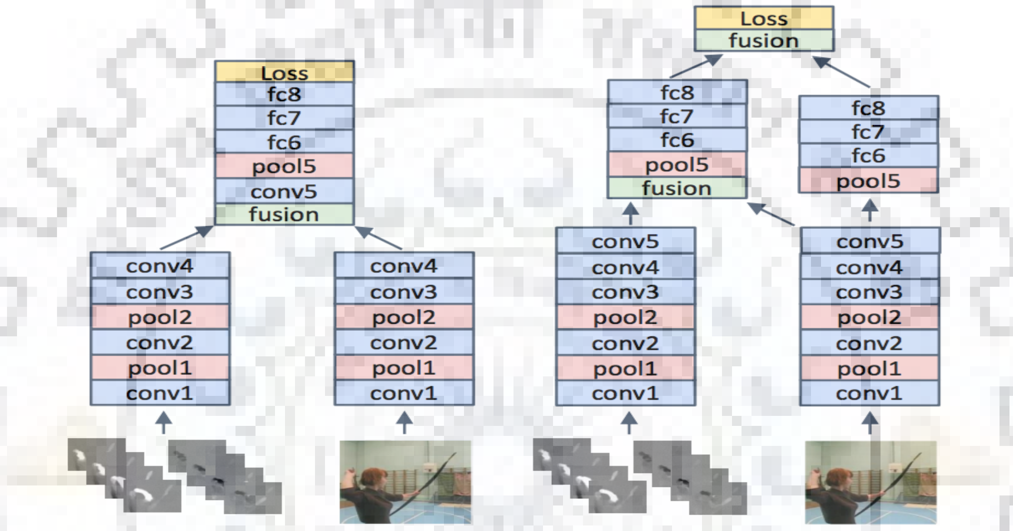


Figure 1.5 Two stream architecture(left) [14] two stream fusion(right) [18]. The one on the right performed better on UCF 101 Split 1 dataset

Temporary segment networks [19] work very similar to two stream fusion network [18] giving the same accuracy on the UCF 101-split 1 dataset and uses optical flow of images. The ActionVlad [20] architecture uses the ActionVlad layer instead of max pooling or average pooling providing good results for action recognition tasks. Another approach came out in 2017 was the hidden two stream which used the two stream fusion network [21] and temporary segment networks as the basis of the network architecture.

There was work done using the IITA ROBITA ISL gesture dataset with respect to classification in real time of both dynamic and static gestures that represent phrases of the Indian Sign Language in [22], [23]. Later on this was extended to interpretation of ISL for human robot interaction in [24]. Among all the approaches discussed above the best suited to us for the classification of ISL gestures of the IIITA ROBITA dataset considering the computational complexity and the length of the dataset the approach is to extract features from the image frames of each video into a dropout layer and feed forward those features into and LSTM network that will give us good results after video classification.

CHAPTER 2

COMPUTER VISION AND DEEP LEARNING

2.1. COMPUTER VISION:

Computer Vision has two primary goals, one from the biological perspective it aims to introduce computational or mathematical models that can imitate the human visual system and two, from engineering point of view it strives to build autonomous systems that can perform some of the same tasks that the human visual system can perform and also outperform the human visual system in a few cases. Different types of vision tasks are related to extraction of 3D features **[12]** and temporal information **[19]** from time-varying data which are obtained from various multimedia devices which provides a more general understanding of such dynamic scenes. Now these two goals are intimately related. The different aspects of the human visual system offer some understanding to engineers on how to design the computer vision systems. Basically, the main part of the human visual system is the visual cortex **[25]** which contains neurons that get activated upon receiving visual stimulus. And it has been observed that different sets of neurons get activated depending upon the visual stimulus that is provided which paved the way for the basis of convolutional neural networks and its role in computer vision.

2.2. BRIEF HISTORY OF COMPUTER VISION:

It is commonly accepted that the father of Computer Vision is Larry Roberts, who in his Ph.D. thesis (cir. 1960) at MIT discussed the possibilities of extracting 3D geometrical information from 2D perspective views of blocks [26]. Many researchers, at MIT and elsewhere, in Artificial Intelligence, followed this work and studied computer vision in the context of the blocks world.

Later, researchers realized that it was necessary to tackle images from the real world. Thus, much research was needed in the so called ``low-level" vision tasks such as edge detection and segmentation. A major milestone was the framework proposed by David Marr (cir. 1978) at MIT [27], who took a bottom-up approach to scene understanding. Looking over the history of computer vision, it is important to note that because of the broad spectrum of potential applications, the trend has been the merge of computer vision with other closely related fields. These include: Image processing (the raw images have to be processed before further analysis) [28]. Photogrammetry (cameras used for imaging have to be calibrated. Determining object poses in 3D [27] is important in both computer vision and photogrammetry).

2.3. APPLICATION OF COMPUTER VISION: PAST, PRESENT AND FUTURE:

Through time computer vision has coma a long way with its applications which include autonomous navigation, industrial inspections, object recognition and tracking, action recognition and human robot interactions and many more. The results have always been mixed at best except for applications like industrial inspections which require 2D image processing and pattern recognition (which has shown enormous success). The main difficulty with various computer vision tasks is that there are a lot of algorithms that work very well when confined to one domain of application. This translates to the fact that due to absence of more general algorithms that can be applied to all types of applications computer vision is limited by brittle algorithms. For a CV application to be successful it should satisfy the following two criteria:

- Possibility of human interaction.
- Ignoring some of the mistakes.

Vision applications can be combined with audio or other modalities to achieve specific goals. Measured against these two criteria, some of the exciting computer vision applications which can be potentially very successful include:

- Image/video databases - Image content-based indexing and retrieval.
- Vision-based human computer interface - e.g., using gesture (combined with speech) in interacting with virtual environments.
- Virtual agent/actor - generating scenes of a synthetic person based on parameters extracted from video sequences of a real person.

It is heartening to see that a number of researchers in computer vision have already started to delve into these and related applications**.**

2.4. HISTORICAL TRENDS IN DEEP LEARNING:

Deep learning is not a new technology, it might appear to be new because of different names it has been given over the past, as well as, it was relatively unpopular in the past. There have been three major waves of development in this field from cybernetics in 1940s-1960s, connectionism in 1980s-1990s and finally the deep learning since 2006.

2.5. DEEP LEARNING AND HUMAN LEARNING:

The earlier models were referred to as Artificial Neural Network as they were thought to be the computational model of biological brain. They were thought to be inspired from the brain of a biological organism and how they learn. Even many breakthroughs in the field of deep learning comes from an attempt to mimic the workings of human brain. One important example of this is the Convolutional Neural Networks. The convolutional neural networks were actually thought to be computational model of our visual cortex. In the visual cortex not all neurons are activated when the receptors are excited. Instead of that different sets of neurons were activated for different types of excitation. This is what the convolution layer filters do specifically by learning features and different neurons(filters) would learn different feature.

Human brain, a marvel in itself, has given researchers the reason to pursue a single algorithm for all the machine learning tasks.

2.6. CONVOLUTIONAL NEURAL NETWORKS:

In mathematics, a convolution is a function which is applied over the output of another function. In our case, we will consider applying a matrix multiplication (filter/Kernel) across an image [6]. In deep learning, cross-correlation is called convolution although convolution operation in math the is a little different from cross-correlation.

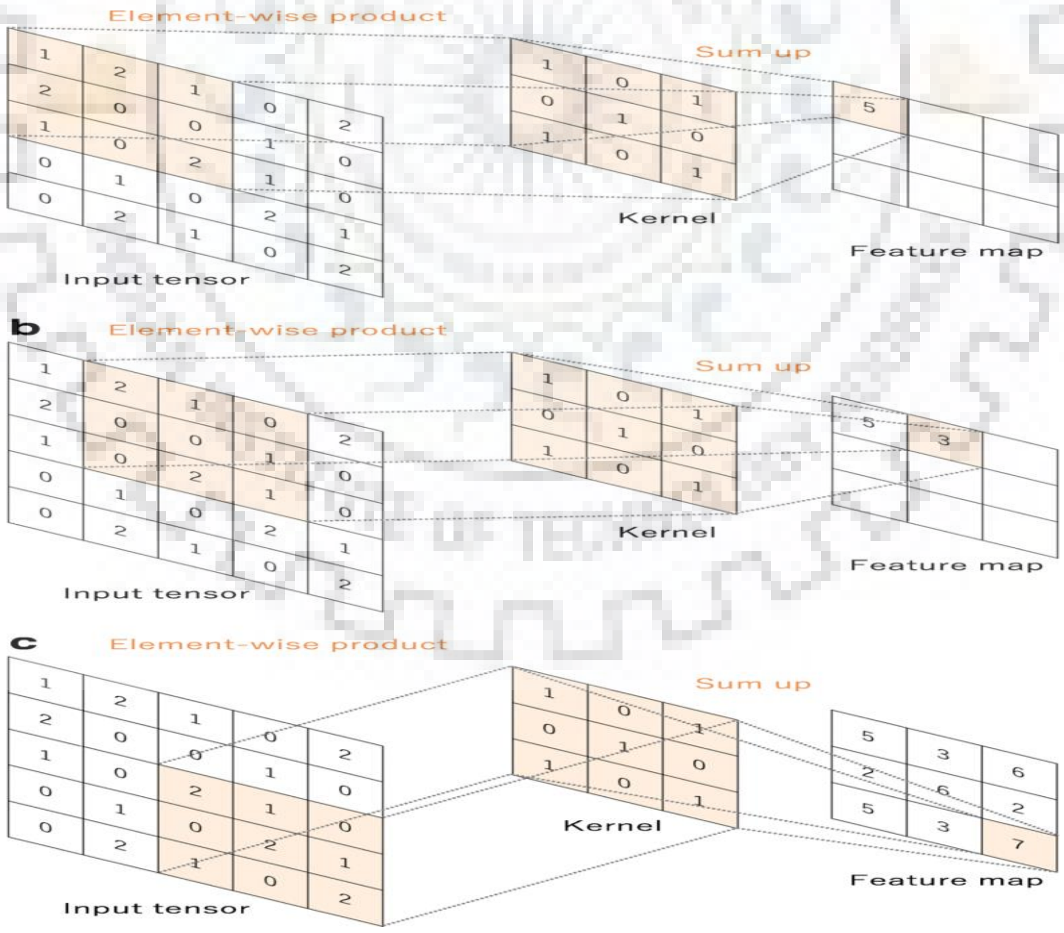


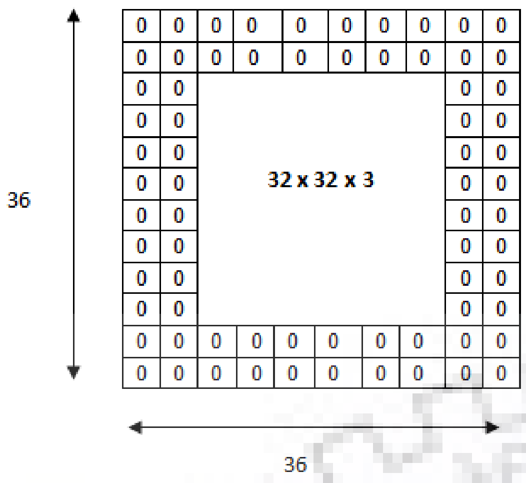
Figure 2.1 Filtering of input image by convolution filter [43]

Convolutional Neural Networks (CNNs) are a special kind of multi-layer neural networks. They are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other. And they still have a loss function (e.g. SVM/Softmax) on the last (fully-connected) layer. However, ConvNet architectures make the explicit assumption that the inputs are images, which allows us to encode certain properties into the architecture. These then make the forward function more efficient to implement and vastly reduce the amount of parameters in the network [29]. Therefore, they can recognize patterns with extreme variability (such as handwritten characters), and with robustness to distortions and simple geometric transformations [30]. CNNs have various architectures: LeNet [29], AlexNet [5], VGGNet [31], GoogLeNet [2], ResNet [7] and etc. Here, LENET-5 one of the simplest architectures is considered to make a prediction of the sign language is discussed later on.

2.6.1. STRIDE AND PADDING:

Stride controls how the filter convolves around the input volume. The filter convolves around the input volume by shifting one/two unit at a time. Stride is normally set in a way so that the output volume is an integer and not a fraction. Let's look at an example. Let's imagine a 7 x 7 input volume, a 3 x 3 filter (Disregard the 3rd dimension for simplicity), and a stride of 2. So, as we can see, the receptive field is shifting by 2 units now and the output volume shrinks as well. Notice that if we tried to set our stride to 3, then we'd have issues with spacing and making sure the receptive fields fit on the input volume. Normally, programmers will increase the stride if they want receptive fields to overlap less and if they want smaller spatial dimensions.

What happens when you apply three 5 x 5 x 3 filters to a 32 x 32 x 3 input volume? The output volume would be 28 x 28 x 3. Notice that the spatial dimensions decrease. As we keep applying conv layers, the size of the volume will decrease faster than we would like. In the early layers of our network, we want to preserve as much information about the original input volume so that we can extract those low level features. Let's say we want to apply the same conv layer but we want the output volume to remain 32 x 32 x 3. To do this, we can apply a zero padding of size 2 to that layer. Zero padding pads the input volume with zeros around the border. If we think about a zero padding of two, then this would result in a 36 x 36 x 3 input volume.



The input volume is 32 x 32 x 3. If we imagine two borders of zeros around the volume, this gives us a 36 x 36 x 3 volume. Then, when we apply our conv layer with our three 5 x 5 x 3 filters and a stride of 1, then we will also get a 32 x 32 x3 output volume.

Figure 2.2 Zero padding with two layers [5]

2.6.2. POOLING:

It is also referred to as a down-sampling layer. In this category, there are also several layer options, with MaxPooling being the most popular. This basically takes a filter (normally of size 2x2) and a stride of the same length. It then applies it to the input volume and outputs the maximum number in every subregion that the filter convolves around.

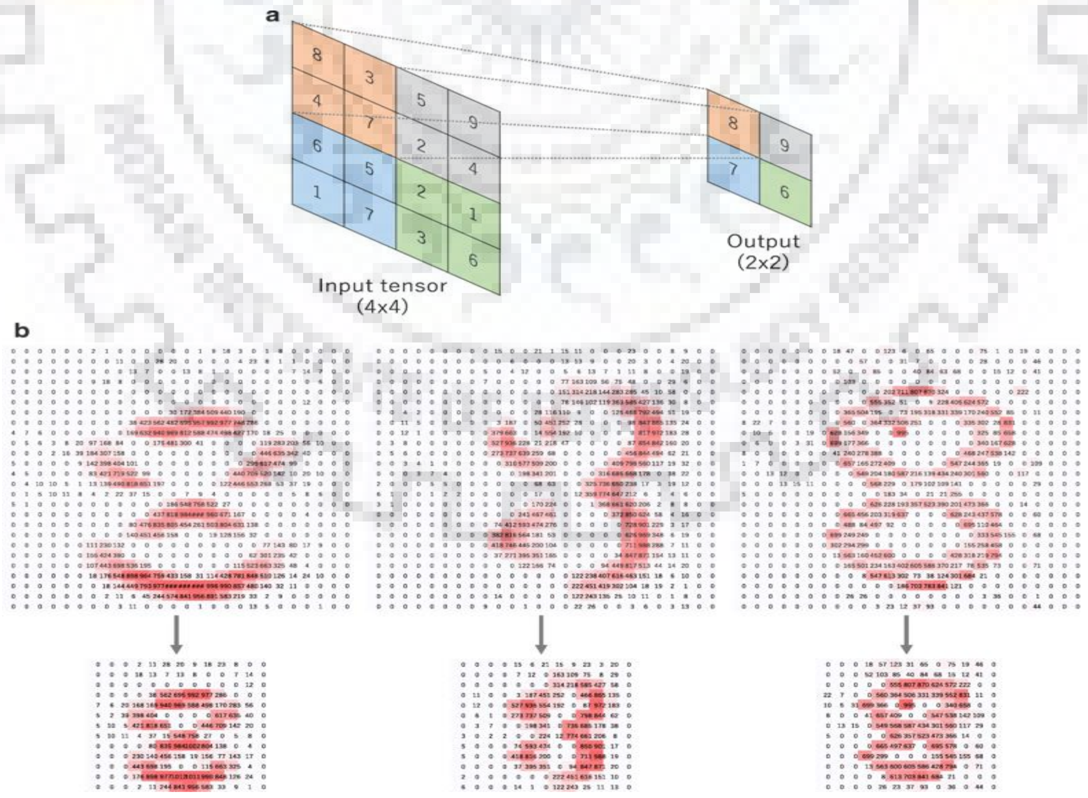


Figure 2.3 Max pooling example [43]

2.7. LENET-5(1998) [29]:

LeNet-5, a pioneering 7-level convolutional network by LeCun et al in 1998, that classifies digits, was applied by several banks to recognise hand-written numbers on checks (cheques) digitized in 32x32 pixel images. The ability to process higher resolution images requires larger and more convolutional layers, so this technique is constrained by the availability of computing resources [5]. For example, leNet-5 Start with an image of 32 x 32 x 1 and goal was to recognize handwritten digit. In the first step we use six 5 x 5 filter with stride 1 and get 28 x 28 x 6. With stride of 1 and no padding we reduce the dimension to 32 x 32 to 28 x 28. Then we use a average pooling with a filter width of 2 and stride of 2 and reduce the dimension by factor of 2 and end up with 14 x 14 x 6. Then we use another convolutional layer with sixteen 5 x 5 filter and end up with 10 x 10 x 16. That time people use valid padding that's why each time height and weight shrinks. Then another pooling layer and end up with 5 x 5 x 16. Then the next layer is a fully connected layer with 120 nodes. The previous layers 400 (5516) then connected with this 120 neurons. Then another layer this 120 nodes connected with a 84 node and use this to connected with that with possible 10 values that will recognize digit from 0 to 9. But in modern version of this neural net we use Softmax function with a ten wave classification output.

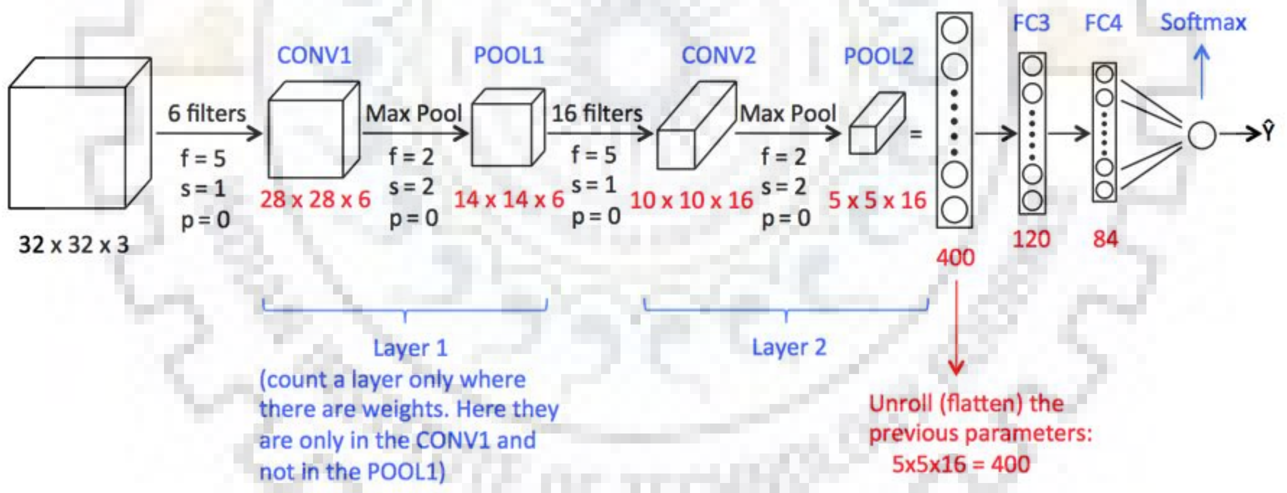


Figure 2.4 LeNet 5(1998) model visualization [29]

2.8. CURRENT SCENARIO:

The current boom in the field of deep learning is fuelled by few things ranging from exploding amount of data generated every second, and the ability of the hardware to process that exploding amount of data. Lots of data is generated every second from the sensors all over the world, most important being the smartphones with cameras.

Also with the advent of modern processors present even in the simple laptops that has the ability to perform up to million computation in just one cycle and new hardware like GPUs, specifically built for performing matrix operations, the ability to process the data is also increased.

The current research in the field of deep learning has become truly interdisciplinary ranging from medicine to entertainment.

CHAPTER 3

METHODOLOGY

3.1. DATA COLLECTION:

3.1.1. ASL FINGER SPELLING DATASET:

ASL Fingerspelling Sign Language MNIST dataset consist of two datasets, Dataset A and Dataset B. This dataset consists of both RGB and Depth images for 24 alphabets (excluding ‘j’ and ‘z’ which requires hand motion). This dataset is organized by the subjects. Now for classification of static sign language alphabets, performance can be significantly improved by using a masked dataset which enables the learning algorithm to compute significantly less number of features which would make the training time and computation so much more less that it can be done without having to rely on a GPU. All the images belonging to different classes in the dataset would consist of static gesture images wherein the hand of the user performing the gesture would be masked to simplify the edge detection and other features that would be learned by the CNN. For prediction to be done in real time the camera would record the user’s hand and mask the skin of the user and then it would be fed to the prediction algorithm giving us much better results in real time.

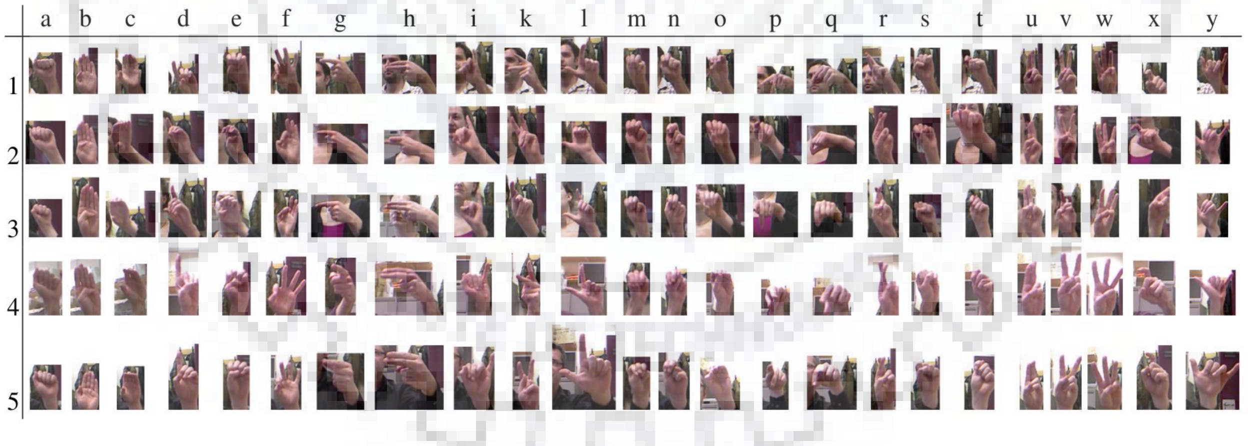


Figure 3.1 ASL Finger spelling data from Sign Language MNIST dataset

Although the dataset that has been used consists of masked images. Images were masked from coloured RGB image where the skin is masked. Below is shown an image from A-I and K-Y classes of the data from the testing set.

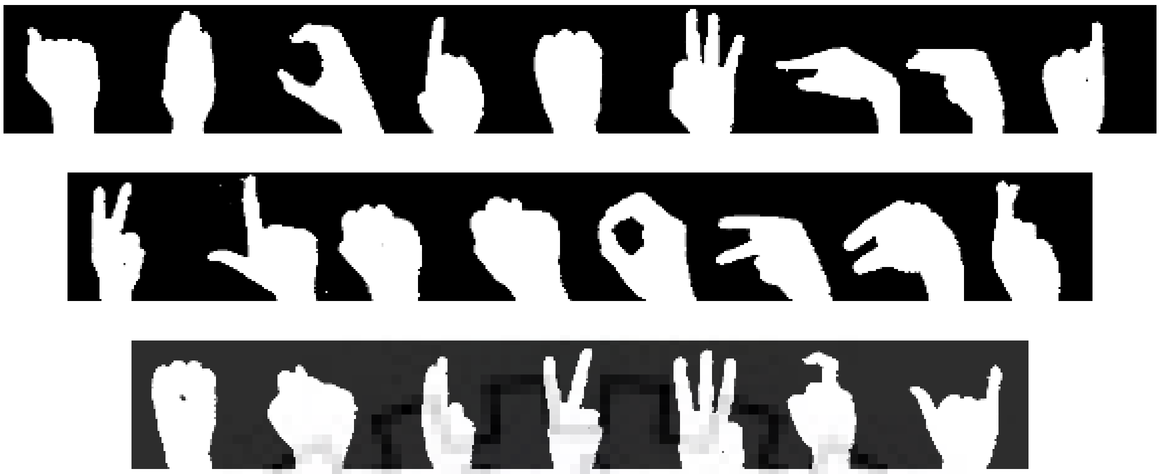


Figure 3.2 Images of each alphabet from the testing set of the ASL fingerspelling dataset

The dataset was obtained from GitHub where it was already split into training and test folders which contained 24 more folders each belonging to each of the 24 classes and the folder names served as the labels of the dataset. Among these the class folders in the training set contained 1750 images each belonging to a single class whereas the class folders in the test set contained 250 images each belonging to a particular class. The dimension of each image was found out to be (64, 64).

3.1.2. IIITA ROBITA ISL GESTURE DATASET:

The dataset as mentioned earlier was provided by IIITA Robotics and AI Lab. They used constant background while recording the video because background removal is computationally complex task and it affects the recognition result in real time [22]. As the feature extraction technique is concerned about the gray scale images so the background was chosen dark. Every ISL gesture implies some class or word which could be captured by waving your both hands in a very appropriate manner. We considered versatile gestures which needs fast and accurate movements of hands. We created repository with huge number of images (sequence of images) of several kind of ISL class/word. Preliminarily ISL dynamic gesture has been recorded with different frame rate per second (fps) and with different sizes which would be used for training and testing during classification. As ISL gesture deals with motion so we recorded video using Sony handy cam with 2.5 mega pixel resolution. We considered 11 specific ISL words as shown in Fig 3.3, under various illumination condition.

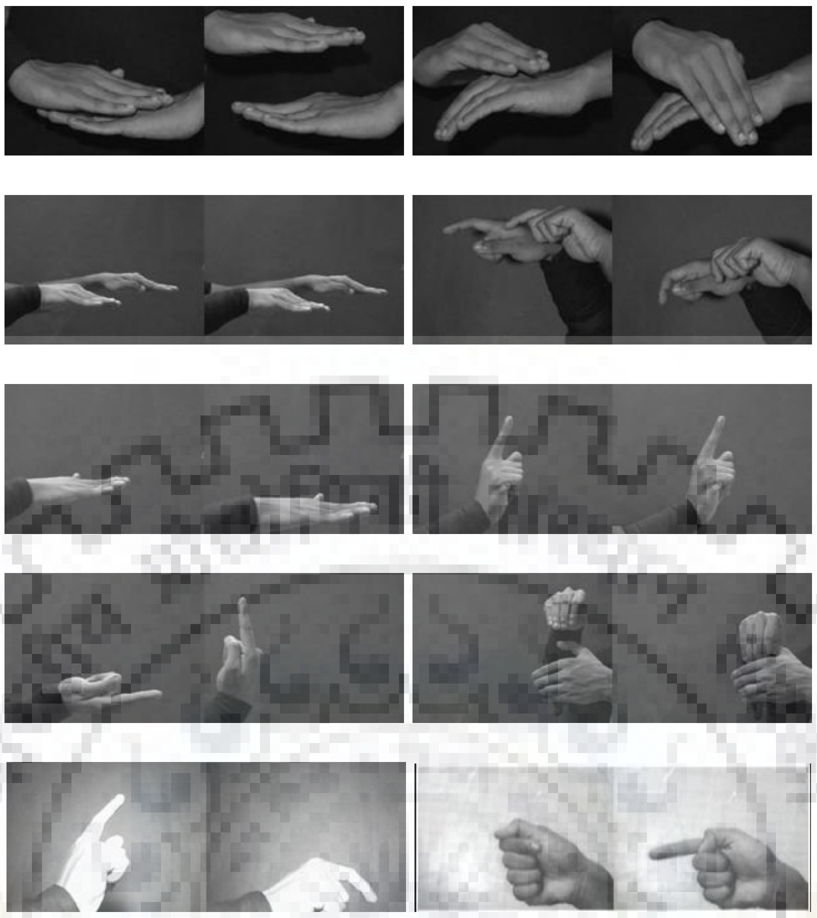


Figure 3.3 Start and End Frames of a few classes from the IIITA ROBITA ISL Gesture dataset [22]

The dataset is only available for research purposes and is supplied by IITA Robotics and AI lab for those who need it free of cost. They share the data via google drive services and since it consists of image frames hence it does not take up a lot of space on the drive. The dataset shared contained 11 classes of dynamic gestures which consisted of image frames of videos where these gestures are being performed. These image frames are already extracted from each set of video and these sets of videos are differentiated into two folders namely gallery list and probe list which will later be sorted into training and test set data. Since our target was always to extract frames from videos uniformly to help simplify the model these sets of video frames were again converted to videos using the Python "ffmpeg" tool and sorted into training and test data for each of the classes. After that frames were again extracted uniformly from each of the videos which were later looped through in order to create the sequence of features. A CSV file was created to help with referencing throughout the training, which would contain the names of the videos, their class, train/test status and frame count. Before doing this of course the videos were split into train/test folders and JPEGs were extracted of each frame from each video. Below is tabulated the number of videos belonging to each class after train/test split.

Table 3.1 Number of videos belonging to different classes split into train/test folders

ISL Word	Training	Testing
Above	13	6
Across	8	4
Advance	12	5
Afraid	15	7
All	10	4
Alone	7	3
Arise	8	4
Bag	7	4
Below	7	4
Bring	10	5
Yes	3	1

It is evident from the above table that the dataset is not very large. This is one of the drawbacks of this dataset since training a robust deep learning model requires huge amount of data for state-of-the-art results. Another limitation of the dataset provided is the lack of diversity among the gestures performed. Although the dataset consists of videos being performed in different illumination conditions it was performed under one background condition only and very few number of users participated in developing the dataset. Hence there are no considerable difference in the way these gestures are performed which would not have been the case if a lot of people had volunteered to contribute in developing a robust dataset.

3.2. ASL ALPHABET EXPERIMENTS:

Since we need to build a model that can run in real time on hardware without the capability of GPUs, we simplified the problem statement to cover only the recognition task over white background.

We trained separately for ASL Alphabet. The data was collected as described in previous chapter. This was done in 3 experiments. Each experiment include three processes.

- Data Collection
- Pre-processing
- Training

Data collection and training process are explained in details in the previous chapters. Pre-processing involves randomization of data and splitting the data for training and testing purpose. It may also include dimension change of images.

Experiment 1 included collection of 2000 images for each class. Images were grayscale and of dimension 64x64 pixels. These were then split into training and testing part in the ratio of 7:1. The model was trained for 25 epochs with 800 steps for each epoch.

Table 3.2 CNN Model Parameters

Layer(type)	**Output Shape**	**Hyper-parameters**
Conv2D	(None, 62, 62, 32)	896
MaxPooling	(None, 31, 31, 32)	0
Conv2D	(None, 29, 29, 32)	9248
MaxPooling	(None, 14, 14, 32)	0
Conv2D	(None, 12, 12, 64)	18496
MaxPooling	(None, 6, 6, 64)	0
Flatten	(None, 2304)	0
Dense	(None, 256)	590080
Dropout	(None, 256)	0
Dense	(None, 24)	6168
Total parameters: 624,888 Trainable parameters: 624,888 Non-trainable parameters: 0		

As we can see the total number of hyper-parameters are quite low compared to complex deeper neural networks which enabled this networked to be trained on CPU only.

The final experiment included dataset of around 2000 images in each class (roughly total of 52000 images) which was then split into 7:1 training is to testing ratio. The dimension of the images were reduced to 64x64.

3.2.1. CNN MODEL CONSTRAINTS AND TRAINING:

3.2.1.1. LOSS FUNCTION:

A loss function (or objective function, or optimization score function) is one of the two parameters required to compile a model.

The loss function used for training is called categorical cross entropy function [2] which is defined as following:

$$H(p,q) = -\sum_{x} p(x)\log(q(x))$$

Where

p = True Distribution

q = Approximated Distribution

This formula corresponds to computing the neg-log-probability of the correct class.

When we develop a model for probabilistic classification, we map the model’s input to probabilistic prediction. Then during the training process the model weights are adjusted in order to bring the probabilistic predictions to brings as close as to the ground truth.

For example:

The ground truth for 5 class classification may look the [1.0 0.0 0.0 0.0 0.0]T and the prediction probabilities after few training epochs can be [0.98 0.02 0.04 0.3 0.1]T. The loss in this example would be -log(0.98).

When using the categorical cross-entropy loss, your targets should be in categorical format (e.g. if you have 10 classes, the target for each sample should be a 10-dimensional vector that is all-zeros except for a 1 at the index corresponding to the class of the sample).

3.2.1.2. OPTIMIZERS:

Optimization algorithms helps us to minimize (or maximize) an Objective function (another name for Error function) E(x) which is simply a mathematical function dependent on the Model’s internal learnable parameters which are used in computing the target values(Y) from the set of predictors(X) used in the model. For example—we call the Weights(W) and the Bias(b) values of the neural network as its internal learnable parameters which are used in computing the output values and are learned and updated in the direction of optimal solution i.e. minimizing the Loss by the network's training process and also play a major role in the training process of the Neural Network Model. The optimizers implemented here are as follows:

3.2.1.2.1. STOCHASTIC GRADIENT DESCENT:

Stochastic Gradient Descent (SGD) **[5]** on the other hand performs a parameter update for each training example. It is usually much faster technique. It performs one update at a time.

$\theta=\theta-\eta\cdot\nabla J(\theta; x(i); y(i))$, where $\{x(i), y(i)\}$ are the training examples.

Now due to these frequent updates, parameters updates have high variance and causes the Loss function to fluctuate to different intensities. This is actually a good thing because it helps us discover new and possibly better local minima, whereas Standard Gradient Descent will only converge to the minimum of the basin as mentioned above.

But the problem with SGD is that due to the frequent updates and fluctuations it ultimately complicates the convergence to the exact minimum and will keep overshooting due to the frequent fluctuations. Although, it has been shown that as we slowly decrease the learning rate-η, SGD shows the same convergence pattern as Standard gradient descent.

The results of training the CNN model on stochastic gradient descent optimization technique using a Softmax classifier for the top layer of the network.

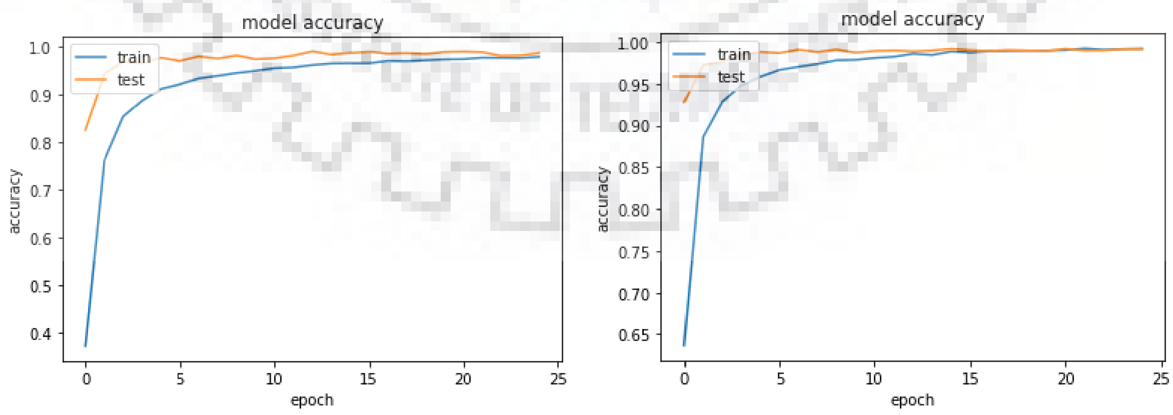


Figure 3.4 Training accuracies for learning rate 0.01(left) and 0.05(right)

3.2.1.2.2. ADAM OPTIMIZER:

Adam stands for Adaptive Moment Estimation. Adaptive Moment Estimation (Adam) [7] is another method that computes adaptive learning rates for each parameter. In addition to storing an exponentially decaying average of past squared gradients like AdaDelta, Adam also keeps an exponentially decaying average of past gradients M(t), similar to momentum:

M(t) and V(t) are values of the first moment which is the Mean and the second moment which is the uncentered variance of the gradients respectively. Similarly, the model was trained using the Adam optimizer for varying learning rates on the same dataset and following are the accuracy curves after training.

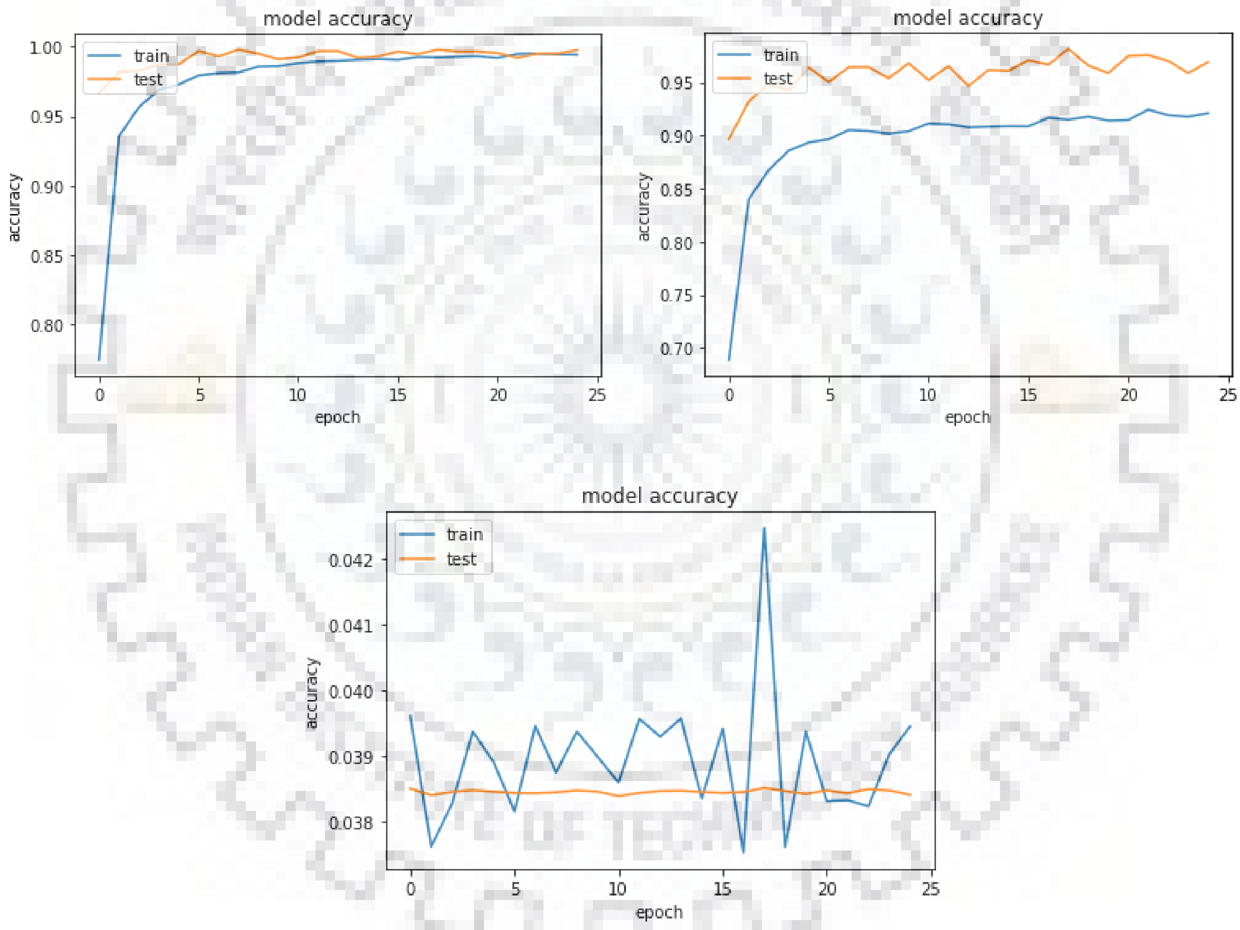


Figure 3.5 Accuracy curves after model was optimized using Adam with varying learning rates 0.001(top left) , 0.01(top right) and 0.05(bottom)

The model performed best while using the Adam optimizer with a learning rate of 0.001 and it was also observed that the model failed to converge producing disastrous results while using a learning rate of 0.05. While generating results for both these optimizers Softmax activation was used on the top layer that is responsible for classification. Softmax classifier will be discussed in the next subsection.

There are other well known optimizers but the two most commonly used ones were described to give an idea about how the optimization algorithm runs the backpropagation for feature weight update at every step.

3.2.1.3. CLASSIFIERS:

3.2.1.3.1. LOGISTIC REGRESSION CLASSIFIER (Sigmoid Classifier):

Logistic regression is a classification algorithm used to assign observations to a discrete set of classes. Unlike linear regression which outputs continuous number values, logistic regression transforms its output using the logistic sigmoid function to return a probability value which can then be mapped to two or more discrete classes.

In the two-class logistic regression, the predicted probabilities are as follows, using the sigmoid function:

$$\Pr(y_i = 0) = \frac{e^{-\beta_0 x_i}}{1 + e^{-\beta_0 x_i}}$$

$$\Pr(y_i = 1) = 1 - \Pr(y_i = 0)$$

So we implemented this classifier to our model using Adam optimizer with learning rates 0.001.

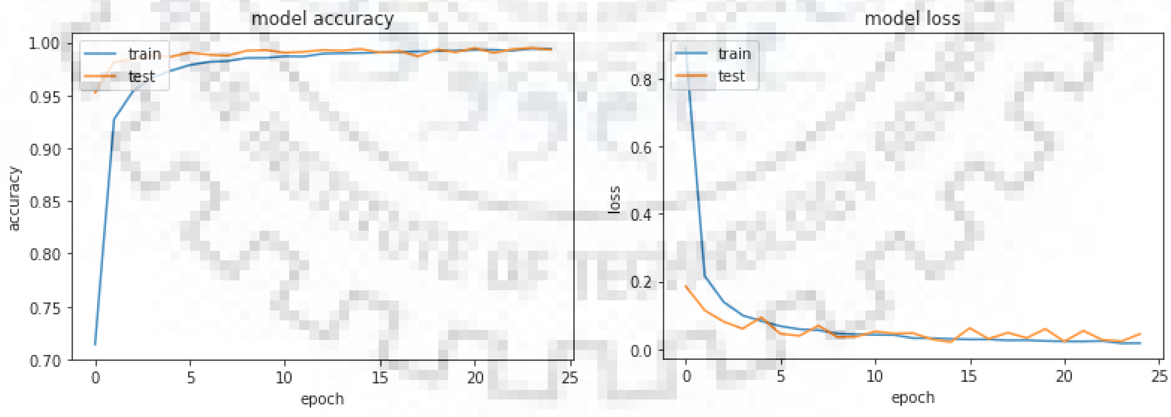


Figure 3.6 Accuracy and Loss of model using Stochastic Gradient Descent Classifier

It is observed that there was not much difference in the results from that of using other classifiers so its upon user to which classifier they see best fit.

3.2.1.3.2. SOFTMAX CLASSIFIER:

In mathematics, the Softmax function [5], or normalized exponential function, is a generalization of the logistic function that "squashes" a K-dimensional vector ‘z’ of arbitrary real values to a K-dimensional vector ‘σ(z)’ of real values, where each entry is in the range (0, 1), and all the entries add up to 1. The Softmax function is given by:

$$f_j(z) = \frac{e^{z_j}}{\sum_k e^{z_k}}$$

Softmax classifier, which has a different loss function. If you’ve heard of the binary Logistic Regression classifier before, the Softmax classifier is its generalization to multiple classes. Unlike the SVM which treats the outputs $f(x_i, W)$ as (uncalibrated and possibly difficult to interpret) scores for each class, the Softmax classifier gives a slightly more intuitive output (normalized class probabilities) and also has a probabilistic interpretation that we will describe shortly. In the Softmax classifier, the function mapping $f(x_i, W) = W*x_i$ stays unchanged, but we now interpret these scores as the unnormalized log probabilities for each class and replace the hinge loss with a cross-entropy loss that has the form:

$$L_i = -f_{y_i} + \log \sum_j e^{f_j}$$

Since Softmax classifier is more advanced than the logistic regression classifier we performed the training of the model using different optimizers, loss functions whose results were discussed above.

3.3. DYNAMIC ISL GESTURE EXPERIMENTS:

3.3.1. FEATURE EXTRACTION:

3.3.1.1. CNNs AND TRANSFER LEARNING:

Image classification [5] problems are solved more easily by the use of deep learning. Convolutional neural networks [29] has paved the pathway for state of the art results using transfer learning [11] for image classification problems and several other applications related to computer vision. Transfer learning is a popular method to build models in a time saving manner and applications where the computational cost needs to be lowered due to unavailability of hardware or it is not feasible to use high specification graphics drivers. In case of transfer learning the model learns basic patterns while solving a different problem hence they are not required to start from scratch. If a robust model is trained on a large benchmark dataset to solve a similar problem relatively then such a model can be considered as a pre-trained model. It is very common to import and use such models from published literature and fine tune them to solve specific problems. The most popular models used for transfer learning are the ImageNet models that were trained on ImageNet data and published state of the art results. Pre-trained models usually are large CNNs [2] which offer high performance and easy training on a wide variety of computer vision tasks. Convolution neural networks consist of two basic parts:

- Convolution base [2]: this part basically consists of stack of convolutional and pooling layers. These layers are mainly responsible for generation of features from the input images that will help in classification task.
- Classifier base [2]: It is composed of fully connected layers which consist of full connections to all the activations of the previous layers. The main purpose of this part is to classify an image based on the detected features using a classification algorithm like logistic regression or Softmax as activation.

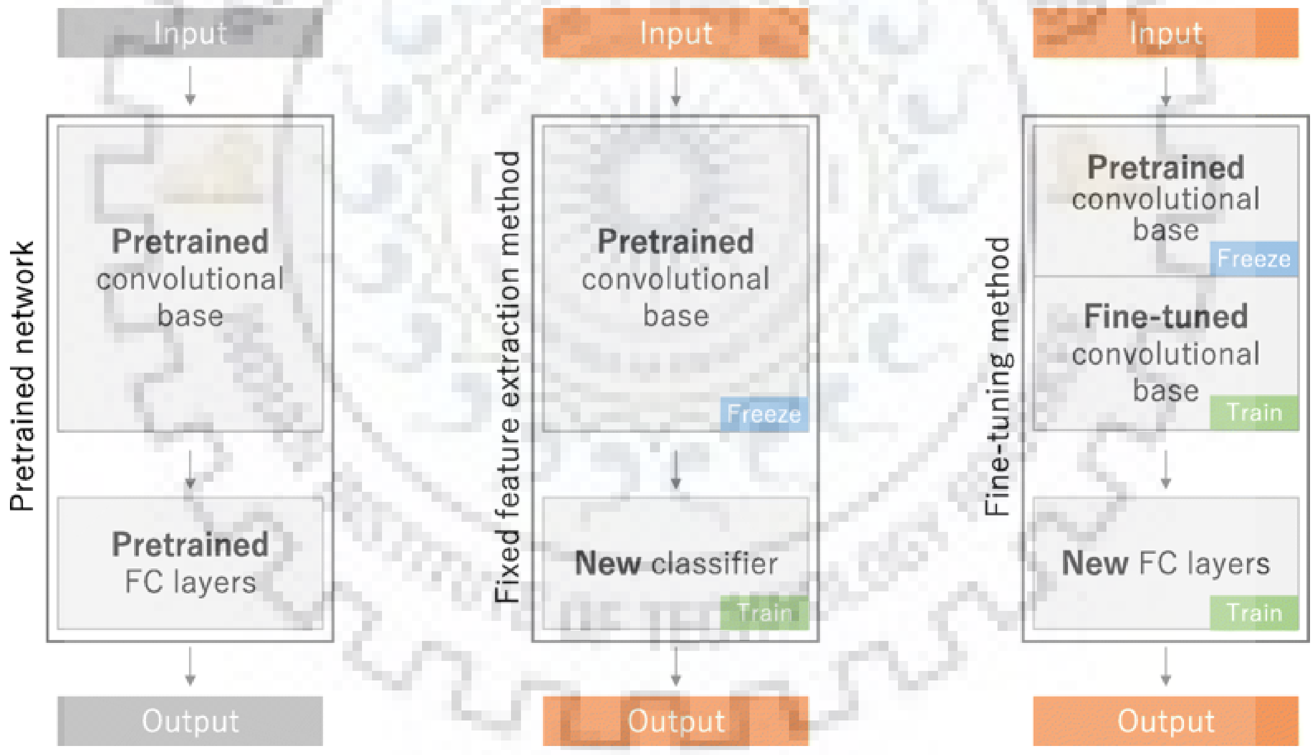
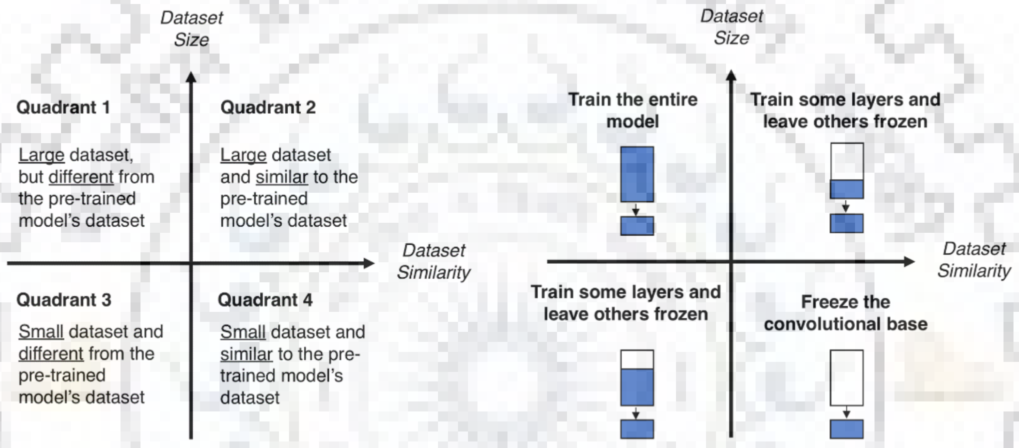

Input
Pretrained network
Pretrained convolutional base
Pretrained FC layers
Output
Input
Fixed feature extraction method
Pretrained convolutional base
Freeze
New classifier
Train
Output
Input
Fine-tuning method
Pretrained convolutional base
Freeze
Fine-tuned convolutional base
Train
New FC layers
Train
Output

Figure 3.7 Transfer learning training strategies on Convolutional Neural Network models [43]

Deep learning models can learn hierarchical feature representations that is the first layers of the networks compute general features which can be efficiently used in different problem domains. The higher layer features which are the top layers can compute specialised features that will mostly depend on the task and the dataset. Thus, we can repurpose a pre-trained model in any of the three ways according to the need of our problem.

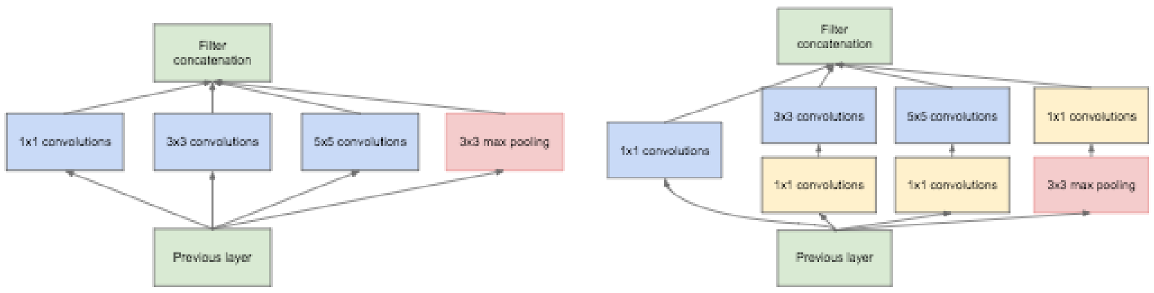
- We can train the whole network from scratch with the new dataset for a different task to enable it to learn features from scratch which might become computationally very expensive.
- Training some layers or all the layers of the convolutional base since the convolution base layers can learn general features that can be used in a variety of problem domains.
- Freezing all the convolution base layers and training the classifier base layers only but this can be done if the dataset is comparatively small with large number of parameters to prevent overfitting.



Dataset Size
Quadrant 1
Large dataset, but different from the pre-trained model's dataset
Quadrant 2
Large dataset and similar to the pre-trained model's dataset
Dataset Similarity
Quadrant 3
Small dataset and different from the pre-trained model's dataset
Quadrant 4
Small dataset and similar to the pre-trained model's dataset
Dataset Size
Train the entire model
Train some layers and leave others frozen
Dataset Similarity
Train some layers and leave others frozen
Freeze the convolutional base

Figure 3.8 Training strategies depending on size of dataset

Now in this paper our aim is to classify videos for which we use CNN to detect spatial features and RNNs [1] to detect temporal features of our training data. Since we are not going to build up our model from scratch we will retrain Google's Inception V3 [2] network on our training data. We will load the pre-trained ImageNet weights of the network and forward pass our data to obtain feature vectors. Now to obtain these feature vectors we save the output of a couple of layers to the disk. We remove the top classification layer and end up with a 2048 vector of features which will serve as the input to RNN after converting them to a sequence of features. From each video frames were extracted, we loop through each of these frames in chronological order and add to a queue. Then we pop off the first frame we added in order to obtain a sequence of features. We can start training our RNN after we get the sequence of features from each and every frame of each video.



(a) Inception module, naïve version (b) Inception module with dimension reductions

Figure 3.9 Inception Module in Block Diagrams [2]

3.3.1.2. INCEPTION V3 MODEL:

For transfer learning application we chose a 42 layer deep neural network with a complexity similar to that of VGGNet [32] called the Inception V3. If we compare the number of trainable parameters among other popular ImageNet models we can observe that AlexNet consists of 60 million parameters, VGGNet cotains as high as 180 million parameters whereas GoogleNet(Inception model) consist of only 7 million parameters which reduces the computation cost drastically when it is used compared to other ImageNet models. The Inception model was first published in Computer Vision and Pattern Recognition 2016 conference and it has more than 2000 citations which makes it evident that using Inception models for transfer learning applications us very common among deep learning researchers. It had a very low error rate which made it the first runner-up for the ImageNet Large Scale Visual Recognition Competition (ILSVRC) in 2015. ImageNet is a dataset that consist of 22000 categories of 15 million labelled high-resolution images. ILSVRC is a subset of around 1000 categories containing roughly 1000 images each. It contains 1.2 million images for training, 50000 for cross-validation and 100000 for testing set. The Inception V3 talks about factorization ideas. The aim of factorizing convolutions is to reduce the number of parameters without decreasing network efficiency.

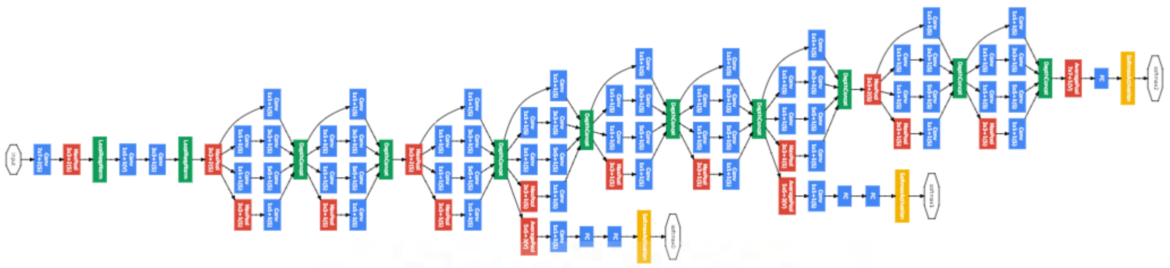


Figure 3.10 Inception V3 Network architecture shows how deep neural networks should be developed from scratch depending upon application [2]

3.3.2. SEQUENCE LEARNING AND VIDEO CLASSIFICATION:

Understanding sequences [1] cannot be done singularly by using CNNs or other shallow fully connected neural network architectures. This requires the help of RNNs. Since RNNs have loops in them information is allowed to persist in them. Hence, they understand sequences by referring to that previous information. The loop-like RNN structure can be thought of as multiple copies of the same network passing on the information to its successor if we unfold the loop a chain like structure unfolds which helps us visualize the passage of sequence information.

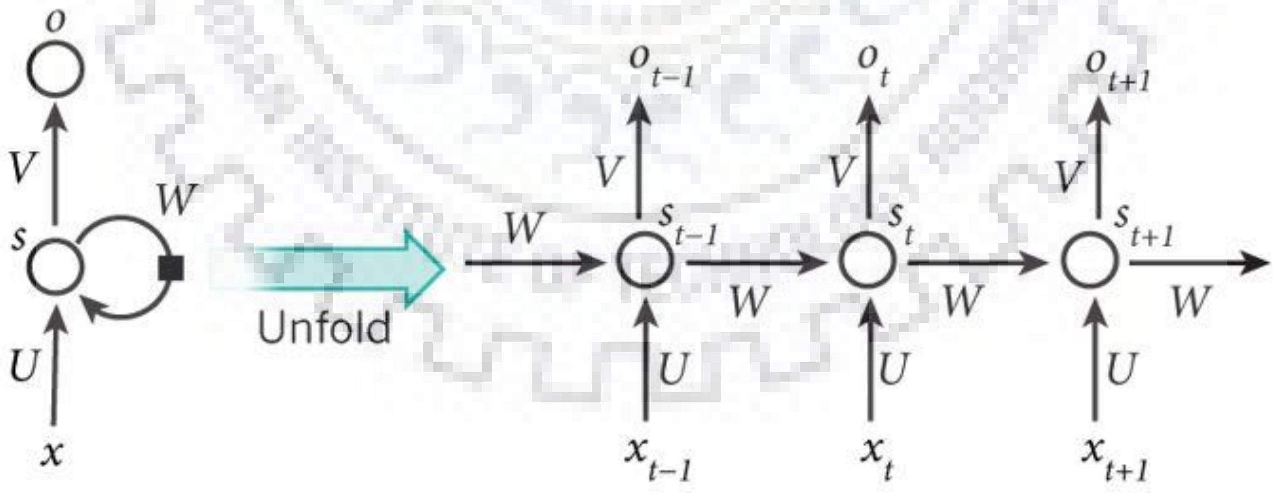


Figure 3.11 Unfolding of a RNN unit repeating module shows it is almost like an extended convolution network except for the feedback connections [44]

The models that were developed for the classification of ISL gestures corresponding to the IIITA ROBITA ISL Gesture Dataset involves the use of RNNs and much deeper networks known as LSTMs which are discussed in the next section in detail.

3.3.2.1. CNN-LSTM MODEL:

3.3.2.1.1. LONG TERM DEPENDENCIES:

Let us take an example to understand long term dependencies from text prediction. Let us consider someone describes themselves through text in the following manner, "I grew up in France…….. I speak fluent French". If we used a machine learning model to predict what language the person was going to say we have to use RNNs. But even for that we need the context of "France" from previous phrases in the text and it might be possible that information might be lost and RNN is not able to learn to connect to such information. Hence, for long term dependencies we need to use an advanced or deeper RNN architecture that is termed as LSTM [1].

3.3.2.1.2. LSTM NETWORKS:

The basic difference between LSTM and RNN module is that the repeating module in a standard RNN contains only one single layer that uses tanh activation. In case of LSTM repeating module there are four layers interacting in a special way in order to retain long term dependencies. Below is a pictorial representation of the repeating module of an LSTM that consists of a 4-layer network. Thus we can easily figure it out that the LSTM structure is much more complex than a standard RNN.

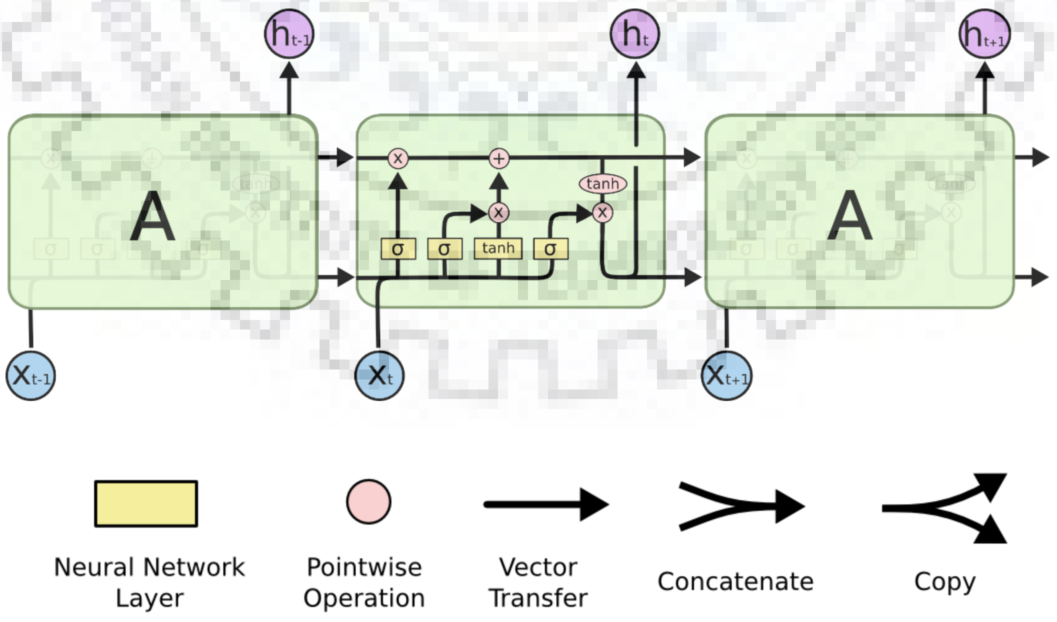


Figure 3.12 Repeating module of LSTM layer

In the above diagram, each line carries an entire vector, from the output of one node to the inputs of others. The pink circles represent pointwise operations, like vector addition, while the yellow boxes are learned neural network layers. Lines merging denote concatenation, while a line forking denote its content being copied and the copies going to different locations.

The key to LSTM repeating module is the cell state which is a horizontal line passing through the top of the diagram which involves simple linear interactions. It is possible for information to pass through cell states unchanged. The LSTM has got the ability to decide if it wants to add or remove information using gates. These are composed of a sigmoid layer and a pointwise multiplication operation.

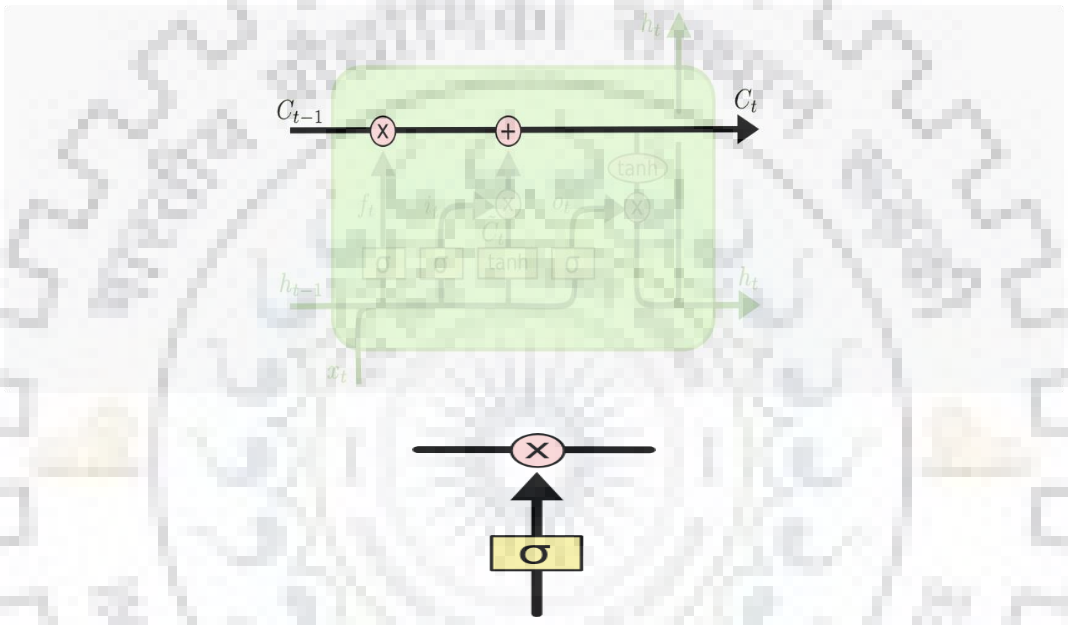


Figure 3.13 Cell state feed forward path inside LSTM repeating module

The sigmoid layer outputs numbers ranging from zero to one indicating how much of the information component it will let through. This is termed as the forget gate layer which is responsible for deciding how much information to throw away. If the output of the forget gate layer is given by f_t and W_f are the weights associated with the neurons of the forget gate layer we can write the following equation:

$$f_t = \sigma(W_f(h_{t-1}, x_t) + b_f)$$

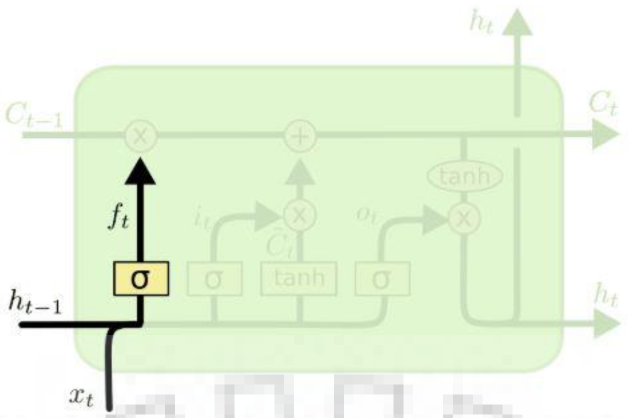


Figure 3.14 Forget Gate Layer inside LSTM repeating module

Where b_f are the biases concerned with the forget gate layer and x_t and h_{t-1} are inputs and outputs of the LSTM module respectively. It is to be noted that h_{t-1} is the output of the previous module that has been fed into the next module. The next step is to find out what new information needs to be added to the cell state. A part of this is done using the input gate layer which is also a sigmoid layer that decides what values we need to update. The other part is a tanh layer that creates a vector of new candidate values which could be added to the cell state. These two are combined together to create an update to the cell state.

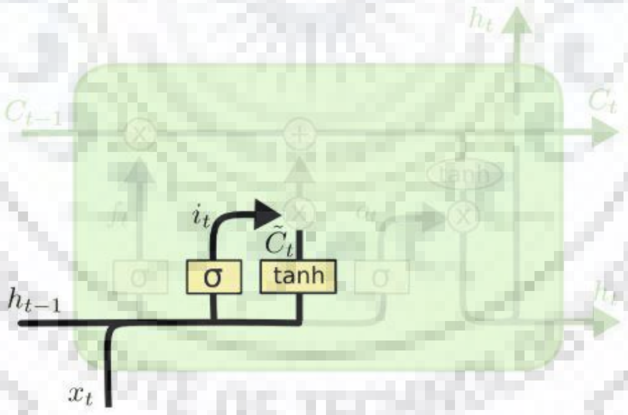


Figure 3.15 Input gate layer and tanh layer inside LSTM repeating module

$$i_t = \sigma\left(W_i\left(h_{t-1,}x_t\right) + b_i\right)$$

$$\widetilde{C_t} = \tanh(W_C(h_{t-1}, x_t) + b_C)$$

Its now time to update the old cell state C_{t-1} to its new state C_t. In order to do that we multiply the old cell state with the forget layer output which denotes how much of the old values we intend to forget and then we add it to the new candidate values scaled by how much we decide to update each state value giving us the updated new state.

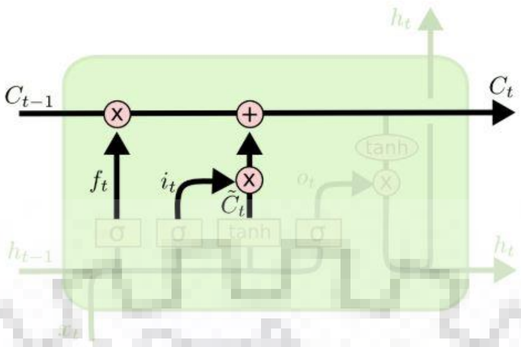


Figure 3.16 Updating the old cell state

$$C_t = f_t * C_{t-1} + \widetilde{C_t} * i_t$$

Hence we finally decide what the new cell state will be depending upon the filtered version of the old cell state. Now its time to decide what we are going to output from the LSTM layer. First, we are going to decide what parts of the cell state we are going to output by running it through a sigmoid layer. Then we put the cell state through a tanh layer to push its values between -1 to +1 and multiply it by the output of the sigmoid layer so that we output only what we decided to.

$$o_t = \sigma(W_o(h_{t-1}, x_t) + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

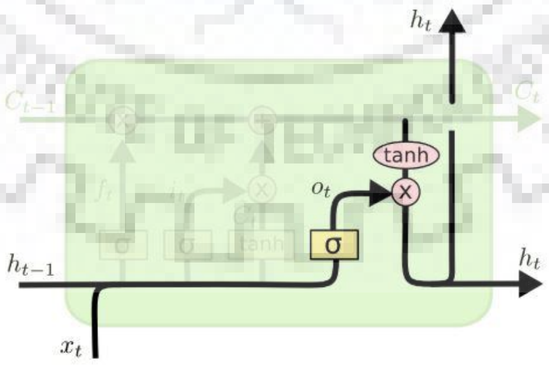


Figure 3.17 Getting the output after cell state update

This was in brief the mathematics behind the LSTM layer and how it retains sequence information can be easily understood from it. After extracting the sequence features using a pre-trained inception V3 on ImageNet weights we pass these sequence vectors on to 4096 wide LSTM layer, followed by a 1024 wide dense layer and a dropout layer in between. The last layer which is the classification layer is another dense layer with 11 neurons as output and Softmax activation which serves as our classifier. The model architecture is as follows:

Table 3.3 LSTM Model Architecture

Layer(type)	**Output Shape**	**Hyper-parameters**
LSTM	(None, 2048)	33562624
Dense	(None, 512)	1049088
Dropout	(None, 512)	0
Dense	(None, 11)	5643
Total parameters: 34,617,355 Trainable parameters: 34,617,355 Non-trainable parameters: 0		

As we can see the number of trainable parameters is huge which increases the computational cost and gives us very good classification results.

3.3.2.2. CNN-MLP MODEL:

An MLP [33] constitutes the most basic and traditional deep learning architectures. These architectures are also known as fully connected networks as all the neurons of this network of a particular layer are connected to all the neurons of the previous and next layers simultaneously. If L is the number of layers then the i^{th} layer of the network l_i will be connected to all the neurons of the layer l_{i-1} and all these connections are modelled using weights in a neural network. A general form of applying a non-linearity to an input series can be governed by the following equation:

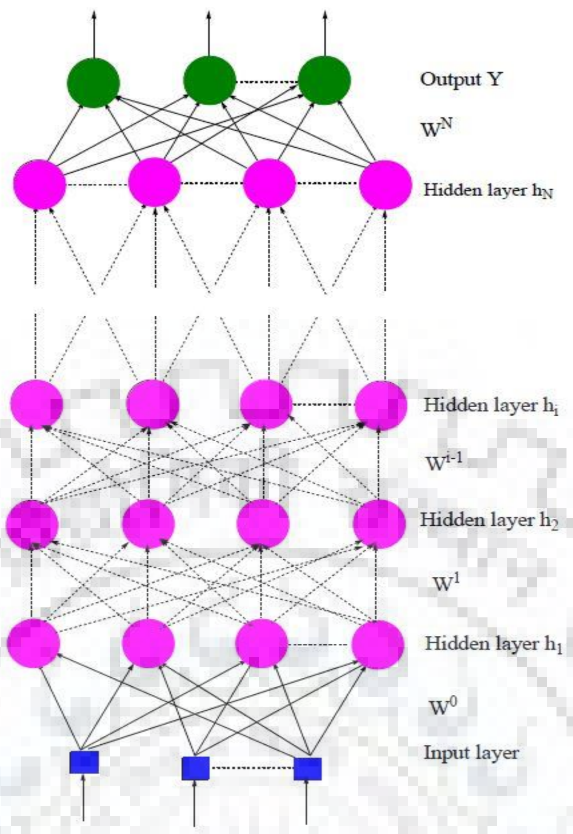


Figure 3.18 Feed forward MLP architecture [33]

$$A_{l_i} = f(W_{l_i} * X + b)$$

Where W_l are the weights that have got dimensions similar to the input features X, b being the bias term and A_l is the activations of the neurons of the layer l. The total number of neurons in a layer are considered as hyperparameters of that layer. The final layer of a multilayer perceptron consist of a total number of neurons equal to that of the number of classes using Softmax activation. Backpropagation is applied to help optimize the multilayer perceptron models. Time series classification [34] using MLPs are a bit difficult as each time stamp has its own weight and temporal information is lost in this way since time series elements are treated independently from each other. The MLP architecture that has been implemented in this paper for video classification has 1 input layer, 4 hidden layers and a Softmax out layer responsible for video classification.

In a similar way as the previous method features were extracted from each frame of each video of the training data but instead of sending sequence of features to the RNN like we did previously we will flatten the sequence and pass the new 2048*40 vector into the fully connected network aka our MLP model. MLPs cannot detect spatial invariance hence the idea is that the MLP will be able to infer temporal features without realizing it as a sequence at all. Since we are using fully connected networks the number of trainable parameters will be very high and is shown as follows in the model summary.

Table 3.4 MLP Model Parameters

Layer(type)	**Output Shape**	**Hyper-parameters**
Flatten	(None, 81920)	0
Dense	(None, 512)	41943552
Dropout	(None, 512)	0
Dense	(None, 512)	262656
Dropout	(None, 512)	0
Dense	(None, 11)	5643
Total parameters: 42,211,851 Trainable parameters: 42,211,851 Non-trainable parameters: 0		

3.3.2.3. LRCN MODEL:

In the paper by Donahue et al [1] discusses a different class of recurrent convolutional networks suitable for large scale visual understanding tasks using end-to-end training of entire architecture and shows the value of such models for activity recognition, image captioning and video description. They also compared RGB and optical flow as input choice and found that a weighted scoring of predictions based on both inputs was the best. This model takes temporal features from videos into consideration. Hence this model takes in images as input unlike the other two models which used sequence of features vector as input.

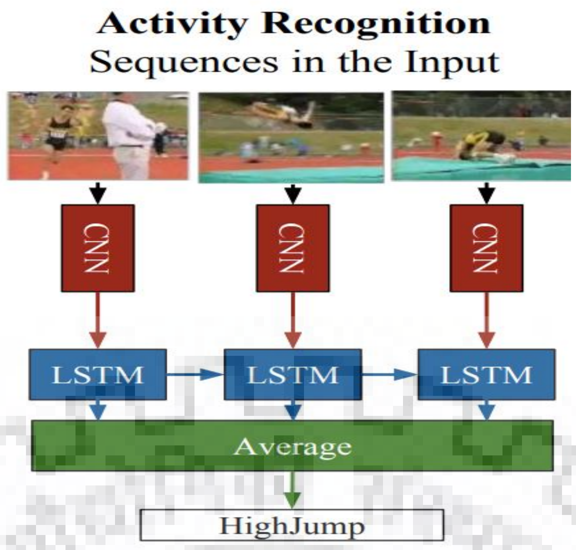


Figure 3.19 LRCN model block diagram used for activity recognition [1]

For building such a network in Keras deep learning framework it becomes very easy due to the availability of Time distributed wrapper which allows us to distribute the layers of the CNN across an extra dimension that is time. Unlike the previous methods where we got to use the pre-trained ImageNet weights, we'll have to train the whole model on our data from scratch here. This could mean that we'll require too much data or a much bigger network to achieve the same level of accuracy as the Inception V3 produced. However, it also means that our CNN weights will be updated on each backprop pass along with the RNN which would be very helpful in solving classification problems of a specific domain. The video data is actually 5 dimensional which consist of (sample, time, length, width, channel) dimensions. Now normally a Keras CNN layers like Conv2D or MaxPooling can work with data having 4 dimensions (sample, length, width, channel). Hence, we use Time Distributed CNN layers which is applicable to 4 dimensions as mentioned above. This Time Distributed convolutional layer [1] was applied along time dimension by applying the same layer to each slices of time to give us a 5 dimensional output.

The LRCN model [1] consists of a convolutional base which consists of 5 blocks. These blocks are made up of 2 time distributed Conv2D layers followed by a time distributed MaxPooling layer using Relu activation. The convolution base is then followed by a time distributed flattening and a dropout layer with a dropout rate of 0.5. The output of this dropout layer is fed into a 512 wide LSTM layer which is then fully connected to the final classification layer of 11 classes using Softmax activation. The model architecture is as follows:

Table 3.5 LRCN Model Architecture

Layer(type)	**Output Shape**	**Hyper-parameters**
Time Distributed Conv2D (1)	(None, 40, 40, 40, 32)	4736
Time Distributed Conv2D (2)	(None, 40, 38, 38, 32)	9248
Time Distributed MaxPooling (3)	(None, 40, 19, 19, 32)	0
Time Distributed Conv2D (4)	(None, 40, 19, 19, 64)	18496
Time Distributed Conv2D (5)	(None, 40, 19, 19, 64)	36928
Time Distributed Maxpooling (6)	(None, 40, 9, 9, 64)	0
Time Distributed Conv2D (7)	(None, 40, 9, 9, 128)	73856
Time Distributed Conv2D (8)	(None, 40, 9, 9, 128)	147584
Time Distributed MaxPooling (9)	(None, 40, 4, 4, 128)	0
Time Distributed Conv2D (10)	(None, 40, 4, 4, 256)	295168
Time Distributed Conv2D (11)	(None, 40, 4, 4, 256)	590080
Time Distributed MaxPooling (12)	(None, 40, 2, 2, 256)	0
Time Distributed Conv2D (13)	(None, 40, 2, 2, 512)	1180160
Time Distributed Conv2D (14)	(None, 40, 2, 2, 512)	2359808
Time Distributed MaxPooling (15)	(None, 40, 1, 1, 512)	0
Time Distributed Flatten (16)	(None, 40, 512)	0
Dropout	(None, 40, 512)	0
LSTM	(None, 256)	787456
Dense	(None, 11)	2827
Total params: 5,506,347 Trainable params: 5,506,347 Non-trainable params: 0		

It can be seen from the above table that the number of trainable parameters are much less as compared to our previous models hence require less computations thus easing the GPU hardware limitations one might face but it does not give results as good as the previous two networks which will be discussed in detail later.

3.3.3. TRAINING:

The input to the first two models were sequence of features vector of size 2048. Whereas images of shape (80, 80, 3) served as inputs to the LRCN model. All the three models were compiled using Adam as optimizer whose learning rate was set at 10^{-5} and decay rate was set at 10^{-6}. The loss function that was used to evaluate the model loss was "categorical cross-entropy". Setting all these parameters training was completed using a 4GB NVIDIA GEFORCE 940MX GPU and the models were trained comparatively faster due to the small size of the dataset and only 11 classes were required to train. For larger datasets with a significantly higher number of classes it would require very high specification GPU hardware for training purposes which will take up a considerable amount of time.

CHAPTER 4

RESULTS AND ANALYSIS

4.1. ASL ALPHABETS CLASSIFICATION RESULTS:

A real time prediction of sign languages was performed after training the model using Adam optimizer with a learning rate of 0.001 using Softmax classifier. Now since we have 24 classes to predict because of the 24 alphabets we performed the gestures corresponding to each class for all classes in a single trial. This process was continued for 15 trials and after that the results of each trial and each gesture was noted down be it true or false which gave us an idea of how our model would perform in real time.

In the following table the results to this experiment are shown and the corresponding accuracy for each class in real time sign prediction was calculated. Each correct prediction is given by "T" and wrong predictions are given by "F".

Table 4.1 Real time Sign language Prediction in 15 trials

Classes	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Accuracy
A	T	T	T	T	T	F	T	T	T	T	T	T	T	T	T	0.93
B	T	T	T	T	T	T	T	T	T	T	F	T	T	T	T	0.93
C	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	1.00
D	T	T	T	T	T	T	F	T	T	T	T	T	F	T	T	0.86
E	F	T	T	T	F	T	F	T	T	F	T	T	F	T	T	0.66
F	T	T	T	T	F	T	T	T	T	T	T	T	T	T	T	0.93
G	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	1.00
H	T	T	T	T	T	F	T	T	T	T	T	T	T	T	T	0.93
I	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	1.00
K	T	F	T	T	T	T	T	F	T	F	F	T	T	T	T	0.73
L	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	1.00
M	T	T	F	T	T	T	F	F	T	T	T	T	T	T	T	0.80
N	T	T	T	T	F	T	T	T	F	T	T	T	T	T	F	0.80
O	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	1.00
P	F	T	T	F	F	T	T	F	T	T	T	T	F	T	T	0.73
Q	T	T	T	T	F	T	T	T	T	F	T	T	T	T	T	0.86
R	T	T	T	T	T	T	T	T	T	T	F	T	T	T	T	0.93

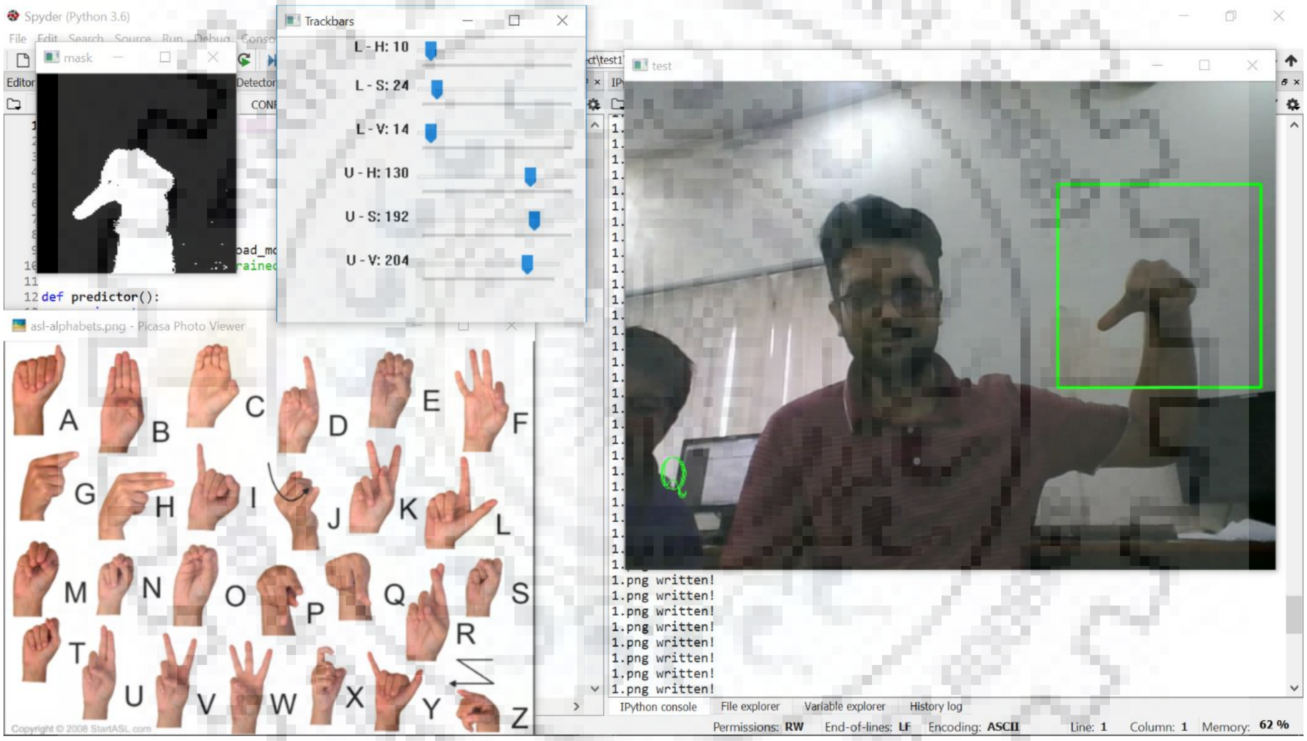
S	F	T	T	T	F	T	F	T	T	T	T	F	T	T	T	0.73
T	F	T	T	T	T	T	F	T	T	T	T	T	F	T	T	0.80
U	T	T	T	T	T	T	T	T	T	T	F	T	T	T	T	0.93
V	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	1.00
W	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	1.00
X	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	1.00
Y	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	1.00

Now let us consider only those cases that gave us errors that is false predictions and try to analyse the false predictions that were made which would give us a clear idea about our model and also the dataset. Some classes were very close to absolute accuracy whereas some classes did not perform that well. In the column of false predictions the number of cases for each false prediction is mentioned in parentheses.

Table 4.2 Classes that were mis-classified and their false predictions

Serial No.	Classes	False Cases	Class of False Predictions (Count of wrong prediction)
1.	A	1	S(1)
2.	B	1	E(1)
3.	D	2	R(2)
4.	E	5	F(1), M(2), D(2)
5.	F	1	W(1)
6.	H	1	G(1)
7.	K	4	V(2), F(2)
8.	M	3	E(2), N(1)
9.	N	3	M(2), E(1)
10.	P	4	G(1), Q(3)
11.	Q	2	A(1), G(1)
12.	R	1	V(1)
13.	S	4	T(2), M(2)
14.	T	3	M(2), S(2)
15.	U	1	Z(1)

The classes that were predicted falsely was because of the reason that the dataset contained hand gestures of only one user. To improve real time prediction the training data should comprise of more diverse data that is data should be generated by using more number of users. Here is an instance of real time prediction shown below that was done thanks to the OpenCV Python library which was used to create the real time prediction interface on ANACONDA platform which supports almost all deep learning frameworks and libraries. It can be seen HSV trackbars were used to mask the hand portion by setting the HSV values which will depend a lot on background, different lighting conditions and different skin tones. The computer webcam served as data acquisition device that captured images at the rate of 30fps. A portion of the webcam feed was cropped that contained the portion where the hand gestures were performed which served as input image to our model for real time prediction. The result of the prediction is displayed on the bottom left corner of the feed and the resulting alphabet is displayed in green font.



Spyder (Python 3.6)
mask
Trackbars
test
asl-alphabets.png - Picasa Photo Viewer
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Q
IPython console
File explorer
Variable explorer
History log
Copyright © 2008 StartASL.com

```
def predictor():
```

L - H: 10
L - S: 24
L - V: 14
U - H: 130
U - S: 192
U - V: 204

```
1.png written!
1.png written!
1.png written!
1.png written!
1.png written!
1.png written!
1.png written!
1.png written!
```

Permissions: RW End-of-lines: LF Encoding: ASCII Line: 1 Column: 1 Memory: 62 %

Figure 4.1 Demonstration of the alphabet Q

4.2. ISL WORDS CLASSIFICATION RESULTS:

The ISL Gesture dataset obtained from IITA Robotics and AI lab contains 11 classes of dynamic gestures. The dataset contained a total of 147 videos which were split into training and test sets. The training set contained a total of 100 videos whereas the test set consists of 47 videos all belonging to 11 classes of dynamic gestures. Three models were applied on the training set and training was performed through an NVIDIA GPU of 4GB 940MX which provided a classification accuracy as high as 100%. The three models that were applied to this data were a multi-layer perceptron model (MLP), Long term recurrent convolutional networks model (LRCN) and an LSTM (Long short-term memory recurrent neural network) model whose features were extracted for each frame of each video gesture using a pre-trained Inception V3 model. The features for the MLP model consisted of the image frames from each video which were extracted before training. After training was done the accuracy was calculated on the test set for each class individually which achieved up to 100% accuracy. Very limited amount of data was used that yielded such good results even though there were a few tricky gestures like "Arise".

4.2.1. RESULTS OF CNN-MLP MODEL:

The MLP model performed excellently well on the dataset used for video classification. It even outperformed the CNN-LSTM model although features for both the models were extracted using the same pre-trained model. The test accuracy reached 100% for all classes of the test data. One of the reasons for getting exceptionally good results is the dataset not being diverse enough which introduces a bias that causes it to classify the test data so well after being trained on the training set data. Although there are doubts over how well the same trained model predict if the gestures to be predicted were not performed using same background. Although CNN-MLP model offers a great solution to dynamic hand gesture recognition over a large scale and more generalised dataset. But such a model would

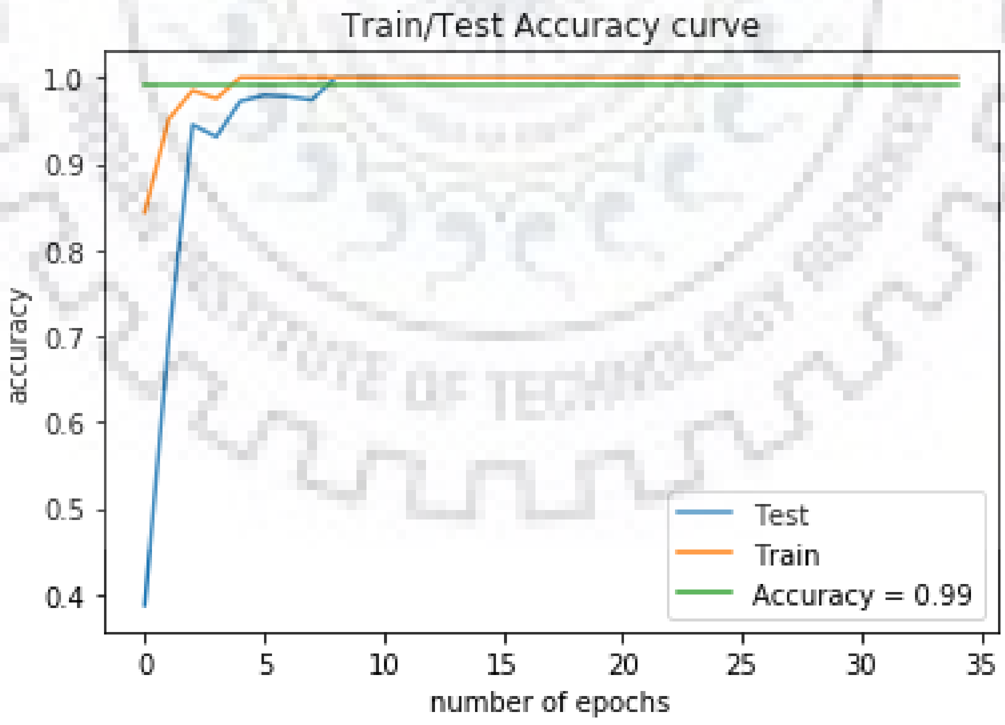


Figure 4.2 Training and testing accuracy plot

require high computational resources if it were to perform on large datasets. The number of trainable parameters of the model was more than 42 million so it has to be kept in mind considering the size of the dataset if it indeed is fruitful to use this method with much more number of classes of gestures if it is worth the computational cost.

4.2.2. RESULTS OF LRCN MODEL:

The LRCN model did not perform as well as the other methods as it is evident from the accuracy curve. The testing accuracy reached around 80% whereas the training accuracy reached 100% which is only possible because of overfitting. The LRCN model had around 5 million parameters and it was trained end to end on the ISL Gesture dataset. LRCN model is best applicable for specific computer vision tasks like sign language recognition. Even after training end to end the number of parameters of this model is quite low compared to other models hence, it can be used wherever there are limitations to computational efficiency of available hardware.

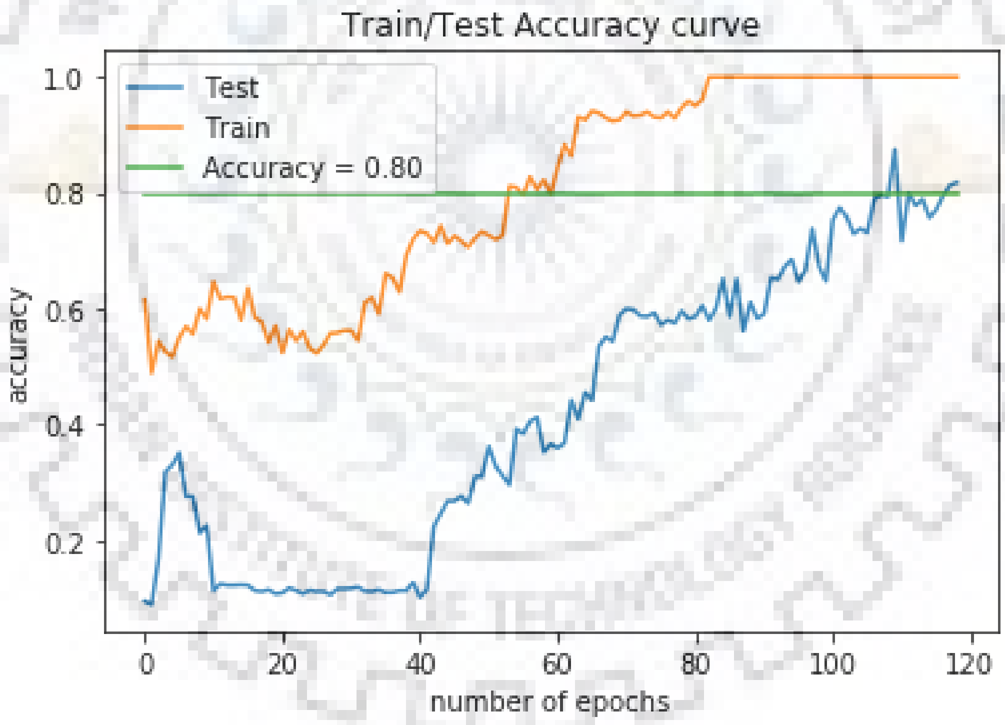


Figure 4.3 Training and Testing accuracy plot

4.2.3. RESULTS OF CNN-LSTM MODEL:

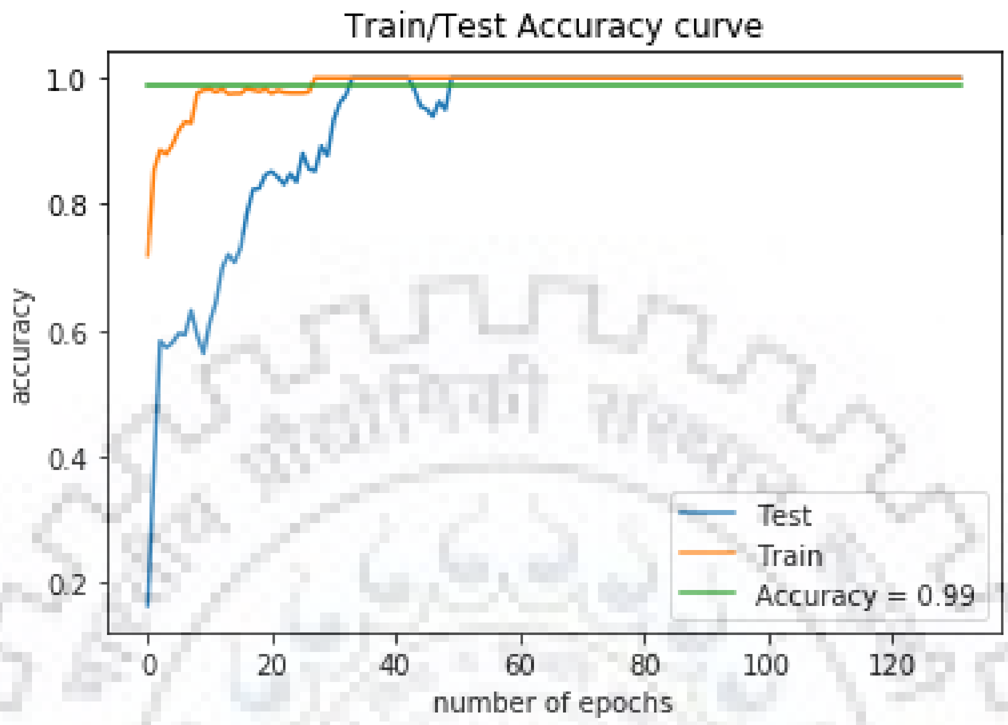


Figure 4.4 Training and Testing accuracy plot

The CNN-LSTM model usually performs very well in activity recognition tasks since it is best suited for sequence learning. CNN base is used to extract spatial features and the LSTM base is responsible for learning the temporal features. From the curve it can be seen that it performs very well on the testing set achieving 100% accuracy which is exceptionally good compared to the classification accuracy that was achieved using statistical machine learning models on the same gesture dataset. This is discussed in detail in the next section. This model had around 34 million hyper parameters which would increase had there been more classes of gesture data or in general if the dataset was large. But then again the requirement for high end GPU hardware would be necessary in order to implement such a model to large datasets. The aim of this project was to compare different methods that would be applicable to gesture recognition problem and test them on smaller datasets to able to learn which models would perform better when a large, diverse and more complex dataset is available. IIT Roorkee Electrical Department has been working on such a dataset that would contain over 100 classes of dynamic gestures and gesture data would be taken from at least 20 users which would further ensure that the dataset will be diverse.

If we look at all three methods on which the ISL Gesture dataset was applied the most computationally expensive was the MLP model but it provided the best accuracy. The CNN-LSTM model performed quite well too and similar networks have worked very well on other activity recognition dataset and it is the safest way to obtain classification over any dataset since it computes both spatial and temporal features using the pre-trained CNN and LSTM respectively. The LRCN model although did not match the accuracy figures of any other model but the main reason behind that is that it is trained end to end so it needs a large, diverse dataset that would be able to properly learn all the features associated with gesture recognition. The ISL Gesture dataset that has been used in this paper is not a large dataset so it is advisable to not use it in algorithms that trains end to end. For such kind of models one should always prefer using transfer learning to extract features as that would always give better results.

4.2.4. RESULT ANALYSIS:

Table 4.3 Table of comparison of results obtained to that of published results

Classes	Euclidean Distance (18 bins)	Euclidean Distance (36 bins)	K-Nearest Neighbour (18 bins)	K-Nearest Neighbour (36 bins)	CNN-MLP Model	LRCN Model	CNN-LSTM Model
Above	99.42	99.81	100	100	100	100	100
Across	100	100	94.71	99.82	100	100	100
Advance	93.83	94.25	97.03	98.63	100	80	100
Afraid	100	100	100	100	100	100	100
All	71.85	79.51	71.58	66.12	100	100	100
Alone	96.39	85.9	77.05	74.1	100	100	100
Arise	68.08	64.42	63.1	61.93	100	75	100
Bag	51.35	65.99	60.02	48.42	100	100	100
Below	86.11	84.47	73.05	70.42	100	100	100
Bring	100	100	100	100	100	100	100
Yes	100	100	100	100	100	100	100

In the paper by A. Nandy et al [22] on the same dataset achieved accuracy as shown in the table above. They used the statistical machine learning algorithms Euclidean Distance and K-Nearest neighbour for 18 bins and 36 bins direction histogram. The 36 bins direction histogram provided very good classification results which was further improved by the deep learning classification models discussed in this report. But the computational cost of using statistical machine learning algorithms was much less compared to that of its deep learning counterparts. Nowadays video classification for action or gesture recognition has gained a lot of pace which was not there before. Hence, the availability of benchmark Sign Language datasets was also scarce before. But nowadays people are working more and more on videos so getting benchmark dynamic sign language datasets will not be a problem for very long. One of such datasets on the ISL is being prepared by Electrical Department of IIT Roorkee. Having contributed on making of the dataset as one of the user it can be said that it will become one of the benchmark dataset on Dynamic ISL gestures representing more than a hundred words. Now the goal has always been to achieve real time translation of sign language and it is a big step towards that to learn as many words as possible. Now the problem is how accurately that can be done in real time. To solve that deep learning models need to be modified to achieve as high accuracy as possible on a very large scale dataset. In this report the models performed exceptionally well for a small dataset with 11 classes so there was no need to fine tune the models for better performance but when applications involve large scale datasets the models need to be tuned and made deeper to extract more complex features. This is what the statistical machine learning models will struggle to achieve. Since its performance kind of saturates with the introduction of more amount of data instead of improving.

CHAPTER 5

CONCLUSION

In this report a study was conducted on real time sign language recognition and translation. This study started with the very basic understanding of any language, that is recognizing gestures used for alphabets of the same language. Recognizing static gestures is very similar to any image classification task. Convolutional Neural Network models have achieved exceptional success in any type of image classification task since the last decade and similarly we got very good results concerned with classifying sign language alphabets. Gradually we moved on to words and phrases and identifying them correctly could lead to successful real time sign language translation. So in order to recognize gestures of different words and classify them successfully we presented three state-of-the art models that has the ability to learn deep spatial and temporal features which can be very useful and used efficiently in any type of computer vision applications that would involve sequential input and output data like activity recognition, image captioning, video descriptions and many others. We have shown that our models have outperformed statistical machine learning models which were used on the same data. This is because deep sequence models (the models that were studied and applied in this report) performs consistently well over other models that do not enforce deep sequence learning. With so much attention now transferred towards Computer Vision applications involving video streams or clips it is safe to say that deep sequence learning will be central to such tasks.

It must also be noted that transfer learning was used to extract features by removing the top two layers of the Inception V3 model. Since the results of classification were exceptionally good for both the CNN-MLP model and CNN-LSTM model it is safe to conclude that the features extracted were robust and more powerful. This concept can be extended for future work too. Our models are already trained on the ISL gesture dataset that means it has generated features involving hand movement. When large scale data is used for sign language recognition involving more than 100 words or phrases it will require more complex and deeper models to be implemented which would mean huge computational resources would be required and it would take a lot of time just to train the network in one go, so much so that researchers opt to train only a few number of classes at a time. Now our models which are already trained on gesture data can be used for transfer learning applications to the video domain. This would mean two things, one is that we won’t have to build the networks from scratch hence, it will save time which can be efficiently used in tuning the model for better performance and the computational cost would be reduced significantly which would mean more classes can be trained on the same GPU. The models that were studied gave very satisfactory results on the ISL gesture dataset but this may not be the case that it generates similar results for other broader and larger scale datasets. For them we need to learn more powerful and generic features. This can be solved by using optical flow of images. Using optical flow is of course not that easy as it increases model complexity and the number of hyper-parameters would increase which would mean high end GPU would be required to perform such a task which was currently not available at our disposal. Pre-processing the input images often lead to learning robust features which would again increase model complexity and was not required on our experiments but when large scale recognition is to be implemented these changes to a sequence model should be considered to achieve state-of-the art results that would surpass the human error rate.

Recognizing and translating sign language in real time without interruption and high accuracy is the future scope of this project which will be extended to sentences and paragraphs. This would ease the communication gap with the deaf/mute community. Other than that humanoid robots can be trained using the same way so they can react in a particular manner to a definite gesture is very big future scope of this project. Another huge aspect for the future is integration of gesture recognition and natural language processing which would pave the way for an even smarter future. Activity recognition and gesture recognition almost go hand in hand so the same deep sequence models can be applied to live video streams to detect violent actions on a large scale which would make our society safer than it already is.

REFERENCES

[1] L. A. H. M. R. S. V. S. G. Jeff Donahue, “Long-term Recurrent Convolutional Networks for Visual Recognition and Description,” *arXiv:1411.4389v4,* pp. 1-14, 2016.

[2] C. W. L. Y. J. P. S. S. R. D. A. D. E. V. V. a. A. R. Szegedy, “Going deeper with convolutions.,” *In Proceedings of the IEEE conference on computer vision and pattern recognition,* pp. 1-9, 2015.

[3] K. a. A. Z. Simonyan, “Very deep convolutional networks for large-scale image recognition.,” *arXiv preprint arXiv:1409.1556,* 2014.

[4] M. S. O.-C. J. R. a. F. S.-I. Rivera-Acosta, “American Sign Language Alphabet Recognition Using a Neuromorphic Sensor and an Artificial Neural Network,” *Sensors 17, no. 10,* p. 2176, 2017.

[5] I. S. G. E. H. Alex Krizhevsky, “ImageNet Classification with Deep Convolutional Neural Networks,” *Neural Information Processing Systems,* pp. 1-8, 2012.

[6] R. F. Matthew D. Zeiler, “Visualizing and Understanding Convolutional Networks,” *ImageNet Large Scale Visual Recognition Competition (ILSVRC),* pp. 1-10, 2013.

[7] X. Z. S. R. J. S. Kaiming He, “Deep Residual Learning for Image Recognition,” *ImageNet Large Scale Visual Recognition Competition (ILSVRC),* pp. 1-10, 2015.

[8] H. A. K. C. S. a. L. C.-L. Wang, “Action recognition by dense trajectories.,” *In CVPR 2011-IEEE Conference on Computer Vision & Pattern Recognition,* pp. 3169-3176, 2011.

[9] I. Laptev, “On space-time interest points.,” *International journal of computer vision 64, no. 2-3 (2005),* pp. 107-123, 2005.

[10] V. R. G. C. a. S. B. Piotr Dollar, “Behavior Recognition via Sparse Spatio-Temporal Features,” *Beijing, China: VS-PETS,* 2005.

[11] H. a. C. S. Wang, “Action recognition with improved trajectories.,” *In Proceedings of the IEEE international conference on computer vision,* pp. 3551-3558, 2013.

[12] S. W. X. M. Y. a. K. Y. Ji, “3D convolutional neural networks for human action recognition,” *IEEE transactions on pattern analysis and machine intelligence 35, no. 1 (2012),* pp. 221-231, 2012.

[13] A. G. T. S. S. T. L. R. S. a. L. F.-F. Karpathy, “Large-scale video classification with convolutional neural networks.,” *In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition,* pp. 1725-1732, 2014.

[14] K. a. A. Z. Simonyan, "Two-stream convolutional networks for action recognition in videos.," *In Advances in neural information processing systems,* pp. 568-576, 2014.

[15] J. M. H. S. V. O. V. R. M. a. G. T. Yue-Hei Ng, "Beyond short snippets: Deep networks for video classification.," *In Proceedings of the IEEE conference on computer vision and pattern recognition,* pp. 4694-4702, 2015.

[16] G. I. L. a. C. S. Varol, "Long-term temporal convolutions for action recognition.," *IEEE transactions on pattern analysis and machine intelligence 40, no. 6,* pp. 1510-1517, 2017.

[17] D. J. R. Z. S. S.-F. C. a. M. P. Tran, "Convnet architecture search for spatiotemporal feature learning.," *arXiv preprint arXiv:1708.05038,* 2017.

[18] C. A. P. a. A. Z. Feichtenhofer, "Convolutional two-stream network fusion for video action recognition," *In Proceedings of the IEEE conference on computer vision and pattern recognition,* pp. 1933-1941, 2016.

[19] L. Y. X. Z. W. Y. Q. D. L. X. T. a. L. V. G. Wang, "Temporal segment networks: Towards good practices for deep action recognition.," *In European conference on computer vision, Springer, Cham,* pp. 20-36, 2016.

[20] R. D. R. A. G. J. S. a. B. R. Girdhar, "Actionvlad: Learning spatio-temporal aggregation for action classification," *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* pp. 971-980, 2017.

[21] Y. Z. L. S. N. a. A. H. Zhu, "Hidden two-stream convolutional networks for action recognition.," *In Asian Conference on Computer Vision, Springer, Cham,* pp. 363-378, 2018.

[22] J. S. P. S. M. P. C. a. G. N. Anup Nandy, "Recognition of Isolated Indian Sign Language Gesture in Real Time," *Springer LNCS-CCIS,* pp. 102-107, March 2010.

[23] P. C. J. S. P. G. C. N. a. S. M. Anup Nandy, "Classification of Indian Sign Language in real time," *International Journal on Computer Engineering and Information Technology(IJCEIT),* pp. 52-57, February, 2010.

[24] S. M. J. S. P. P. C. a. G. Anup Nandy, "Recognizing & Interpreting Indian Sign Language Gesture for Human Robot Interaction," *IEEE XPLORE DIGITAL LIBRARY(PROCEEDING OF ICCCT'10),* pp. 712-717, September, 2010.

[25] D. H. H. a. T. N. Wiesel, "Receptive fields of single neurones in the cat's striate cortex," *The Physiological Society,* pp. 574-591, 1959.

[26] L. G. Roberts, "Machine perception of three-dimensional solids," *Massachusetts Institute of Technology,* pp. 8-36, 1963.

[27] D. Lowe, "Three-Dimensional Object Recognition from Single," *Artificial Intelligence,* pp. 355-395, 1987.

[28] J. M. Jianbo Shi, “Normalized Cuts and Image Segmentation,” *Computer Science Division, University of California at Berkeley,* pp. 2-13, 1997.

[29] L. B. Y. B. P. H. Y. Lecun, “Gradient-Based Learning Applied to Document Recognition,” *PROCEEDINGS OF THE IEEE,* pp. 2-16, 1998.

[30] M. D. a. R. F. Zeiler, “Visualizing and understanding convolutional networks.,” *In European conference on computer vision, springer, Cham,* pp. 818-833, 2014.

[31] A. Z. Karen Simonyan, “VERY DEEP CONVOLUTIONAL FOR LARGE-SCALE IMAGE RECOGNITION,” *International Conference on Learning Representations,* pp. 1-8, 2014.

[32] A. Z. Karen Simonyan, “VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION,” *International Conference on Learning Representations,* pp. 1-12, 2015.

[33] M. A. J. I. Y. G. M. E. Hassan Ramchoun, “Multilayer Perceptron: Architecture Optimization and Training,” *International Journal of Interactive Multimedia and Artificial Intelligence,* pp. 26-30, September 2016.

[34] G. F. J. W. L. I. a. P.-A. M. Hassan Ismail Fawaz, “Deep learning for time series classification: a review,” *Data Mining and Knowledge Discovery,* pp. 1-47, 2019.

[35] W. D. R. S. L.-J. L. K. L. L. F.-F. Jia Deng, “ImageNet: A Large-Scale Hierarchical Image Database,” *IEEE Conference on Computer Vision and Pattern Recognition,* pp. 1-7, 2009.

[36] A. V. Aravindh Mahendran, “Visualizing Deep Convolutional Neural Networks Using Natural,” *International Journal Of Computer Vision,* pp. 1-16, 2016.

[37] K. X. Z. S. R. a. J. S. He, “Deep residual learning for image recognition,” *In Proceedings of the IEEE conference on computer vision and pattern recognition,* pp. 770-778, 2016.

[38] G. Z. L. L. V. D. M. a. K. Q. W. Huang, “Densely connected convolutional networks,” *In Proceedings of the IEEE conference on computer vision and pattern recognition,* pp. 4700-4708, 2017.

[39] M. J. Paul Viola, “Rapid Object Detection using a Boosted Cascade of Simple,” *COMPUTER VISION AND PATTERN RECOGNITION,* pp. 1-7, 2001.

[40] L. K. J. D.-Y. Y. a. B. E. S. Sun, “Human action recognition using factorized spatio-temporal convolutional networks.,” *In Proceedings of the IEEE International Conference on Computer Vision,* pp. 4597-4605, 2015.

[41] D. L. B. R. F. L. T. a. M. P. Tran, “Learning spatiotemporal features with 3d convolutional networks.,” *In Proceedings of the IEEE international conference on computer vision,* pp. 4489-4497, 2015.

[42] S. M. P. C. a. G. Anup Nandy, “Gesture based imitation learning for Human Robot Interaction,” *UACEE International Journal of Artificial Intelligence and Neural Networks ISSN:- 2250-3749(online),* 2010.

[43] Yamashita, Rikiya, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. "Convolutional neural networks: an overview and application in radiology." *Insights into imaging* 9, no. 4: 611-629, (2018)

[44] LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." *nature* 521, no. 7553: 436, 2015