Dynamic Gesture Classification of American Sign Language using Deep Learning

By

Devina Vaghasiya

A thesis submitted in partial fulfillment

of the requirements for the degree of

Master of Science (MSc) in Computational Sciences

The Faculty of Graduate Studies
Laurentian University
Sudbury, Ontario, Canada

THESIS DEFENCE COMMITTEE/COMITÉ DE SOUTENANCE DE THÈSE
Laurentian Université/Université Laurentienne
Faculty of Graduate Studies/Faculté des études supérieures

Title of Thesis Titre de la thèse	Dynamic Gesture Classification of American Sign Language using Deep Learning
Name of Candidate Nom du candidat	Vaghasiya, Devina
Degree Diplôme	Master of Science
Department/Program Département/Programme	Computational Sciences
Date of Defence Date de la soutenance	March 10, 2021

APPROVED/APPROUVÉ

Thesis Examiners/Examinateurs de thèse:

Dr. Kapldrum Passi
(Supervisor/Directeur(trice) de thèse)

Dr. Ratvinder Grewal
(Committee member/Membre du comité)

Dr. Meysar Zeinali
(Committee member/Membre du comité)

Dr. Pradeep Atray
(External Examiner/Examinateur externe)

Approved for the Faculty of Graduate Studies
Approuvé pour la Faculté des études supérieures
Dr. Lace Marie Brogden
Madame Lace Marie Brogden
Acting Dean, Faculty of Graduate Studies
Doyenne intérimaire, Faculté des études supérieures

ACCESSIBILITY CLAUSE AND PERMISSION TO USE

Abstract

American Sign Language (ASL) is a visual method of communication, utilized primarily by the hearing-impaired people. ASL is a sign language with 5 fundamental criterions: state of the hand, location (place of articulation), movement, palm orientation, and facial expressions. Since it is the most well-known gesture-based communication (sign language) of the world, it is essential to address dynamic sign gesture recognition for American Sign Language. To address the static sign language recognition in American Sign language a lot of studies have been done and researchers have claimed approximately 99% accuracy in static sign language recognition. There are very few studies currently available for dynamic gesture recognition in ASL.

In this study, a subset of American Sign Language dataset was used, namely World-Level American Sign Language (WLASL) which has originally more than 2000 classes for gesture-based classification of American Sign Language from which we have chosen 100 classes. A combination of VGG16-LSTM, VGG19-LSTM, ResNet101-LSTM, Inception-LSTM and Inception3D based Convolutional Neural Networks (CNN) models were used for extracting spatial and temporal features respectively and applied them on the processed and extracted classes of videos from WLASL dataset. We found our model Inception3D outperformed the Visual Geometry Group-Long Short-Term Memory (VGG-LSTM) architectures, and ResNet101-LSTM models. These models have been compared based on model evaluation metric accuracy, thereby providing suitable insights on model selections.

Keywords: *American Sign gestures, Deep learning, Convolutional neural network, Long Short-Term Memory (LSTM)*

Acknowledgments

I would like to express my deepest gratitude and appreciation to my supervisor, Dr. K. Passi, who gave me the opportunity and all the possible support to make this research successful. I believe I have got the finest supervisor who has been standing with me throughout the educational journey and he was always available to clear all my doubts and handled all the queries with patience.

I want to thank my technical coordinator, Mark Thompson for providing all the server requirements for training phase. He was always available and supported all the dependencies required in the environments.

I also want to thank my parents, my brother, and all my friends who made my research experience enjoyable.

Finally, and most significantly, I offer special gratitude to my parents for help and persistence and believing in me and trusting my caliber.

Table of Contents

Table of Figures

List of tables

Chapter 1

Introduction

1.1 Sign Language

Communication connects people by conveying messages to one other, expressing their inner sentiments, and interchanging ideas verbally or non-verbally. Nonetheless, the deaf community is incapable of communicating verbally. Sign language was intended to help hearing-impeded communities in expressing their sentiments to others. Communication through sign language involves gestures for communicating by moving the fingers, hands, arms, head, body and outward appearances. Therefore, it tends to be arduous for society to learn and practice sign language. One of the main reasons for these communities with disability (hearing and speech impeded) being isolated from society is the difficulty in learning the sign language by the general population. The Human-computer Interaction (HCI) technology has provided the ease of use of the computer systems to the general public to understand the sign language. Different software/hardware systems have been proposed in recent times to analyze the sign gestures and provide the correct interpretation.

Different countries have their own sign language, for example, American Sign Language (ASL), Indian Sign Language (ISL), French Sign Language (FSL), and Puerto Rican Sign Language (PRSL), to name a few. There are various gesture languages used in the western continent, Table 1 gives information about it. Gesture language is relying on region and it has

important differences from other dialects. It is fundamental to comprehend gesture-based communication when speaking with hard of hearing and their families [1].

Table 1. Sign Language in the Americas [2]

North America	Central America	South America
• American Sign Language •Inuit Sign Language • Quebec Sign Language •Puerto Rican Sign Language	•Costa Rican Sign Language •Guatemalan Sign Language •Honduras Sign Language •Mayan Sign Language •Mexican Sign Language •Nicaraguan Sign Language •Panamanian Sign Language •Salvadorian Sign Language •Tijuana Sign Language	•Argentine Sign Language •Bolivian Sign Language •Brazilian Sign Language •Chilean Sign Language •Colombian Sign Language •Ecuadorian Sign Language •Paraguayan Sign Language •Peruvian Sign Language

1.2 History of American Sign Language (ASL)

In the United States, the formal education of deaf students began in 1817 with the American School's foundation for the Deaf in Hartford, Connecticut [3]. American Sign Language (ASL) use has proliferated generally by schools for the hard of hearing and Deaf people group associations. Despite its wide use, no accurate count of ASL clients has been taken. Dependable evaluations for American ASL people range from 250,000 to 500,000 people, including various offspring of hard of hearing grown-ups [4].

ASL signs have few phonemic components, for example, movement of the face, the torso, and the hands. ASL is not a pantomime(gesture) structure, but iconicity plays a vital role in ASL. American Sign Language syntax (grammar) is unrelated to that of English; English loan words (a loanword is a word embraced from one language and incorporated into another language without interpretation) are often borrowed through fingerspelling. ASL has a

verbal agreement and aspectual marking and has a productive system of forming agglutinative classifiers. Many linguists believe ASL to be a subject–verb–object (SVO) language [5].

The method of instruction was Signed English, which was an attempt in a visual modality to reflect the English structure and punctuation on the hands. It was created with the expectation that they would get this if deaf students had access to the English scheme. The earlier version of Signed English was dependent on Signed French because the French model borrowed the first American instruction. America's first language of instruction was changed to Signed French with some invented signs to represent English parts. In 1835, ASL was the dominant language of instruction for the deaf in schools [6].

ASL is a visual method of communication, used primarily by deaf and hard of hearing people. People use different mediums like hands, face, and eyes instead of vocal tract or ears. Linguistic communication may be a vision-based language that uses various mediums like hand gestures and handshapes to represent concepts or ideas.

The sign language (ASL) used in America is one of the world's well-known communications via gestures, among other sign languages. American Sign Language has five basic criterions: Shape of the hand, Location, Movement, Palm Orientation and Facial Expressions. As we can see in Figure 1 and Figure 2, the above 5 points relate to different letters and numbers of English [7].

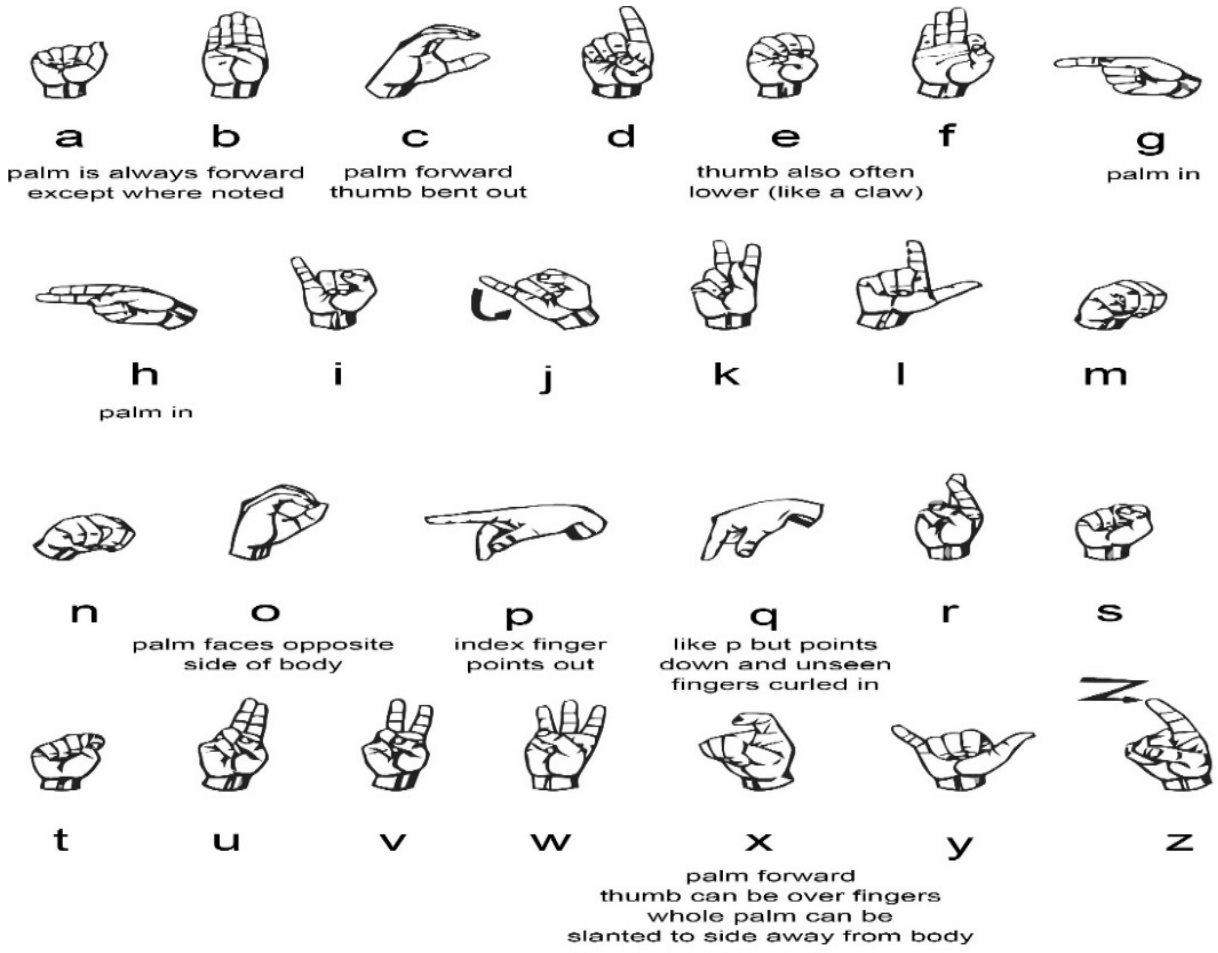


Figure 1. Characters A-Z of ASL (Source: www.nidcd.nih.gov) [7]

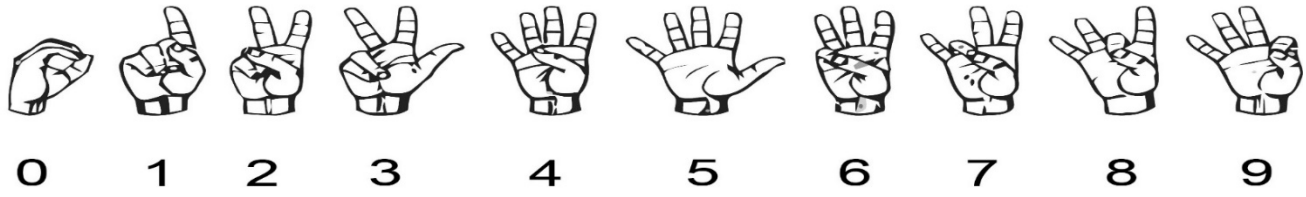


Figure 2. Letters 0-9 of ASL (Source: www.nidcd.nih.gov) [7]

Serving as a basic structure block for comprehension of sentences, the gesture identifying task is also arduous:

- The meaning of gestures mainly relies on the combination of body motions, manual movements and head poses, and subtle differences in different substances. As appeared in Figure 3, the signs for "dance" and "read" only vary in the orientations of hands [8].
- The dictionary of signs in daily use is enormous and usually in the magnitude of thousands. In contrast, related tasks such as gesture recognition and action recognition only contain a few hundred categories. This dramatically challenges the scalability of recognition methods [8].
- A word in gesture-based communication may have multiple counterparts in natural languages. For example, the sign shown in Figure 4 can be interpreted as "wish" or "hungry" depending on the context.

Figure 3. ASL signs "read" (top) and "dance" (bottom) differ only in the orientation of the hands [8]

Figure 4. The "wish" (top) and the "hungry" (bottom) correlated to the same sign [8]

In the recent years with respect to sign language identification, various research projects have been conducted. Predominantly, the techniques for identifying sign language can be divided into two general classes: sensor- and vision-based approaches. The sensor-based system utilizes various sensors that mostly attach to the hands to capture hand gestures. The first one is Flex sensors, and the other one is Inertial Measurement Units (IMUs). These are the two most common sensors for tracking hand gestures [9]. To distinguish the bending movement of fingers and the direction of the hand theses sensors are used. The flex sensor connected to the finger measures the bending angle by converting the gesture into digital output. The higher the curve's degree in the flex sensor, the higher the digital work value. However, the flex sensor can't measure the direction of the finger and hand. Thus, the IMU (Inertial Measurement Unit) sensor was embraced to solve this issue. Thus, the vision-based methodology is an alternative to communication via sign language recognition. Motion algorithms analyze the gestures recorded by video camera or visual sensor [10].

1.3 Misconception About Sign Languages

Multiple sign languages are used in the world, and they are not the same in any way. People generally believe that the languages are for the deaf community, and they miss the fact that few languages originated from villages. It indicates all the sign languages are different; they might be related to specific conditions but not the same, for example, it has its own punctuation and its own way of articulation. Sign languages have their unique linguistic structures. Communication through sign language has the trouble of verbal correspondence, yet it is self-deciding from the letter sets. The best model is British Sign Language and American Sign Language, which are various dialects. The real factors show that in need of a hearing aid people from the United States and Britain cannot pass on their specific signal-based communications [11]. Sign languages are challenging to use compared to spoken languages, as at times, the gesture is not as perfect as it should be. There is no worldwide gesture-based communication (sign language), and to state that the communication through sign language utilized in the USA and Canada is the same as the sign language used in India, is again a myth. Sign languages are not the simple visualization of spoken languages; instead, they are complicated and convoluted.

Another general misconception about communication through sign language is that it is internationally comprehensible, which isn't valid. As clarified above, the Sign language utilized by hard of hearing people in America is not equivalent to what is used in Britain. The distinctive gesture-based communications may be comparative in certain letter sets, but a hard of hearing individual from one country can't transmit fluently with an individual from another country [11].

Since communication via gesture is a language with distinct characteristics, finger spelling or the dictionary of characters (letter set) can't be used as a choice to sign. It is used in signifying the words with a non-existing sign or when the sign is not known. Additionally, a hard

of hearing people would require hours to convey a few minutes of messages through fingerspelling.

1.4 Objective of this Research

The most crucial factor of this research is the use of Artificial Intelligence to help the deaf community. There has been a tremendous improvement to classify ASL and provide textual information of the static gestures. Researchers have claimed almost 99% accuracy and have successfully achieved remarkable results in static gesture classification [1]. This research is based on dynamic gesture recognition and has not been addressed in depth so far as there are limited datasets available on dynamic gesture recognition and the datasets and the ones available are not good enough for testing because of its poor video quality and not enough videos per class. The objective is to take the video sequences of American Sign Language and classify the words using deep learning cascaded models.

1.5 Overview of methodology

Figure 5, shows the block diagram of the process to classify the ASL videos. To start with the dataset, we have acquired a WLASL dataset which contain more than 2000 classes [8]. After collecting the dataset, 100 classes were extracted after sorting in descending order according to the number of videos per class. From the videos of 100 classes, hands were extracted using YOLOv3 models and config files. The videos were converted into frames using video data generator which is an open-source library compatible with Keras image data generator. To classify the videos, CNN based models and a RNN was used to extract spatial temporal features. CNN models such as VGG16, VGG19, ResNet101, Inception v3 and

Inception 3d with LSTM were used to make the prediction. The pretrained CNN architecture was used for this video classification task and the hyperparameters were fine-tuned for the model.

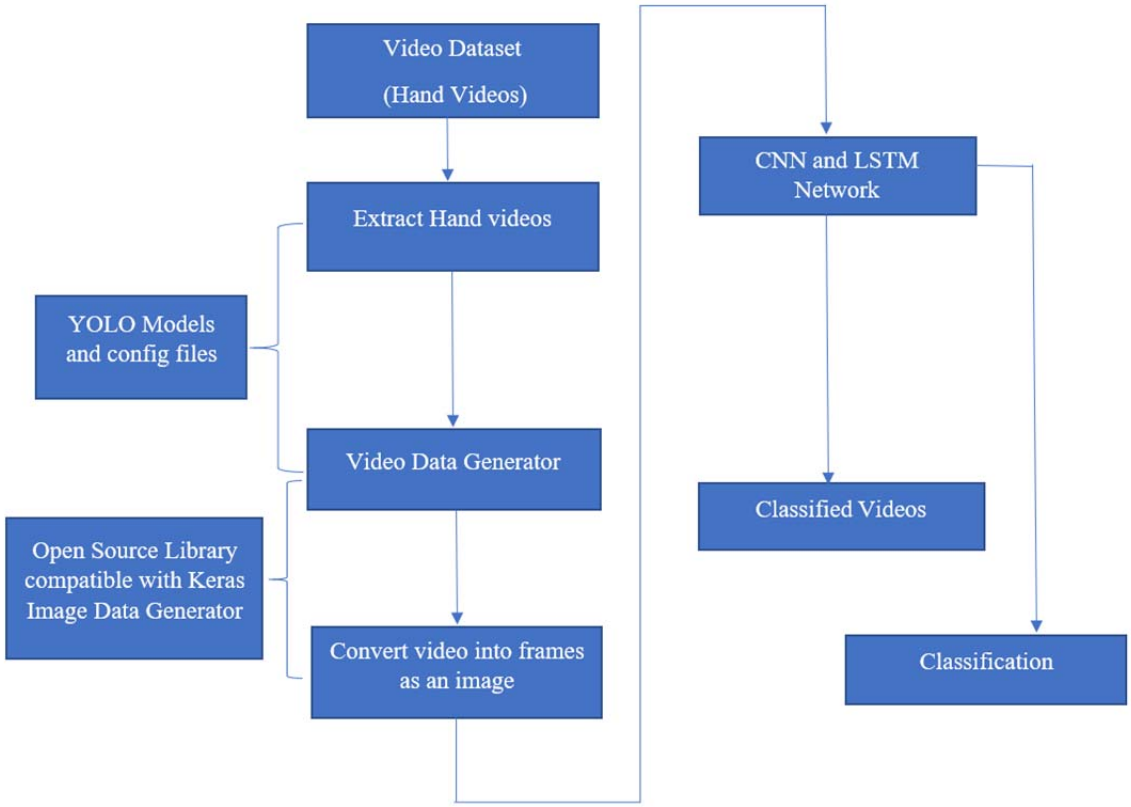


Figure 5. Block Diagram of the process to classify the ASL videos

1.6 Contributions

Contributions in this thesis are listed as follows.

- A cascaded CNN (Convolutional Neural Network) and RNN (Recurrent Neural Network)-LSTM (Long Short-Term Memory) network were improvised to classify the videos.

- Top 100 classes were used to extract hand videos. The yolo model was pre-trained on the hands' dataset.
- Hyperparameters were tuned to obtain the best model for predicting the signs in the ASL.
- The ASL signs from the videos were classified for top-k classes and the performance was measured for accuracy of prediction.

1.7 Outline of the thesis

Chapter 2 describes the literature review of related work on different techniques using machine learning and deep learning approaches.

Chapter 3 discusses the data set and data processing. WLASL dataset has been used for classification of ASL and the chapter describes how this dataset was acquired and prepared for our work.

Chapter 4 discusses the deep learning methods and transfer learning, which are based on CNN architecture such as VGG16, VGG19, Inception, and ResNet, etc.

Chapter 5 discusses the results and analysis.

Chapter 6 presents the conclusions and future work.

Chapter 2

Literature Survey

People with hearing and discourse inabilities use communication through sign language. People use gestures-based communication to convey their thoughts and feelings as a form of gestural (non-verbal) communication. But non-signers discover it hard to decipher, so during clinical and lawful appointments, training and school meetings, public announcements by the government, qualified sign language translators are required. There has been a developing interest for deciphering services over the past five years. Other means have been added, such as remote human interpretation of video using high-speed internet access. Hence, they would give a simple-to-utilize communication through sign language interpretation service that can be used but has considerable constraints, such as internet connectivity and a suitable computer and daily costs. To address this, we look forward to a solution using artificial intelligence and machine learning to identify gestures in sign language.

This chapter deals with the literature review and survey of the research and past studies for the problem, which shows that there have been several ways to deal with address sign recognition in video using several different methods. Cohen et al. [12] used Hidden Markov Models (HMM) combined with Bayesian Network (BN) classifier and Gaussian Tree Augmented Naive Bayes (GTANB) classifier to recognize facial expressions from video sequences.

Neural Networks are often used to evaluate video segments that require removing visual information in terms of feature vectors. It faces problem for instance, hand detection, subject

fragmentation from the context and environment, lighting, variance, impediment, motion, and location by dividing the dataset into segments, extracting characteristics, and utilizing Euclidean distance and K-nearest neighbor to distinguish the signs [13].

In [13], Nandy et al. research describes how continuous recognition of Indian Sign Language can be achieved. First the frames are extracted from video data, the data is preprocessed, and features are extracted and optimized gradually. By converting the video to a series of RGB frames, preprocessing is completed. Each frame is the same size. Using the AHS curve, skin colour separation is used to remove skin areas. The collected photographs were translated into binary form. By measuring a gradient between the frames, keyframes were retrieved. Using an orientation histogram, characteristics were extracted from the keyframes. Euclidean distance, Manhattan distance, chessboard distance and Mahala Nobis distance were utilized. They concluded that K-nearest neighbor (KNN) gives the best performance with up to 100 % recognition.

A study recognized difficulties with sign language recognition in a paper by Jie et al. [14], such as comprehension problems where the signals are split down to individual terms and problems with constant sign language recognition. They tried to solve the issue through isolation of individual signals, which avoids an additional step of preprocessing and another additional post-processing step since they assumed that temporal segmentation is essential for sign language recognition propagates through subsequent steps without its errors. Integrated with individual words' intensive labelling, gesture-based communication recognition without temporal segmentation faces a significant challenge. This challenge was overcome by introducing a Hierarchical Attention Network with Latent Space (LS-HAN), which avoids temporal segmentation. The framework includes a two-stream CNN for the generation of video attribute

description, a Latent Space for semantic distance bridging and a space-based recognition, Hierarchical Attention Network. Videos are divided into 16-frame per clip and resized at 227x227. They concluded that LS-HAN gave the highest accuracy with 82.7%.

Some sign language processing techniques require the use of an exterior system such as a Leap Motion controller to understand motion and movements, such as Chong et al.'s work [10]. The analysis varies from several other works as it contains 26 letters and 10 digits of the full grammar of the American Sign Language. The purpose of the work is to research and identify complex gestures and to extract features. Support Vector Machine (SVM) and Deep Neural Networks (DNN) achieved an accuracy of 80.3% and 93.8% respectively.

Linqin et al. [15] used RGB-D data to identify human movements for contact between humans and computers. A unifying descriptor of features is produced by measuring the Euclidean distance between hand joints and shoulder characteristics. A Dynamic Time Warping (IDTW) algorithm uses weighted distance and narrow search paths to get the final recognition results. The test findings of this process show an average precision of 96.5 %. The concept is to improve the identification of real-time movements that could also be applied to sign language recognition.

Another solution to the issue is the work performed by Ronchetti et al. [16] on the Argentinian sign language: the use of the Argentinian Sign Language handshape database and the image processing method, the retrieval of descriptors and handshape classification using ProbSom. Support Vector Machines (SVM), Neural Networks and Random Trees were used for the classification. The approach's average precision was more than 90%.

Hardie et al. [17] used Myo armband, an external computer to gather data regarding the location of a person's hands and fingers over time. The study utilizes a dataset compiled by a group at the University of South Wales that provides criteria for 95 specific signals, such as hand positions, hand revolution, and finger bending. There is an incoming stream in each sign, and they determine which sign the stream falls into. SVM and logistic regression methods are used for classification. The lower quality of the data needs a more modern methodology, so various temporal classification methods were being studied. They split the data into 70% for training and use the remaining 30% as test instances. They used SVM, logistic regression, LSTM and SPM for classification and got the highest precision of 56.6% with baseline SVM (Linear kernel).

Gracia and Alarcon [18] applied transfer learning American sign language fingerspelling alphabet. They utilized GoogleNet that was prepared on the ILSVRC2012 dataset and the Surrey University and Massey University ASL datasets. Their data is made out of hands in 24 different directions, whereas the ILSVRC data is composed of 1000 uniquely different objects or classes. They resized 256x256 and took out random crops of 224x224 to coordinate the input size of the GoogLeNet model, then they normalized the input data and fed it in the network for classification. They obtained a-e Top-5 Val-accuracy as 100%.

Rim Barioul et al. [19] used four FMG commercial sensors to recognize American Sign Language. The sensors are associated as a FSR sensor's bracelet. The study compared the accuracy of recognition of nine ASL alphabets by raw FMG and Extreme Learning Machine (ELM). Five-fold cross-validation was used with ELM training and an accuracy of 89.65% was achieved, whereas raw FMG resulted in an accuracy of 69.69%.

Mathieu De Coster et al. [20] addressed sign language recognition using transformer networks. In their research they applied a blend of feature extraction utilizing OpenPose for

human keypoint estimation and Convolutional Neural Networks (CNN) was used for feature learning. They used three architectures based on transformers and one on LSTMs. The isolated signs in the Flemish Sign Language (FSL) corpus are identified using multi-head mechanism achieving 74.7% accuracy on a vocabulary of 100 classes. The limitation for them was a large dataset, and in conclusion it is suggested as future work.

Al Amin Hosain et al. [21] identified gestures for American Sign Language (ASL) from videos. They proposed a pattern Recursive Neural Network (RNN) gesture recognition model from 2D skeletal data trajectories estimated from videos. The model was extended by incorporating hand images from another hand shape recognition model. The final model is a fusion of two LSTM RNN models for skeletal and hand image data. The model was trained on the GMU-ASL51 dataset of 12 users and 51 ASL gestures where fusion LSTMs gave an average accuracy of 89%. The research concluded this 2D skeleton-based model is better than 3D models and sensor-based models.

Sign recognition deep learning CNN model was developed by Passi and Goswami [1] where they propose methods to identify the hand gestures, skin color and hand contours. The model achieved 99% accuracy for alphabets with different orientations, illumination conditions, and cluttered background. A high accuracy was achieved due to the use of high-quality pictures for the dataset, and the number of the training pictures.

Patrice Ferlet [22] discussed video data generators compatible with Keras image data generator. Basically, when we had video sequences, we had to convert that into frames that can be done using OpenCV, but we wanted to perform data augmentation as the number of videos per class was less. In this article, Patrice explained the different video data generators, sliding

generators, which can be used to convert video sequences to frames. He also showed examples using three classes of action database. After a careful reading of the article, we got the first step to dive into the deep neural network.

The literature survey shows that there have been various methodologies for detecting sign language's meaning within neural networks themselves. The input data to the neural networks play an important role in defining the network topology, such as getting RGB input and the depth field from a 3DCNN model. Thus, the observations of our model were compared to two very close solutions to sign language detection for the purpose of verification.

To remove spatial features, Lu et al. [23] used a general Convolutional Neural Network (CNN) network and used a Long Short-Term Memory (LSTM) to remove sequence characteristics. Vivek et al. [24] used CNN models for their architecture with RGB inputs. With a custom dataset of their own making, the developers of [24] focused on American Sign Language. The architecture in [23] was ResNet, a pre-trained CNN model along with their custom LSTM. Although design in [24] used a CNN for stationary hand movements, we had to take the liberty to expand their base model with the LSTM from our network.

This work is closely related to Word-level Deep Sign Language Recognition WLASL [8] by Dongxu Li et al. They proposed a new dataset for American Sign Language Recognition, which consists of 2000 classes in RGB format performed by 100 signers. The dataset is scraped from publicly available videos from YouTube and other online sources. They trained Pose-GRU, Pose-TGCN, VGG-GRU and I3D on the new dataset, and they concluded I3D performs better among all with the highest Top-10 accuracy of 89.92% in 100 classes. They pre-processed the video dataset by taking 50 consecutive frames from each video and cropping to 224x224 size.

The dataset was split into 4:1:1 training, validation and test set respectively and after training, the top-k accuracy metric was calculated to compare models.

Each video was then preprocessed before being utilized as an input to the model. Preprocessing the model involved initially avoiding background noise from the videos. The dataset was then expanded to increase the size of training and test for better performance in training. To do preprocessing we had multiple testing at our end on ten classes but those were all image processing based like thresholding, canny, HSV filter and few more. Since the results were not satisfactory, we decided to take out hands out of videos.

To overcome the above issue, in this research, a yolo model was used which was pretrained on hand for 100 classes of WLASL hands. After this the model consists of sliding video generator to convert videos in frames which is followed by VGG-16 and LSTM layers for classification/sequence prediction as our baseline model. So, the calculated top-k accuracy was used while testing. The data was divided into 85:15 and 15% videos were taken out in a separate folder for testing. The model was trained on 85% data with 70% data utilized for training and 15% used for validation. Once the model was trained then the weights were saved for testing. The test set (15%) was converted to NumPy-array and was used for prediction and then top-k approach was followed to obtain the accuracy.

Chapter 3

Datasets

This chapter will describe the dynamic American Sign Language dataset and how this dataset was acquired and prepared for our work.

3.1 Description of datasets

We have acquired a dataset for this work, namely World-Level American Sign Language WLASL [8]. WLASL dataset contain more than 2000 classes. It is the ASL-based dataset. We have taken out a sample using a sliding generator, shown in the video samples' frames in Figure 6.

Figure 6. Video Frames of WLASL dataset for two classes

Figure 6 is the frames of a video sample of the toilet and think classes. It is taken out of the WLASL dataset, which is in RGB format.

3.2 Acquiring and Preparation of the datasets

We use a recorded Word-Level American Sign Language (WLASL) dataset [8].

[1] The dataset consists of 2000 different signs captured multiple times with different context and video conditions. The videos in the dataset were recorded at a common frame rate.

[2] Spatial features are extracted from the video stream to recognize the sign language through pre-trained models of CNN (Convolutional Neural Network) namely VGG16, VGG19, ResNet, and Inception. Further, temporal characteristics are extracted from video sequences using a LSTM (Long Short-Term Memory) RNN model.

WLASL Dataset was downloaded using the procedure provided by the author Dongxu.li [8]. The data downloaded is pre-processed for converting all video files to mp4, and all the video files are

```
try:
    os.makedirs(dest + lines['gloss'])
except OSError:
    print("Directory creation failed")
```

present in a single folder without labels. The library contains a JSON file with all the information about each video file. The class to which each video file belongs is provided under the tag 'gloss', and the title of the video file is provided under the label 'video_id'. The data is scraped using JSON and os libraries of python. The os library is used for reading and creation of folders using loops. A single folder for each class is created and filled with videos that belong to a ton of that class. This segregation is done so that the Keras-video-generators library can read the video data frames and classes. The initial arguments are used to import libraries and set the source and destination paths. The for loop is used for reading all the lines of the JSON file. The code snippet below is used to create a class directory and provide an error message if the directory is already present.

The code snippet below is used to transfer files from the source directory to the new class directories.

```
Try:
   os.rename(path + data['video_id'] + '.mp4', dest + lines['gloss'] + "/" + data['video_id'] +
'.mp4')
   print(dest + lines['gloss'] + "/" + data['video_id'] + '.mp4')
except:
   print("File transfer failed")
```

3.3 Extraction of hands from WLASL dataset

The hands were extracted from the frames of the gesture videos by using the YOLOv3 model [8]. The program shown below in the snippet consists of three functions used for processing video data. The function process_videos is used to list and read the video files in the WLASL dataset folder. The class folders inside the dataset are iterated over, and the video files inside each folder are read. The videos are converted into frames using the function video_to_frames. This function reads all the frames of the video files applies the YOLO hands DNN (Darknet Neural Networks) to these frames. The YOLO hands DNN provides coordinates for bounding boxes when hands are recognized inside a frame. The function cv2.dnn.blobFromImage used for creates 4-dimensional blob from image. It resizes and crop image from center, swap Blue and Red channels. The darknet neural network (dnn.) is a library and it allows to load pre-trained network via the TensorFlow framework and then use them to classify the input image. The net.setInput(blob) set the input to the network. Then it runs forward pass to compute outputs of layers listed in blob using function net.forward(ln). To identify objects, we used net.setInput(blob) and layerOutputs = net.forward(ln) instructions to calculate the network response. In the second stage, define the bounding box to detect the object. When Confidence value is greater than 0.5 it will return the boxes center (x, y) – coordinates of the box followed by the boxes' width and height.

```
blob = cv2.dnn.blobFromImage(frame, 1 / 255.0, (416, 416), swapRB=True, crop=False)
net.setInput(blob)
layerOutputs = net.forward(ln)
# initialize our lists of detected bounding boxes, confidences,
# and class IDs, respectively
boxes = []
confidences = []
classIDs = []
```

```
# loop over each of the layer outputs
for output in layerOutputs:
    # loop over each of the detections
    for detection in output:
        # extract the class ID and confidence (i.e., probability)
        # of the current object detection
        scores = detection[5:]
        classID = np.argmax(scores)
        confidence = scores[classID]
        # filter out weak predictions by ensuring the detected
        # probability is greater than the minimum probability
        if confidence > 0.5:
            # scale the bounding box coordinates back relative to
            # the size of the image, keeping in mind that YOLO
            # actually returns the center (x, y)-coordinates of
            # the bounding box followed by the boxes' width and
            # height
            box = detection[0:4] * np.array([W, H, W, H])
            (centerX, centerY, width, height) = box.astype("int")
            # use the center (x, y)-coordinates to derive the top
            # and left corner of the bounding box
            x = int(centerX - (width / 2))
            y = int(centerY - (height / 2))
            # update our list of bounding box coordinates,
            # confidences, and class IDs
            boxes.append([x, y, int(width), int(height)])
            confidences.append(float(confidence))
            classIDs.append(classID)
```

These bounding boxes are overlaid on the original video frames, which are stacked in an array of images. The below code snippet used to save the bounding box coordinates and frames.

```
if len(idxs) > 0:
    # loop over the indexes we are keeping
    for i in idxs.flatten():
        # extract the bounding box coordinates
        (x, y) = (boxes[i][0], boxes[i][1])
        (w, h) = (boxes[i][2], boxes[i][3])
        result[y:y+h, x:x+w] = frame[y:y+h, x:x+w]
        frames.append(result)
```

The function convert_frames_to_videos is used to convert an image array into a video file of the format mp4. The video file to be formed has the frame rate set to 25, and the size is taken from

the size of the frames inside the array. All the processed video files are written to class folders inside the destination path, like the source folder.

The following is the list of the top 100 classes that were used for this research. Some signs in ASL have gestures involving body parts and facial expressions along with hands. Among the 100 classes given below, some signs include facial expressions with hands as well which are highlighted below:

‘blowing’, ‘brown’, ‘candy’, ‘cool’, ‘corn’, ‘dark’, ‘deaf’, ‘drink’, ‘eat’, ‘forget’, ‘hat’, ‘hearing’, ‘hot’, ‘kiss’, ‘man’, ‘mother’, ‘orange’, ‘pink’, ‘pizza’, ‘secretary’, ‘son’, ‘tell’, ‘thanksgiving’, ‘thin’, ‘thursday’, ‘water’, ‘what’, ‘who’, ‘woman’, ‘yes’

'accident', 'africa', 'all', 'apple', 'backpack', 'basketball', 'bed', 'before', 'bird', 'black', 'blue', 'book', 'but', 'can', 'chair', 'change', 'cheat', 'city', 'clothes', 'color', 'computer', 'cook', 'cousin', 'cow', 'dance', 'decide', 'dog', 'enjoy', 'family', 'fine', 'finish', 'fish', 'full', 'give', 'go', 'graduate', 'help', 'jacket', 'language', 'last', 'later', 'letter', 'like', 'many', 'meet', 'no', 'now', 'paint', 'paper', 'play', 'pull', 'purple', 'right', 'school', 'shirt', 'short', 'study', 'table', 'tall', 'time', 'visit', 'wait', 'walk', 'want', 'white', 'wife', 'work', 'wrong', 'year', 'yellow’

Chapter 4

Deep Learning Methods and Transfer Learning

4.1 Convolutional Neural Networks

The ability of Convolutional Neural Networks (CNNs) to learn and distinguish between image-based patterns makes them highly specific for use in the classification of images and videos, making Convolutional Neural Networks help in the development of the state-of-the-art models for image/video classification. This section will discuss how Convolutional Neural Networks work and understand Convolutional Neural Networks [25]. The main objective of this chapter is to study the different functionalities associated with Convolutional Neural Networks such as Cost function, Gradient Descent Optimization, Normalization and Regularization methods and the other pre-trained models which are based on CNN architecture such as VGG16 [26], VGG19 [26], Google Net or Inception [27] etc. which we use to classify the WLASL dataset [8].

4.2 Training of CNN

The training process of a Convolutional Neural Network can be best understood as the identification of kernels in the convolution layers and then finding the best weights in fully connected layers to minimize the loss or cost functions so that the model can give accurate predictions on the test dataset [25][28]. In CNN, the most interesting parameters are the filters in the different layers. Then a regular gradient descent method is used. As the epochs are run during

training, the parameters gradually change until they reach their final values, which serve as useful features successfully used to achieve the desired task. The general architecture of the CNN is shown in Figure 7 below.

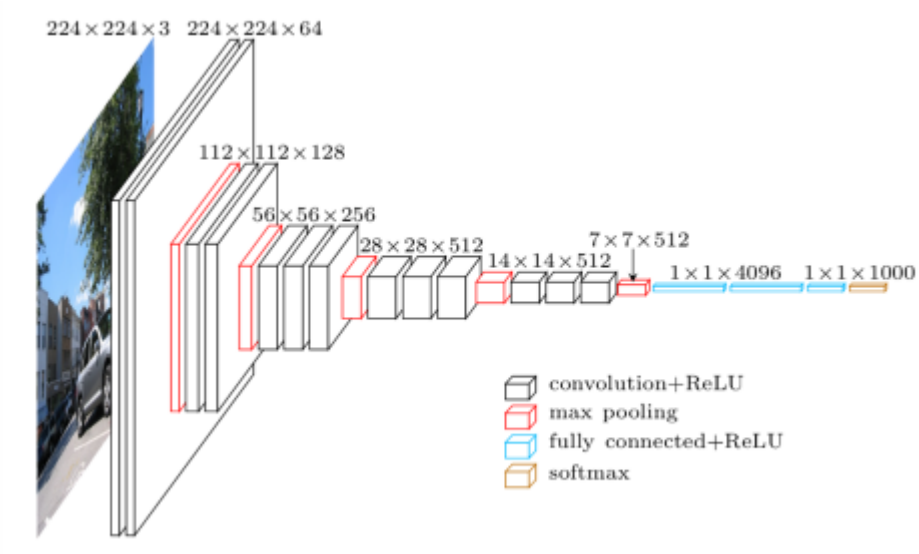


Figure 7. Architecture of CNN [29]

The CNN has four types of layers in it, the convolutional layer, the pooling layer, the ReLU correction layer and the fully connected layer, which are described below:

The convolutional layer is the first and most important layer in convolutional neural networks. With kernels of the convolutional layers and the parameters of the model, an error is calculated at the output layer using the categorical cross-entropy as a loss function in the forward mechanism, which is called forward propagation on the training set of data. Next, these weights are updated using the loss function by the gradient descent optimization algorithm in a backward mechanism, which is called backpropagation. The weights and bias are adjusted using gradient descent and backpropagation to calculate the adjustments to weights that minimize the loss function. While performing backpropagation in the training of the neural networks, the cost

function, which is categorical cross-entropy as well as the optimization method which is gradient descent work together.

The pooling layer is generally positioned between two convolution layers. The pooling operation is applied on each feature map. The pooling activity comprises in decreasing the size of the frames(images) while saving their important characteristics. To perform this, the image was cut into the regular cells, and then the greatest value was kept inside every cell. The well-known decisions are 2x2 adjoining cells that do not overlap. After applying pooling layer, we get output as similar number of feature maps as an input, but these are smaller. It lessens the number of parameters to learn and the amount of computation performed in the network, which improves the efficiency of the network.

The Rectified Linear Unites (ReLU) known as the real non-linear function is defined by $\mathrm{ReLU}(x) = \max(0, x)$. It is the most commonly used activation function in the CNN models. The aim of applying ReLU layer with CNN is to increase the non-linearity in frames as an image because the image is naturally non-linear. When we look at any image, it contains a lot of non-linear features such as pixels, the borders, colours, etc.

The Fully connected layer is a fundamental part of Convolutional Neural Networks (CNNs), which has been demonstrated in perceiving and analyzing frames as an input to the network. The CNN cycle starts with convolutional and pooling layers, then the features are extracted from the images and analyzed independently. The weighted features are then passed into a fully connected neural network structure to obtain the final classification decision. The weight values are learnt through the backpropagation of the gradients.

The training of the network by the backpropagation algorithm can be classified into the following three stages:

- The feedforward mechanism of the input to output layer with the initialization of weights, biases are given the specific parameters and architecture of the model.
- The calculation of the categorical cross-entropy or error using the backpropagation algorithm.
- Using Gradient Descent optimization algorithm to adjust the weights and retrain the process until the loss is minimized.

Firstly, before the application of the backpropagation algorithm, weights are initialized, which are randomly chosen from the input layer to hidden layers, generally having the values between –0.5 & 0.5. The data passed into the input layer is first normalized to avoid overcalculations and unnecessary overfitting in training [28].

In the backpropagation algorithm, two adjustments are performed for weights of the layers and for the biases; weights are updated based on the measurements of how each pixel affects the weight kernel and hence the loss function. In Convolutional Neural Networks, weights are convolution kernels and values of kernels are updated in backpropagation on convolutions. When input features are passed to the Convolutional Neural Network, the loss or error is calculated by comparing scores with truth predictions in the last layer. This calculated loss is backpropagated to the next layer to update the weights, and the process continues.

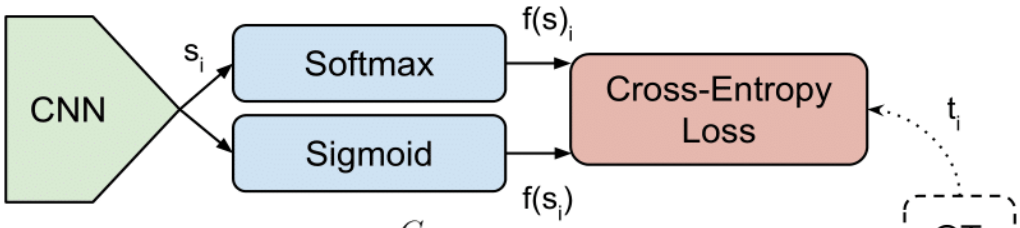
It can be concluded that in a forward phase, the input image converted into the feature maps is passed completely through the Convolutional Neural Network. Each layer will convolute inputs, intermediate values such as kernel weights, strides etc., calculating the loss or error. In contrast, during the backward propagation phase, the gradients are backpropagated, and each layer receives a gradient of the loss from the output and returns the gradient of the loss from the inputs.

4.3 Cost function

The forward pass during the training is responsible for the computation of loss, which is achieved through a Cost Function. In contrast, during a backward pass, the weights are updated using the gradient computation. So, the cost function plays a very important role in the training process of the Convolutional Networks or any other Artificial Neural Networks (ANN) algorithm. The training of an ANN, which involves reducing error, is extremely slow and time costly. The cost function helps to solve this problem by a cross-entropy error function during the backpropagation. To make the training process faster, instead of minimizing squares of the differences between the true labels and predicting outcomes, a categorical cross-entropy error function is minimized or optimized.

A cost function, also known as the Loss function, calculates the differences between the predictions and the ground or truth labels to give a numerical value of the cost using a forward propagation algorithm's forward mechanism. For each propagation, the loss is calculated, the forward-propagation algorithm achieves it by measuring the partial derivatives with regards to the parameters of the Neural Network. The cost function for the Convolutional Neural Network is not specifically different. The way CNNs are used is not as unique as it works on the same processes and principles that other Neural Network architectures work. It must be understood that the cost function is defined by the task, such as regression or classification, not specifically by the architecture of the model. For example, in our case, for classification, the cross-entropy loss function is utilized to calculate the distance between the output of the activation function

and the ground truth label [30]. The cost function, which is categorical cross-entropy, is defined by the equation in Figure 8.



$$CE = -\sum_{i}^{C} t_i log(f(s)_i)$$

$$CE = -\sum_{i=1}^{C'=2} t_i log(f(s_i)) = -t_1 log(f(s_1)) - (1 - t_1) log(1 - f(s_1))$$

Figure 8. Cross-Entropy Loss computation Process and Equations [31]

In Figure 8, t_i and s_i are the ground truth and Convolutional Neural Network predictions for each class i in C (C=2 for binary classification and C>2 for multi-classification). An activation function (Sigmoid in case of binary Classification, SoftMax in case of multi- classification) is applied to the predicted scores before the cross-entropy loss computation. Hence, $f(s_i)$ here refers to the activation function.

4.4 Gradient Descent

As discussed above, the cost function is used to calculate the errors in the forward pass, the aim being to minimize the computed errors. An algorithm is used to update the weights and retrain the model during backpropagation using updated weights, for this operation Gradient Descent Algorithm is used.

Over the years, researchers have been working on improving new optimization algorithms. Deep Neural Networks are significantly an optimization problem. The main

objective is to find a global optimum minimum using a computationally time-efficient convergence process that can only be accomplished using gradient descent algorithms optimization [30]. The main objective of the neural network training and optimization using Gradient Descent is to discover the best values of the parameters providing a best-fitted model with the least error and correct predictions for input features to the target features. A general optimization process can be mathematically denoted as the equation shown below [30][32]:

$$f: R^n \rightarrow R, Find\ \hat{x} = \text{argmin} f(x), x \epsilon R$$

where f is the cost function.

Gradient descent is the widely used optimization algorithm in neural networks that can continuously adjust or update the learning parameters of the model training, which includes kernels and weights and hence minimizing the computed loss or error term. The gradient descent optimization is depicted in Figure 9.

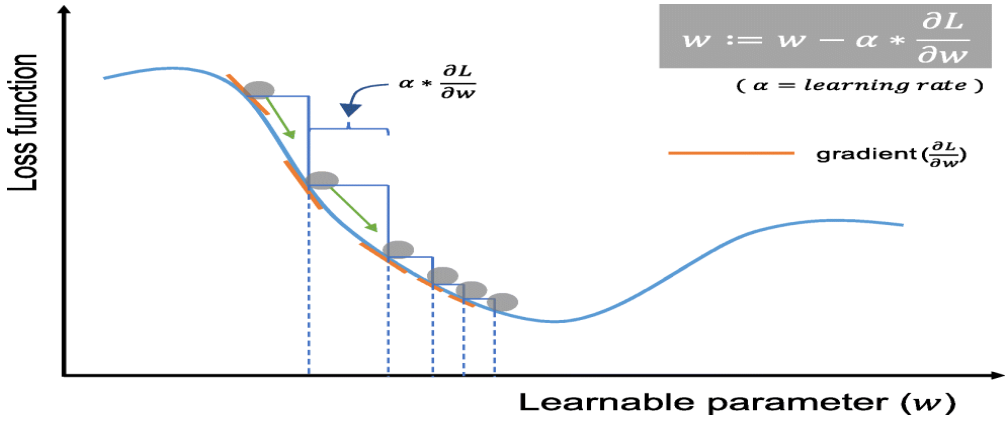


Figure 9. Gradient Descent Optimization Algorithm [25]

Specifically, the gradient descent is a partial derivative of the cost with respect to each parameter which must be learned during the training, and a single update of a learning parameter can be mathematically written as:

$$W := w - \alpha * \partial L/\partial w$$

where w stands for a parameter to be optimized, α stands for a learning rate, and L stands for a loss function.

Most of the time, because of memory constraints, the gradients of the cost function are computed by a small subset of the training dataset rather than the full dataset, which is called a mini-batch which is called Mini-batch gradient descent method, also known as stochastic gradient descent or SGD where mini-batch size is also a hyperparameter which can also be optimized for better results. Many improvements have been proposed on stochastic gradient descent optimization algorithms, which are widely used. Some of them are stochastic gradient descent with momentum, Root Mean Square Propagation or RMSprop, and Adaptive Moment Learning or Adam [32]. Adam is discussed in detail in the next section.

4.5 Adam Optimization

Currently, first order-based optimization algorithms play an important role in deep learning on account of their efficiency and adequacy in dealing with large-scale optimization problems. Adam or Adaptive Moment-based optimization learning algorithm is a first-order gradient-based optimization algorithm with low memory requirements. It is quite computationally efficient for problems requiring a large number of parameters.

Stochastic Gradient Descent (SGD) iteratively updates the gradient and the parameters to move down the slope until it converges. SGD is commonly used to train deep neural networks. Different techniques have been used to automatically modify the learning rate in SGD by taking the square root of the average of the square of the previous gradients [32][33][34]. Adagrad

(Adaptive Gradient Algorithm) [36] was proposed for sparse data. Since this method uses all the past gradients, the learning rate can decrease rapidly which is a major concern as it may result in jumping past the convergence point [36]. The exponential moving average of past squared gradients has been instead used in algorithms Adadelta [28][30], RMSprop [37], and Adam [38]. Adam (Adaptive Moment Optimization algorithm) is a heuristic of both Momentum-based and Root Mean Square Propagation (RMSProp). Adam is an adaptive learning rate method which is a blend of Stochastic Gradient Descent (SGD) with momentum and RMSprop. It calculates the squared gradients for scaling and alters the learning rate like RMSprop, also utilizing the moving average of the momentum of the gradient in a similar way as used by the SGD. The update equations of the Adaptive Moment Optimization algorithm (Adam) are given as the following.

$$v_t = \beta_1 * v_{t-1} - (1 - \beta_1) * g_t$$

$$s_t = \beta_2 * s_{t-1} - (1 - \beta_2) * g_t * g_t$$

$$\Delta\omega_t = -\eta \frac{v_t}{\sqrt{s_t + \epsilon}} * g_t$$

$$\omega_{t+1} = \omega_t + \Delta w_t$$

$\eta = Initial\ Learning\ rate$
$g_t = Gradient\ at\ time\ t\ along\ \omega_j$
$v_t = \ Exploential\ Average\ of\ gradients\ along\ \omega_j$
$s_t = Exploential\ Average\ of\ Squares\ of\ gradients\ along\ \omega_j$
$\beta_1, \beta_2 = Hyper\ Parameters$

Assuming that, there are two hyperparameters β_1 and β_2 with an initial learning rate of η, first the exponential moving average V_t of the gradient is calculated and next, the S_t of the learning parameter is calculated, which is demonstrated in the first equation and second equation

respectively. Next, to decide the learning step, the learning rate is multiplied with the average of the gradient for example, as in the case of the momentum and dividing it with the root mean square of the sum of the exponential average of the squares of gradients as shown in the equation. At each iteration, this update is added continuously. The hyperparameters β_1 and β_2 are generally initiated in the range 0.9 and 0.99, respectively, and the value of $\in$ is considered to be $1e^{-10}$ initially.

4.6 Regularization

Convolutional Neural Networks or any other complex networks consisting of an enormous number of learning parameters are quite complex. Hence, the chances of overfitting are a difficult issue in such networks. Convolutional Neural Networks are also time-costly, which makes overfitting a serious problem as to accomplish a better performing model in the desired timeframe. So, the overfitting in the training of a neural network can occur very easily and is dependent on various factors. Regularization is a method with which we can make some modifications in the training or learning process so that the model can train itself in a more generalized way, which also improves the model's performance on the unseen data.

As in the machine learning algorithms, the regularization penalizes the coefficients to avoid overfitting. Similarly, in deep learning, the regularization penalizes the weights of layers at each node. There are various methods of regularization in Convolutional Neural Networks.

1. Dropout
2. Data Augmentation
3. Batch Normalization

4.6.1 Dropout

Dropout is a regularization technique and is widely used in complex networks such as Convolutional Neural Networks as it produces very good results [39]. In dropout regularization technique, at every iteration, the addition of dropout layers reduces the number of nodes by removing nodes and its connections by randomly selecting some nodes. The random selection of nodes at each iteration results in different output values.

While comparing the process with the machine learning algorithms, it can also be compared with an ensemble technique, which is highly significant and widely used in machine learning. It has been proven that ensemble models perform far better than a single model as they are capable of capturing more randomness and can learn high dimensional datasets. In a similar fashion, a Convolutional Neural Network or any other network dropout also performs better. The activation values are zeroed for the randomly selected nodes in the dropout layers. Robust features are learnt through the dropout regularization in comparison to predicting the labels with a small subset of nodes.

4.6.2 Data Augmentation

Data Augmentation is another simpler yet very powerful way to reduce overfitting in complex neural networks. Whereas in machine learning it is not possible to increase the sample size of the training data, data augmentation can be used to increase the sample size of training data in Convolutional Neural Networks. In the case of images, the number of samples of the training data can be increased by rotating the image, flipping, shearing, shifting, and scaling .

This regularization technique is known as data augmentation and it can assist in the improvement of the accuracy of the CNNs to a great extent.

There are several ways of augmentations of data mainly in the geometric transformation, which are based on basic image manipulations. Geometric transformation data augmentation techniques are widely used as they can be easily implemented. The only disadvantages that geometric transformation techniques possess are computational costs, an increase in the training period. In geometric transformation, various image processing functions are implemented, which are described below.

Flipping- Flipping is the easiest and useful geometric data augmentation technique that involves flipping the image in the horizontal and vertical axis. Flipping on the horizontal axis is more common than flipping on the vertical axis.

Colour Space- Colour Space augmentation technique involves isolating a single colour from a three-channel colour space (R, G and B). Colour space augmentation is also a quite effective and useful technique.

Cropping-When the images have mixed heights and width dimensions, cropping can be performed to extract a central patch of each image as a practical processing step for image data.

Rotation- Rotation based data augmentations method can be performed by rotating the image clockwise or anti-clockwise between 1° and 359°.

Translation- To avoid the positional bias of the images having a different position of the object, images can also be shifted to the left, right, up, or downside just like rotation augmentation.

Noise injection- Another augmentation technique is Noise injection, which involves the injection of a matrix of random data points usually taken from a distribution such as Gaussian distribution.

4.6.3 Batch Normalization

Batch normalization is another useful regularization method that helps in increasing the overall performance of complex neural networks such as CNNs and avoids the chances of overfitting while training. Batch normalization involves the normalization of the dataset within the activations of the different layers in a neural network. The batch mean is subtracted from each activation and divided by the standard deviation. So, the idea is basically to normalize the inputs of each layer of the neural network in a way so that it can have a zero mean output activation and a standard deviation of a single unit. This is quite similar to the standardization of the inputs. This batch normalization technique is quite a standard method for pre-processing values of the input data [40][41].

To be specifically answering the question, why do we use batch normalization in our convolution neural networks or other complex architectures are supported by the reason that Batch normalization reduces the amount by which the hidden unit values shift around, which is also known as covariance shift. The normalization is performed in the input layer by adjusting and scaling the activations.

For instance, when data features that are varying from a range of 0 to 1 and some of them are ranging from 100 to 1000, it is highly recommended to normalize the features to make the learning process of the network faster. We perform the scaling in the input layer, so why should we not do the same normalization in the hidden layers in which the values are varying

continuously. Batch normalization helps in making the training process ten times faster than without performing batch normalization. Another advantage of Batch Normalization is that it allows each layer of a complex neural network such as CNNs to learn the features by itself and independently from other layers. Figure 10 shows the batch normalization process in a neural network.

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Figure 10. Process of batch Normalization [40]

During the training, first batch mean and batch variance is calculated as shown in the first two equations in Figure 10 represented by μ_b and σ_b, then in the next step, the layer inputs are normalized using the mean and variance calculated above. Finally, at the last step, scaling and shifting are performed to obtain the output of the layer, as shown in the last equation. It must be noted that the γ and β are learned during the training process only with the parameters of the Convolutional Neural Networks or any other network [40].

4.7 Architectural innovations and applications of CNN

CNN 's study is also underway and has great scope for progress. The major changes in the success of CNN were mainly encountered from 2015-2019. Various experimental studies have shown that state-of-the-art deep architectures such as VGG, ResNet, ResNext, etc., have also shown strong outcomes for difficult problems of identification and localization, such as semantic and instance-based segmentation of objects, scene parsing, placement of scenes, etc. CNNs can be broadly classified into seven types, depending on the form of architectural modifications, specifically: spatial exploitation, depth, multi-path, duration, exploitation of feature maps, channel boosting, and CNNs dependent on attention.

4.7.1 VGG

A comprehensible and efficient design theory for CNN architectures was suggested by Simonyan [19]. Their architecture was modular in the pattern of layers, dubbed VGG. VGG replaced the 11x11 and 5x5 filters with a 3x3 filters layer stack and experimentally demonstrated the effect of large filters (5x5 and 7x7) could be caused by the parallel placement of small filters (3x3). By minimizing the number of parameters, the use of small filters gives an added advantage of low computation time. These results set a new pattern in analysis for CNN to work with a narrower filter. By putting 1x1 convolutions between the convolutional layers, VGG governs the difficulty of a network, which additionally learns a linear combination of the resulting feature maps [19].

4.7.2 ResNet

He and Zhang [20] developed the ResNet models, which are focused on deep architectures that have shown strong convergence behaviours and convincing precision. Focused

on this, they took first place in 2015 in the classification challenge for ILSVRC and Common Objects in Sense (COCO). Several stacked residual units were designed by ResNet and produced with several different layer numbers: 18, 34, 50, 101, 152, and 1202. However, it is possible to vary the number of operations based on various architectures. For all the models, convolutional and pooling layers are composed of residual units. ResNet is VGG net-like, but ResNet is about eight times deeper than VGG. Good compensation for depth and output is the ResNet 18, and this network is made out of five coevolutionary layers, one average pooling, and a SoftMax fully connected layer. At the end of the network, ResNet 50 comprises 49 convolutional layers and a fully connected layer. Finally, ResNet 101 was selected to build this study to save computational costs and training time [20].

4.7.3 Inception-V3, V4 and Inception-ResNet

Improved versions of Inception-V1 and V2 are inception-V3, V4 and Inception-ResNet. Without impacting the generalization, the idea of Inception-V3 was to reduce the computational expense of deep networks. Szegedy et al. replaced large scale (5x5 and 7x7) philtres with minimal and asymmetric (1x7 and 1x5) philtres for this purpose and used 1x1 convolution as a bottleneck before large philtres [42]. Concurrent positioning of a large-size philtre of 1x1 convolution makes the standard convolution process more of a cross-channel correlation. The capacity of 1x1 philtres in the Neural Network architecture was exploited in one of Lin et al.'s previous works used the same term intelligently, 1x1 convolutional operation was used in Inception-V3, which maps the input data into 3 or 4 different spaces smaller than the initial input space. Inception V3 then maps all correlations in these smaller 3D spaces via regular (3x3 or 5x5) convolutions [42].

4.8 Recurrent Neural Networks

Recurrent Neural Networks (RNN) is a type of neural network where nodes form a directed graph. RNN displays dynamic behavior for time-series data. The structure and layout of the RNN results in high accuracy for time series data. Two broad classes of Recurrent Neural Networks finite impulse and infinite impulse and both of them display dynamic behaviour. The infinite impulse RNN is a directed cyclic graph (DCG), whereas the finite impulse RNN is a directed acyclic graph (DAG) [43].

Traditional RNNs operate on the assumption that outputs of RNNs depend on the prior inputs within the sequence. RNNs can retrain information because of "memory" which captures information from previous inputs to determine the current inputs and outputs, which makes them very effective time series data. Long Short-Term Memory (LSTM) is a type of RNN that has been recognized for its ability to perform better in identifying and learning the sequential data very well. In addition, in terms of temporal sequence when compared to other deep neural networks, LSTM has proved to show a better learning ability [29].

Long Short-Term Memory can be considered to work with a cell state and a carry state for maintaining the information in the form of a gradient, which is assured to be not lost as the input sequence is processed further in the architecture. At each time step in the Long Short-Term Memory considers the current state, the carry state, and the cell state. Figure 11 shows the architectural anatomy of the Long Short-Term Memory network [43][45].

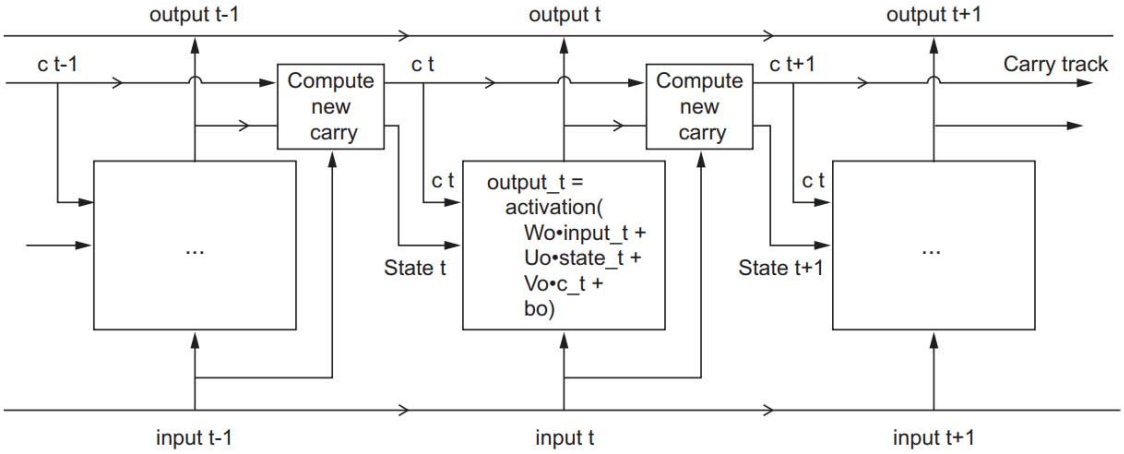


Figure 11. Anatomy of a Long Short-Term Memory (LSTM) Neural Network [43][45]

Long Short-Term Memory has three different gates and weight vectors. There is a forget gate for forgetting the non-essential information. The input gate is responsible for working and handling the current input. Finally, the output gate further generates the predictions for each time step [46][47]. It must be noted that the function of each cell element is dependent on the weights of the network layers, which are initialized and adjusted during training, as we saw in the Convolutional Neutral Network training process discussed above.

Chapter 5

Results and Analysis

5.1 Preliminaries

The dataset used for this research was a subset of World-Level American Sign Language Dataset [8]. A custom dataset can also be used, or a smaller subset can be utilized due to certain inconsistencies present in the original dataset. The presence of numerous interpretations of a single sign gesture and ambiguity in the signs prompt to lower the accuracy of the model due to similar characteristics being removed during the training process for different signs. The most frequent 100 words in American Sign Language (ASL) were used and the models with the top 10 classes and top 3 classes were tested as after scraping the data, there were 2000 classes which is too large to process as videos, so it was decided to use a maximum of 100 classes. For better understanding of the word-level sign recognition task the dataset was sorted into descending order according to the sample number of videos per class. Different meanings can have similar sign gestures, resulting in classification errors. For this reason, the models are evaluated using the top k = {3, 10, 100} classes using prediction of highest probability value. The following are the top 100 classes that were used for this research.

'accident', 'africa', 'all', 'apple', 'backpack', 'basketball', 'bed', 'before', 'bird', 'black', 'blue', 'book', 'bowling', 'brown', 'but', 'can', 'candy', 'chair', 'change', 'cheat', 'city', 'clothes', 'color', 'computer', 'cook', 'cool', 'corn', 'cousin', 'cow', 'dance', 'dark', 'deaf', 'decide', 'dog', 'drink', 'eat', 'enjoy', 'family', 'fine', 'finish', 'fish', 'forget', 'full', 'give', 'go', 'graduate', 'hat', 'hearing', 'help', 'hot', 'jacket', 'kiss', 'language', 'last', 'later', 'letter', 'like', 'man', 'many', 'meet', 'mother', 'no', 'now', 'orange', 'paint', 'paper', 'pink', 'pizza', 'play', 'pull', 'purple', 'right', 'school', 'secretary', 'shirt', 'short', 'son', 'study', 'table', 'tall', 'tell', 'thanksgiving', 'thin', 'thursday', 'time', 'visit', 'wait', 'walk',

'want', 'water', 'what', 'white', 'who', 'wife', 'woman', 'work', 'wrong', 'year', 'yellow', 'yes'

Each video was then preprocessed before being utilized as an input to the model. Preprocessing the model involved initially avoiding background noise from the videos. The dataset was expanded to increase the size of training set for better performance.

5.2 Model Testing

The deep learning model proposed for recognition of hand gestures in ASL videos is shown in Figure 12.

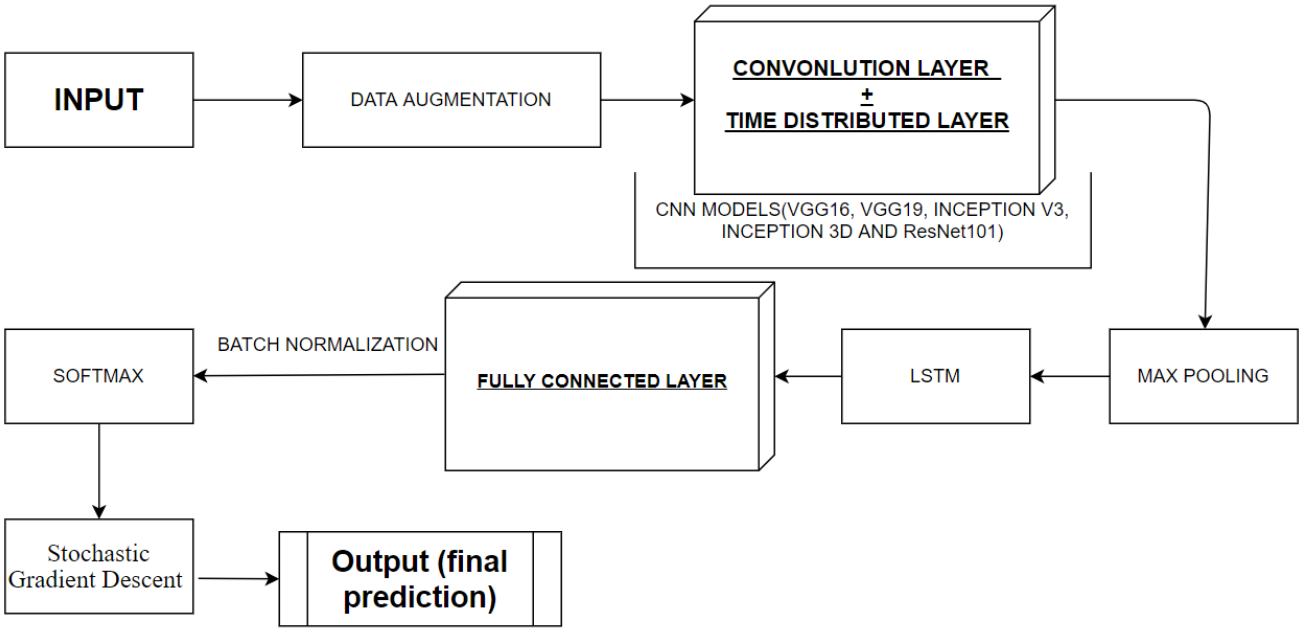


Figure 12. Cascaded CNN – LSTM network architecture for video classification

To begin with, we downloaded a dataset, namely World-Level American Sign Language (WLASL) [8]. WLASL contain 2000 classes. Among those 2000 classes we used 100 classes for this research. Moving forward, classes were used to further extract the hands from the videos of the gestures using the YOLOv3 model which is pre-trained on hand. The obtained hand videos are then converted into frames using sliding video data generator. The data set was partitioned into 85:15 with 85% for training and 15% videos in a separate folder for testing. The 85% training data was split into 70% of the data for training and 15% for validation.

Moving forward with the training process of the model, the frames we obtain from the sliding video generator are the frames of the hand gestures. These frames are then processed using the data augmentation technique to increase the sample size of training data in Convolutional Neutral Networks. The frames we get after applying the data augmentation technique is used as input frames that have the size of 224x224 with 3 RGB channels to the first layer of CNN models (VGG16/ VGG19/ Inception-3D/ Inception v3/ ResNet 101) which is convolution layer along with the time distributed layer. Time distributed layer is useful as we have used video frames as our input and it further enables us to use a single model for each input instead of several input models. After the convolution layer, max pooling layer is added to reduce the parameters to learn the model well and the amount of computation performed in the network, which helps to improve the efficiency of the network.

The output from the max pooling layer is input to a large network of 256 LSTM units. The large layer of 256 LSTM units is followed by a fully connected layer. A fully connected layer connects every neuron of the previous layer to every neuron of the next layer with the number of neurons. After the fully connected layer, for final prediction a SoftMax layer is used to train at the final stage. Before applying the SoftMax, we used batch normalization as it has the effect of

stabilizing process and dramatically reduces the number of epochs required to train the model. Moreover, by adding the batch normalization between the layers of network, it helps to improve the efficiency of the model and prevent the model from overfitting. The final layer consists of a gradient descent optimizer, a stochastic optimizer that minimizes the categorical cross entropy of the given loss function. The loss function is used to apply back-propagations and repeated iterations to converge and learn the training data. The model's hyperparameters were set at compilation which follows epochs and a dropout. For stochastic gradient descent, ADAM optimizer was selected for regularization. The dropout was selected by executing a number of epochs and comparing the accuracy of prediction. As dropout was changed from 0.1 to 0.3 and epochs size was increased up to 100 to get higher accuracy and prevent the model from overfitting.

The test set was converted to NumPy-array to store the frames into NumPy-array which was used for prediction and then top-k approach was used to obtain the accuracy.

In ASL, different meanings can have similar sign gestures, resulting in classification errors. For this reason, the models are evaluated using the top k = {3, 10, 100} classes using prediction of highest probability value.

With the same model, we have used different layer configurations for comparing the accuracy on the cascaded CNN-RNN architecture. The different layer configurations refer to the number of layers used in our models based on CNN-RNN configurations, such as VGG16/VGG19 models have 16 and 19 hidden layers respectively, and similarly, other models have different number of layers. InceptionV3 has 48 layers, Inception3D has 42 layers and ResNet-101 has 101 layers. These hidden layers are basically repeated blocks of convolutions with an input in the beginning

and finally an output with proper activation to get the predictions, which is SoftMax in our case. The best model was selected by testing with different combinations of dropout variance, the number of LSTM layers and the number of LSTM nodes (256 LSTM units, 512 LSTM units and three layers of 64 LSTM units). The model with 256 units gave the best performance among the different alternatives. Model comparison was based on the performance metric of the accuracy of class estimation.

5.3 Results and Discussion

Figure 13 shows the comparison of the models in a boxplot. The Inception3D-LSTM has performed the best, followed by InceptionV3-LSTM, Resnet101-LSTM, VGG19-LSTM and finally VGG16-LSTM.

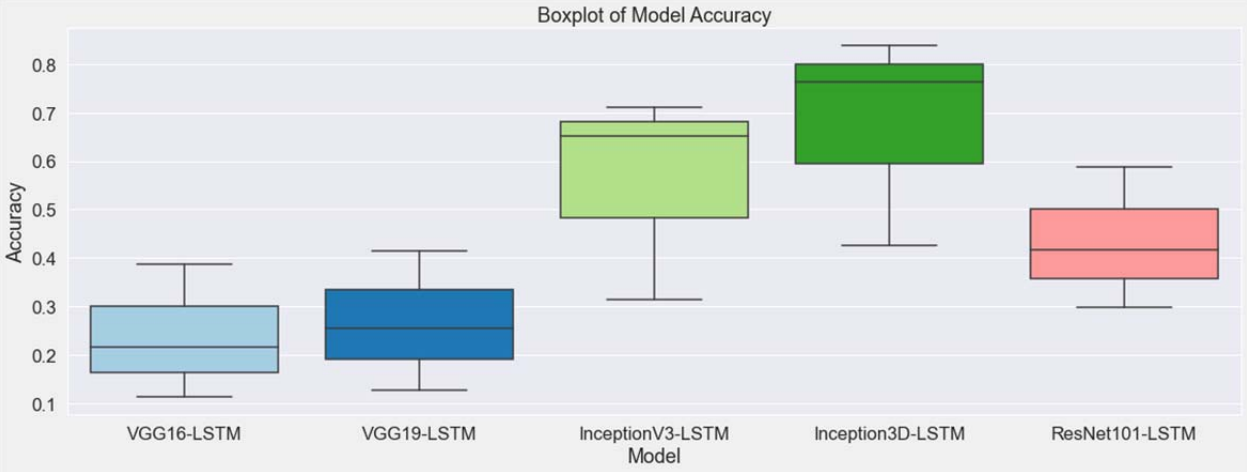


Figure 13. Boxplot of Model Accuracies

From the boxplot in Figure 13, we observe that dark green color shows the highest accuracy of 83%, which is achieved by Inception3D-LSTM, followed by InceptionV3-LSTM with 71% accuracy, which is denoted by light green color, next using ResNet-LSTM with an accuracy of 58% was achieved that is denoted by light red color. We got the lowest accuracy by using VGG19 and VGG16, which is 41% and 38%, respectively. The boxplot of VGG19 and

VGG16 is denoted by dark blue and light blue colors correspondingly.

Figure 14 shows the model accuracies comparison based on the number of classes, where we can see green color shows the top 100 classes, orange color top 10 classes and blue color shows the top 3 classes. It can be seen that when the number of classes are higher, the overall accuracies is reduced, on the other hand when we select fewer number of classes, the accuracy increases. This is mainly due to the ambiguity in the gestures for different meanings and the number of videos available in each class. Some of the classes have both facial and hand gestures. As only hand gestures were retained for recognition of the sign, such gestures could not be predicted accurately.

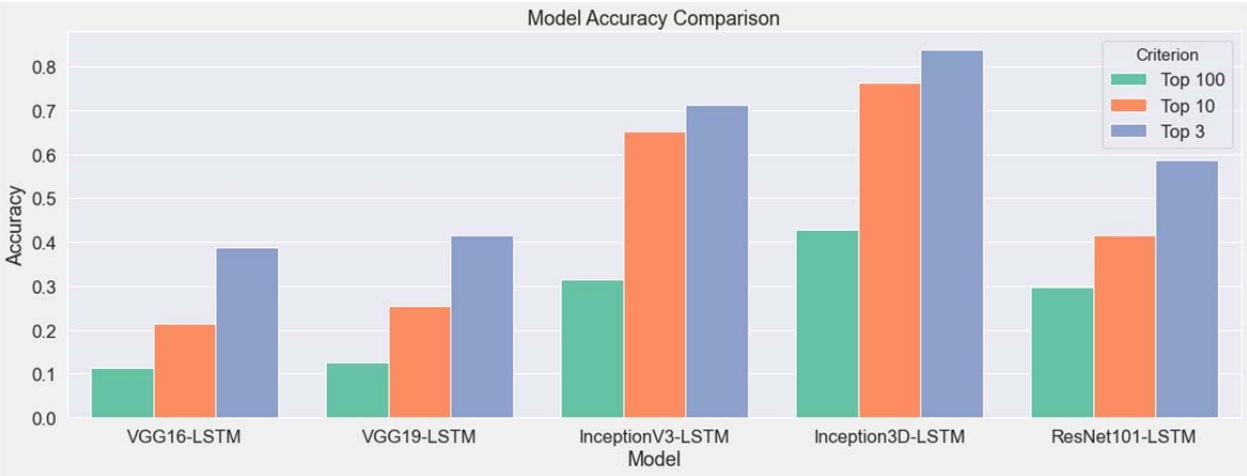


Figure 14. Model accuracies comparison based on the number of classes

Figure 15 shows the top-k accuracies comparison of different models. From the graph it can be observed that Inception3D-LSTM gives the highest accuracy among all classes. This is followed by InceptionV3-LSTM, ResNet101-LSTM, VGG19-LSTM and finally VGG16-LSTM.

Figure 15. Top-k accuracies Comparison of different models

Table 2 shows the accuracy of all the models for top-k classes.

Table 2. Summary Table of Model Accuracy

Model	**Criterion**	**Accuracy**
VGG16-LSTM	Top 100	0.1127
VGG19-LSTM	Top 100	0.1265
InceptionV3-LSTM	Top 100	0.3141
Inception 3D-LSTM	Top 100	0.4269
ResNet101-LSTM	Top 100	0.2985
VGG16-LSTM	Top 10	0.2155
VGG19-LSTM	Top 10	0.2542
InceptionV3-LSTM	Top 10	0.6521
Inception 3D-LSTM	Top 10	0.7621
ResNet101-LSTM	Top 10	0.4156
VGG16-LSTM	Top 3	0.3875
VGG19-LSTM	Top 3	0.4152

InceptionV3-LSTM	Top 3	0.7113
Inception 3D-LSTM	Top 3	0.838
ResNet101-LSTM	Top 3	0.5875

5.4. Statistical Analysis

Deep learning models are statistical and probabilistic which work based on calculations and statistical methods and recognizing patterns in the data and to make forecasts. There are chances that observations which consists of drawing samples from a population indicates an effect which can occur because of sampling errors. If the output of a model indicates a p-value < 0.05 indicating 95% confidence interval, a conclusion can be made from the assumptions that the output values reflect the characteristics of the entire population. On the other hand, if the p-value > 0.05, the output values do not reflect the characteristics of the entire population. Statistical significance tests are used to evaluate whether the differences between the models are significant or occurred by some coincidence. The mean values of the performance metrics for different machine learning methods were tested for statistically significant differences. ANOVA (One-way analysis of variance) was performed to determine that the performance of different models had statistically significant differences [49].

However, to perform ANOVA, the data should satisfy the following assumptions: (a) normal distribution (b) homogenous variance (c) absence of significant outliers and (d) independence of observations [50]. Data normality was tested by Shapiro-Wilk normality test [51] and homogeneous variance was checked through Levene's analysis [52]. Let us consider the null hypothesis (H0) as "all models show similar performance" and the alternative hypothesis

(H1) states “all models show different performance”. Ho is accepted if the mean values of the performance metrics for different models do not show statistically significant differences. The alternate hypothesis (H1) is accepted and H0 is rejected if the mean values of the performance metrics show statistically significant differences, i.e., p-value < 0.05. A Tukey post-hoc test [52] was also performed to identify the models that demonstrate statistically significant differences in the performance metrics of the models. A custom R script was used to perform the statistical analyses.

Normality check was conducted on the Accuracy scores of the models presented in the Q-Q plot and Shapiro test’s results based on the assumptions. It cannot be stated that "the sample does not have a normal distribution", but only " the theory that the sample comes from a population which does not have a normal distribution can be rejected". But the sample does not have a fair normal distribution looking at the Q-Q plot as shown in Figure 16, but we would not expect it to as it is only a sample.

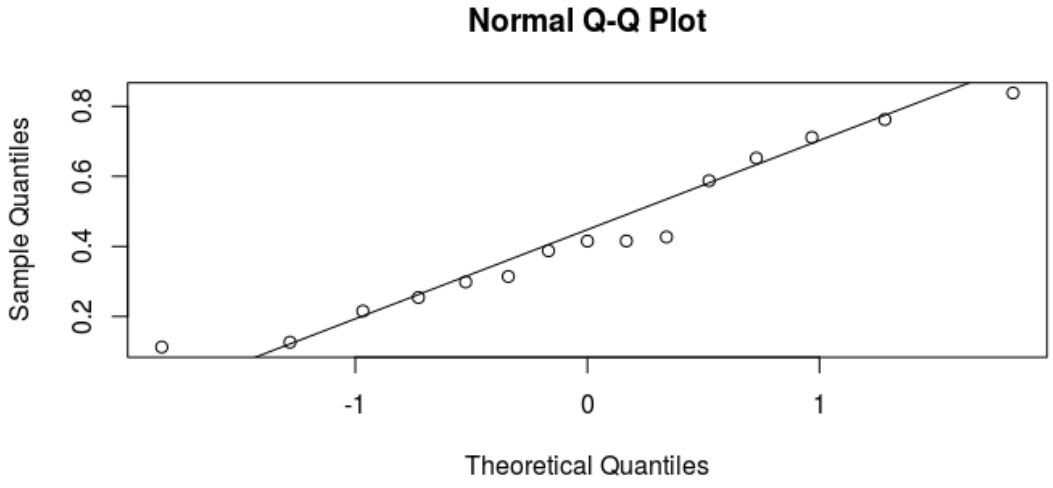


Figure 16. Q-Q plot Normality check for the distribution

The following results show the output for the Shapiro's test.

```
Shapiro-Wilk normality test

data:  df$Accuracy
W = 0.94597, p-value = 0.4633
```

From the output, the p-value which is more than 0.05 implying that the distribution of the data is normal distribution.

Levene's test was performed next. Let us consider the null hypothesis H0 as “all variances are equal”. Note that this test is meant to be used with normally distributed data but can tolerate relatively low deviation from normality.

```
Levene's Test for Homogeneity of Variance (center = mean)
      Df F value Pr(>F)
group  4  0.6534 0.6376
      10
```

The test reveals a p-value > 0.05, testifying that Ho is accepted, i.e., there are no significant difference between the group variances. The ANOVA analysis was conducted on the model as groups as well as the feature selection methods as the groups. The following output shows the one -way ANOVA for models and feature selection methods. One-way ANOVA for Models and Accuracy is shown in the results below.

```
           Df Sum Sq Mean Sq F value Pr(>F)
Model       4 0.4222 0.10554   3.402 0.05.
Residuals  10 0.3102 0.03102
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The p-value is equal/less than the usual threshold of 0.05. So, it can be stated confidently that there is a statistical difference between the models. One-way ANOVA for Number of Features Selection Method and Accuracy is shown in the results below.

```
          Df Sum Sq Mean Sq F value Pr(>F)
Criterion    2 0.2807 0.14033   3.728 0.05.
Residuals   12 0.4517 0.03764
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The p-value is again equal to the usual threshold of 0.05. So, it can be said that there is a statistical difference between the number of features selected.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

Deep learning models were used to identify dynamic gestures for American Sign Language and obtain the best performing model. The aim was to modify and test different deep learning models to increase the accuracy of prediction. The hand gesture recognition methods presented in this thesis were implemented using ordinary laptop computer. Experiments were performed on normal speed of sign detection videos from Word-Level American Sign Language Database (WLASL) for the hearing impaired and achieved around 83% accuracy on top 3 classes and 42% accuracy in top 100 classes. It was observed from the results that as the number of classes increase, the ambiguity also increases. The reason for increase of ambiguity is similar kind of features in different classes; this directly affects the accuracy, and it decreases as we increase the number of classes. It was also concluded that from top 100 classes some signs include facial expression as well, it leads to lower accuracy as we have extracted only hands from the videos because of this constraint model doesn't get the exact sample for prediction. As the computational complexity of the model depends on the number of classes and samples for each class, the classification process was challenging. Although the results are not conclusive, promising results were obtained for top 3 videos whereas the performance deteriorated with top 100 videos, which encourages further research.

In this thesis a methodology was proposed to use deep learning to understand American Sign Language. There is a way to tackle the problems faced by the persons with impaired

hearing and speech. It consists of two key components that analyses and assign the signs from videos into numbers of classes. Different cascaded networks were examined that were based on CNN and RNN, where, in the initial phase after preprocessing the data, the data was fed into CNN for extraction of spatial features, and then the data was passed to RNN for temporal feature extraction and the final layer used SoftMax for probabilistic confidence of the different classes. The overwhelming number of features in a vector of 2048 increases the complexity as a greater number of features are going through the dense layer, due to which the network gives bad prediction. Experimenting with various Recurrent Neural Network (RNN) architectures for the performance of the pooling layer will be one of the possible enhancements with Gated Recurrent Unit (GRU) and Independent RNNs included. Using Capsule Networks instead of VGG16, VGG19, ResNet can produce better results in terms of CNN improvements. CNN and RNN models can be integrated into one ensemble model. Typically, the usage of two separate models that feed into one another, results in the loss of data and the training time increases, whereas the use of one ensemble facilitates careful control of input data and reliable model corrections.

6.2 Limitation and Future work

The model proposed in this thesis predicts well when the number of classes range less than 10 with a greater number of videos per class and one of its drawbacks is bad generalisation on large dataset (a greater number of classes). Since the prediction is not robust it is not a very good model to be used for commercial use, but it can be improved with large dataset. For better results, a dataset is required which has more videos per class. There can be a possible solution using posenet, to make a single shot detector, which can help to build a robust model. Extracting features of facial expressions is more challenging compared to feature extraction of hand gestures in videos, consequently facial expression recognition is a possible future work.

References

[1] Passi, K. and Goswami, S., 2019, December. “Real Time Static Gesture Detection Using Deep Learning. In International Conference on Big Data Analytics,” . Springer, Cham, pp. 408-426 Available Online: https://zone.biblio.laurentian.ca/bitstream/10219/3468/1/Thesis-Sandip%20Goswami.pdf .

[2] Emond A, Ridd M, Sutherland H, Allsop L, Alexander A, Kyle J. “The current health of the signing Deaf community in the UK compared with the general population: a cross-sectional study,”. BMJ Open. 2015, 5(1).

[3] Crasborn, O., & Van Der Kooij, E. (2013). “The phonology of focus in Sign Language of the Netherlands,”. Journal of Linguistics, 49(3), 515-565. doi:10.1017/S0022226713000054.

[4] Mitchell, Ross; Young, Travas; Bachleda, Bellamie; Karchmer, Michael (2006). "How Many People Use ASL in the United States ? Why Estimates Need Updating,". Sign Language Studies. 6 (3). ISSN 0302-1475. Retrieved November 27, 2012.

[5] Meyer, C.F. “Introducing English Linguistics International Student Edition,” 2010. Publisher: Cambridge University Press. url: https://books.google.ca/books?id=MWbrvUiYzSkC.

[6] Drasgow E. “American Sign Language,”. Publisher: Encyclopedia Bitannica, July 2019. url: https://www.britannica.com/topic/American-Sign-Language.

[7] National Institute on Deafness and Other Communication Disorders (NIDCD), “American Sign Language,”. NIH Publication No. 11-4756, 2019. Website: https://www.nidcd.nih.gov/order.

[8] Li, D., Rodriguez, C., Yu, X. and Li, H., 2020. “Word-level deep sign language recognition from video: A new large-scale dataset and methods comparison,”. In the IEEE Winter Conference on Applications of Computer Vision (pp. 1448-1458).

[9] Lee B.G., Lee S.M. “Smart Wearable Hand Device for Sign Language Interpretation system with Sensors Fusion,”. IEEE Sens. J. 2018;18:1224–1232. doi: 10.1109/JSEN.2017.2779466. Available online from: https://www.mdpi.com/1424-8220/20/21/6256/htm .

[10] T.W. Chong and B.G. Lee, "American Sign Language Recognition Using Leap Motion Controller with Machine Learning Approach,". Sensors, vol. 18, 2018.

[11] Matheson G. “6 Myths About Sign Language [Internet],”. Ai. [cited 2019Apr24]. Available from: https://blog.ai-media.tv/blog/6-myths-about-sign-language.

[12] Cohen, I. Sebe, N., Chen, L., Garg, A., and Huang, T.S. (2003) “Facial expression recognition from video sequences: temporal and static modelling,” Computer Vision and Image Understanding, pp.160–187.

[13] Nandy, Anup & Prasad, Jay & Mondal, Soumik & Chakraborty, Pavan & Nandi, G. (2010). “Recognition of Isolated Indian Sign Language Gesture in Real Time,”. 70. 102-107. 10.1007/978-3-642-12214-9_18.

[14] Huang, Jie, et al. “Video-based sign language recognition without temporal segmentation,”. Proceedings of the AAAI conference on Artificial Intelligence, Vol. 32. No. 1. 2018.

[15] C. Linqin, C. Shuangjie and X. Min, "Dynamic hand gesture recognition using RGB-D data for natural human-computer interaction,". Journal of Intelligent and Fuzzy Systems, 2017.

[16] F. Ronchetti, Q. Facundo and A. E. Cesar, "Handshake recognition for argentinian sign language using probsom,". Journal of Computer Science & Technology, 2016.

[17] Cate, Hardie, Fahim Dalvi, and Zeshan Hussain. “Sign Language Recognition using temporal classification,”. arXiv preprint arXiv:1701.01875 (2017). Retrieved from https://arxiv.org/abs/1701.01875

[18] Garcia, B. and Viesca, S.A., 2016. “Real-time American sign language recognition with convolutional neural networks,”. Convolutional Neural Networks for Visual Recognition, 2, pp.225-232.

[19] Simonyan, K. and Zisserman, A., 2014. “Very deep convolutional networks for large-scale image recognition,”. arXiv preprint arXiv:1409.1556. Website: https://arxiv.org/abs/1409.1556.

[20] He, K., Zhang, X., Ren, S. and Sun, J., 2016. “Deep residual learning for image recognition,”. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[21] Santhalingam, P.S., Pathak, P., Košecké, J. and Rangwala, H., 2020, October. “Body Pose and Deep Hand-shape Feature Based American Sign Language Recognition,”. In 2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA) (pp. 207-215). IEEE.

[22] Patrice. Ferlet., 2019. “Training a neural network with an image sequence – example with a video as input, ”. Smile Innovation a series of articles about innovative technologies from medium publication. Available at: https://medium.com/smileinnovation/training-neural-network-with-image-sequence-an-example-with-video-as-input-c3407f7a0b0f.

[23] D. Lu, C. Qiu and Y. Xiao, "Temporal Convolutional Neural Network for Gesture Recognition," 2018. IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS), Singapore, 2018, pp. 367-371, doi: 10.1109/ICIS.2018.8466467.

[24] Bheda. Vivek and Dianna Radpour. "Using Deep Convolutional Networks for Gesture Recognition in American Sign Language,". arXiv abs/1710.06836 (2017). Retrieved from: https://arxiv.org/abs/1710.06836

[25] Yamashita, R., Nishio, M., Do, R.K.G. et al. "Convolutional neural networks: an overview and application in radiology," Insights Imaging 9**,** 611–629 (2018). https://doi.org/10.1007/s13244-018-0639-9.

[26] Barioul, R., Ghribi, S.F., Derbel, H.B.J. and Kanoun, O., 2020, June. "Four Sensors Bracelet for American Sign Language Recognition based on Wrist Force Myography,". In 2020 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA) (pp. 1-5). IEEE.

[27] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A., "Going deeper with convolutions," 2015. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).

[28] Nasr, G.E. and Badr, E. A. and Joun, C., title: "Cross Entropy Error Function in Neural Networks: Forecasting Gasoline Demand," 2002. Isbn: 157735141X, publisher: AAAI Press, book: Proceedings of the Fifteenth International Florida Artificial Intelligence Research Society Conference, (pp. 381-384).

[29] Kousai Smeda., “Understand the architecture of CNN,” 2019. Available online at: https://towardsdatascience.com/understand-the-architecture-of-cnn-90a25e244c7.

[30] Dogo, Eustace & Afolabi, Oluwatobi & Nwulu, Nnamdi & Twala, Bhekisipho & Aigbavboa, Clinton. (2018). “A Comparative Analysis of Gradient Descent-Based Optimization Algorithms on Convolutional Neural Networks,”. doi: 10.1109/CTEMS.2018.8769211. Conference: 2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS), K.L.S. Gogte Institute of Technology, Belagavi, Karnataka, India

[31] Raul Gomez, (2018), “Cross-Entropy Loss [ONLINE],”. Available at: https://gombru.github.io/assets/cross_entropy_loss/intro.png [Accessed 22 July 2020].

[32] S. Ruder, "An overview of gradient descent optimization algorithms," 2016. Retrieved from http://arxiv.org/abs/1609.04747, 29 October 2018.

[33] T. Schaul, I. Antonoglou and D. Silver, "Unit Tests for Stochastic Optimization,". arXiv Preprint arXiv:1312.6055, Dec 20, 2013. Retrieved from https://arxiv.org/abs/1312.6055.

[34] H. Robbins and S. Monro, "A stochastic approximation method in Herbert Robbins and Sutton Monro,”. Berlin, Germany: Springer, 1985, pp. 102–109.

[35] Duchi, John & Hazan, Elad & Singer, Yoram. (2011). “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization,”. Journal of Machine Learning Research 12, no. 7. pp. 2121-2159.

[36] L. Bottou, "Stochastic gradient learning in neural networks,". Proceedings of Neuro-Nimes, vol. 91, no. 8, p. 12, 1991.

[37] T. Tieleman and G. Hinton, "Lecture 6.5-RMSPROP: Divide the gradient by a running average of its recent magnitude,". COURSERA, Neural Network Machine Learning, vol. 4, no. 2, pp. 26–31, 2012.

[38] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization,". Dec. 2014, arXiv:1412.6980. [Online]. Retrieved from https://arxiv.org/abs/ 1412.6980.

[39] Nitish Srivastava and Geoffrey Hinton and Alex Krizhevsky and Ilya Sutskever and Ruslan Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,”. Journal of Machine Learning Research., vol.15, no. 56, pp.1929-1958, 2014.
url: http://jmlr.org/papers/v15/srivastava14a.html

[40] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift,". arXiv preprint arXiv:1502.03167 (2015). Retrieved from: https://arxiv.org/abs/1502.03167.

[41] Luke T, Geoff N. “Improving deep learning using generic data augmentation,”. arXiv preprint. 2017. Retrieved from: https://arxiv.org/abs/1708.06020.

[42] Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. “Rethinking the inception architecture for computer vision,”. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 2818–2826.

[43] H. Sak, A. Senior and F. Beaufays, "Long Short Term Memory recurrent neural network architectures for large scale acoustic modelling,". proceedings of the annual conference of the international speech communication association 2014.

[44] M. Milos, "Comparitive analysis of Recurrent and Finite Impulse Response Neural Networks in Time Series Prediction,". Indian Journal of Computer and Engineering, 2012, pp. 180-191.

[45] D. Britz, "Recurrent Neural Networks Tutorial, Part 1 – Introduction to RNNs,". WILDML, 17 September 2015. [Online]. Available: http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/.

[46] Schmidhuber, Jurgen. "Habilitation thesis: System modeling and optimization,". Journal of Page 150 ff demonstrates credit assignment across the equivalent of 1,200 layers in an unfolded RNN (1993).

[47] F. Gers, N. Nicol, and J. Schmidhuber, "Learning Precise Timing with LSTM Recurrent Networks,". Journal of Machine Learning Research, vol. 3, doi: 10.1162/153244303768966139, pp. 115-143, 2002.

[48] W. Tao, Ming C. Leu, Z. Yin, "American Sign Language alphabet recognition using Convolutional Neural Networks with multiview augmentation and inference fusion,". Engineering Applications of Artificial Intelligence, vol. 76, 2018, Pages 202-213, ISSN 0952-1976, https://doi.org/10.1016/j.engappai.2018.09.006.

[49] Zhang, Jun, and Peng Ching-An. "Enhanced Proliferation and Differentiation of Mesenchymal Stem Cells by Astaxanthin-Encapsulated Polymeric Micelles,". PLoS One, vol. 14, no. 5, Public Library of Science, May 2019, p. e0216755.

[50] T. K. Kim, "Understanding one-way ANOVA using conceptual figures,". Korean J. Anesthesiol., vol. 70, no. 1, p. 22, 2017.

[51] B. W. Yap and C. H. Sim, "Comparisons of various types of normality tests," J. Stat. Comput. Simul., vol. 81, no. 12, pp. 2141–2155, Dec. 2011.

[52] Y. J. Kim and R. A. Cribbie, "ANOVA and the variance homogeneity assumption: Exploring a better gatekeeper," Brit. J. Math. Stat. Psychol., vol. 71, no. 1, pp. 1–12, Feb. 2018.

[53] Sofiane Sahir, " Canny Edge Detection Step by Step in Python – Computer Vision", 2019. Available online at: https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123.

[54] Kasukurthi, Nikhil, et al. "American Sign Language Alphabet Recognition using Deep Learning", arXiv preprint arXiv: 1905.05487 (2019). Retrieved from: https://arxiv.org/abs/1905.05487