Islamic University of Technology (IUT)

Department of Computer Science and Engineering (CSE)

PointLSTM and Depth-CRNN based Hand Gesture Recognition

Authors

Amira Haque - 170041025

Mirza Zamiur Rahman - 170041039

Reeshoon Sayera - 170041061

Supervisor

Dr. Hasan Mahmud

Assistant Professor, Department of CSE

A thesis submitted to the Department of CSE

in partial fulfillment of the requirements for the degree of B.Sc.

Engineering in CSE

Academic Year: 2020-21

April - 2022

Declaration of Authorship

This is to certify that the work presented in this thesis is the outcome of the analysis and experiments carried out by Reeshoon Sayera, Mirza Zamiur Rahman and Amira Haque under the supervision of Hasan Mahmud,Assistant Professor, Department of Computer Science and Engineering (CSE), Islamic University of Technology (IUT), Dhaka, Bangladesh. It is also declared that neither of this thesis nor any part of this thesis has been submitted anywhere else for any degree or diploma. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Authors:

Amira Haque 13/05/22

Amira Haque

Student ID - 170041025

13/05/22

Mirza Zamiur Rahman

Student ID - 170041039

R Sayera 13/5/22

Reeshoon Sayera

Student ID - 170041061

Approved By:

Supervisor:

Dr. Hasan Mahmud

Assistant Professor

Systems and Software Lab (SSL)

Department of Computer Science and Engineering

Islamic University of Technology (IUT)

Acknowledgement

We would like to express our grateful appreciation for **Dr. Hasan Mahmud**, Associate Professor, Department of Computer Science & Engineering, IUT for being our adviser and mentor. His motivation, suggestions and insights for this research have been invaluable. Without his support and proper guidance this research would never have been possible. His valuable opinion, time and input provided throughout the thesis work, from first phase of thesis topics introduction, subject selection, proposing algorithm, modification till the project implementation and finalization which helped us to do our thesis work in proper way. We are really grateful to him.

We are also grateful to **Dr. Kamrul Hasan**, Professor, Department of Computer Science & Engineering, IUT for his valuable inspection and suggestions on our thesis work.

Abstract

Hand gestures represent spatiotemporal body language conveyed by various aspects of the hand, such as the palm, shape of the hand, and finger position, with the aim of conveying a particular message to the recipient. Computer Vision has different modalities of input, such as depth image, skeletal joint points or RGB images. Raw depth images are found to have poor contrast in the region of interest, which makes it difficult for the model to learn important information. Recently, in deep learning-based dynamic hand gesture recognition, researchers have attempted to combine different input modality to improve recognition accuracy. In this paper, we use depth quantized image features and point clouds to recognize dynamic hand gestures (DHG). We look at the impact of fusing depth-quantized features in Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) with point clouds in lstm-based multi-modal fusion networks.

Keywords— Dynamic-Hand-Gestures, Multimodal, Point Clouds, Depth Images, depth-quantized

Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Overview

Human activity and gesture recognition is an important component of the rapidly growing domain of ambient intelligence,in particular in assisting living and smart homes. Any spatio-temporal movement of the hand that is done with the intention of conveying a specific meaning can be considered as a hand gesture. Hand gesture recognition system has multiple modalities of input, such as depth based images, skeletal joint points or RGB images. It has been observed that depth images have low contrast in the hand region of interest while skeletal based methods highly rely on the quality of the estimated data. Compared to skeletal data, point clouds reflect the latent geometric structure of the object surface and contain rich spatial information, which provides reliable and complementary cues for gesture recognition. Moreover, in recent times multimodal learning has helped improve accuracy in gesture recognition.

In this paper, we combined the power of two deep learning techniques, convolutional neural networks with RNN and PointLSTM using depth images and their point clouds. We trained the CNN+RNN network with depth images as the CNN can extract features from the depth map while RNN works well in recognizing the sequences of movement and the PointLSTM network with point clouds which provides richer and reliable gesture recognition. Our work also focused on different types of fusion methods to extract deep learning features from the network predictions.

1.2 Problem Statement

To recognize dynamic hand gestures from benchmark dataset using deep learning architectures CRNN and PointLSTM based on depth and point cloud data.

1.3 Motivation & Scopes

Affluent spatial information can be obtained from point clouds providing us with authentic recognition of gestures. PointLSTM formulates gesture recognition as an irregular sequence recognition problem and aims to capture long term spatial correlations across point cloud sequences. PointLSTM has achieved state of the art results of 95.9 accuracy on the SHREC 17 dataset.[3]Moreover, fusion of scores from networks trained by different modality helps to achieve higher accuracy. Inspired from fusion of multi modal input networks, we wanted to extract the best features of depth images and point clouds. Hence, training CRNN on depth images and PointLSTM on point clouds may help improve accuracies after combining the scores.

1.4 Thesis Contribution

Our contribution in this work is the creation of an architecture by merging two existing architectures that utilizes multi modality of input and features (depth image and point cloud) and can give better results in recognizing hand gestures compared to existing architectures.

1.5 Research Challenges

In case of dynamic hand gesture recognition, efficiently capturing spatio-temporal information is a real challenge. Poor lighting conditions and difference in skin color limits the application of RGB cameras in detecting hand gestures. In case of complex backgrounds, variable external illumination and shadows of hands, the difficulty to achieve proper accuracy also increases. Other factors are environmental background, light and rotation, translation and scale changes etc.[5]

3D convolutional networks are rising in popularity to recognize gestures from video data due the advancement of deep learning. But a limitation to this method is that only a small portion of the frame of a video is occupied by the hand region and the rest is irrelevant information. [3] Sequence of hand joints can be estimated as an intermediate step due to the development of depth sensors but such skeletal based data heavily relies on the quality of the estimation.

1.6 Thesis Outline

In Chapter 1 we have discussed our study in a precise and concise manner. Chapter 2 deals with the necessary literature review for our study and the so far development done for hand gesture recognition.Here, we have also focused on our main related works. Firstly, a paper where we get introduced to the Point Net architecture. Secondly, the Point LSTM publication which focuses on point cloud based recognition. Lastly, the publication on the CRNN depth and skeleton based approach.In Chapter 3 we have stated the skeleton of our proposed method, proposed algorithm and also the flowchart to provide a detailed insight of the working procedure of our proposed method.In the initial part of Chapter 4 we gave an overview of the SHREC dataset and the things we have done for preparing the dataset and making it suitable for our architecture. The second part of chapter 4 includes our experimental analysis.The second last chapter is about our future plans, further possible improvements and conclusion.Lastly, the final segment of this study contains all the references and credits used.

Chapter 2 Literature Review

2.1 Overview

This section deals with the necessary literature review for our study and the so far development done for hand gesture recognition. Firstly, we gave a brief overview of the hand gesture recognition steps. Then the recognition approaches under camera vision-based sensor approach. Then, we have focused on our main related works. Firstly, a paper where we get introduced to the Point Net architecture. Secondly, the Point LSTM publication which focuses on point cloud based recognition. Lastly, the publication on the CRNN depth and skeleton based approach. This also includes an overview of multimodal fusion and the three types feature level fusion, score level fusion and decision level fusion.

2.2 Hand gesture recognition steps

2.2.1 Hand gesture frame acquisition

It is the method to capture the human hand gesture by the computer.
The nature of the acquisition is mainly of two types, one the static hand gesture recognition and second is the dynamic hand gesture recognition. Static hand gesture recognition is where each gesture is represented by a single image. Whereas, hand movements are used to express motions in dynamic hand gesture recognition.
There are two main approaches used to interpret hand gestures. The first one is the data glove based approach and the second one is the camera vision-based sensor approach. For the data glove based[4] approach according to hand movements or finger bending a physical response is detected by the sensors. The computer connected to the glove with wire is used to process the data that is collected. This glove-based system of sensors could be made portable by using a sensor attached to a microcontroller.

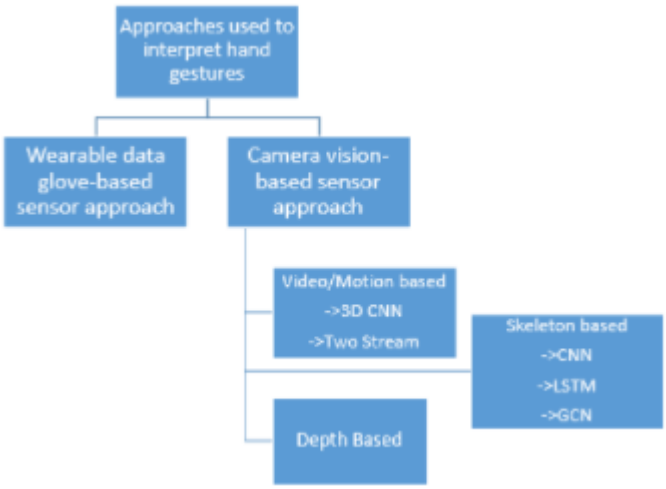

Figure 1: Approaches to interpret hand gestures

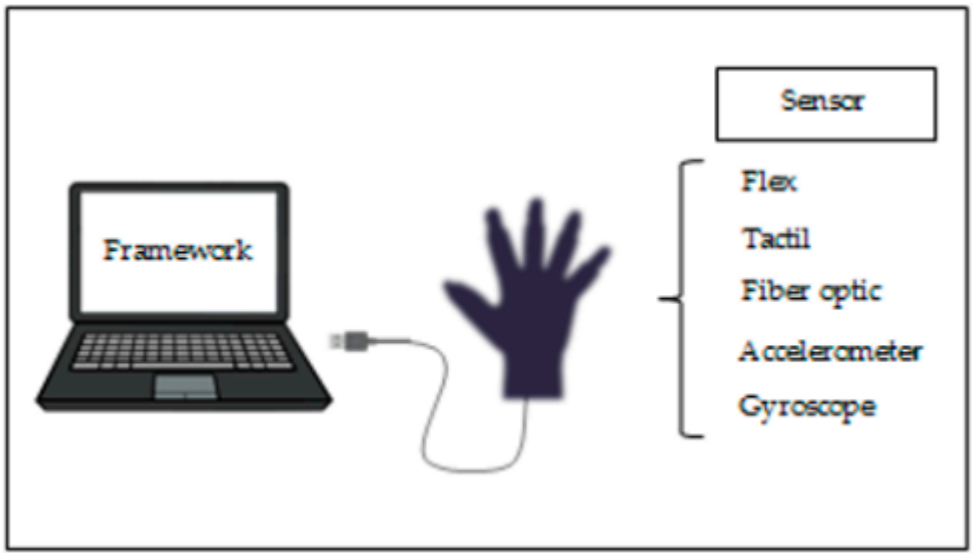

Figure 2: Data Glove Based Approach[4]

However, there a number of limitations of this approach. To begin with, the user needs to be physically connected to the computer making the interaction between user and computer more difficult. Moreover, this approach is not suitable for the elderly people who are suffering from chronic diseases that result in loss of muscle function, may be unable to wear and take off gloves, causing them discomfort and limiting them to use it for shorter periods. Additionally, sensors may also cause skin damage, infection or serious reactions in people with sensitive, allergic skin or those suffering burns. Lastly,

a very high-budget approach involving high costs.

For the camera vision-based sensor[4] approach the instrumented glove is replaced with a camera. There are many computer vision based algorithms that are involved that can detect hands using camera footage. The segmentation and detection of features such as skin color, appearance, skeleton, motion, depth, 3D model and many more are possible by such algorithms. It is a very promising and more cost effective method that does not require any wearable technology.

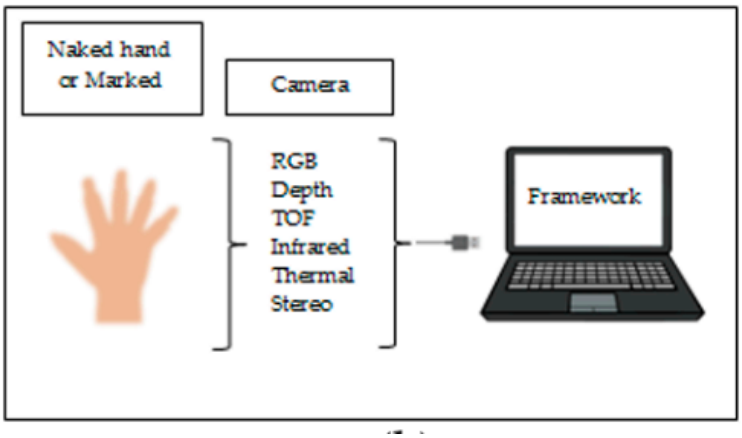

Figure 3: Camera Vision Based Approach[4]

However, for this approach the limitations include dissimilarity in the lighting, problems related to the complexity of background, the effect of occlusions, trade off between processing time against resolution and frame rate, objects which are foreground or background may present the same skin color tone or otherwise appear as hands.

2.2.2 Hand tracking

It is the ability of the computer to trace the hand and separate it from the background and from the surrounding objects.

It is done by using the two following way First, the user's hand and the background are given different shades of colors to be able to identify and remove the background using multi-scale color feature hierarchies.Secondly, the use of clustering algorithms that have the ability of treating each finger as a cluster and removing the empty spaces in-between them.

2.2.3 Feature Extraction

Features needed to be extracted changes and can vary from one application to another. Some of the features that could be taken into consideration are: fingers status, thumb status, skin color, alignments of fingers, and the palm position.

Features can be extracted using multiple several methods. To begin with, discrete Fourier Transform (DFT) operations of vertical and horizontal histograms can be applied. Techniques related to digital image processing like outline investigating based on edge recognition, Histogram Equalization, Median filtering, Average filtering,Morphological filtering, wavelet change are used to eliminate noise, to improve the contrast under different illumination, to separate the hand from the background and to cut the region containing the hand. Furthermore, extracting characteristics of the hand using Histogram of Oriented Gradients (HOG) can also be used. Lastly, to reduce the proportions and extract features of images of the human hand Principal Component Analysis (PCA) can be used.

2.2.4 Classification to reach output gesture

Features extracted are then sent to training and testing the classification algorithm to reach the output gesture. The different classification algorithms include:

Artificial Neural Networks (ANN):
As the name neural network suggests, these are computing systems that were developed from the inspiration of biological network of the neurons in animal brains. Similar to biological neurons, ANN have artificial neurons and these are connected with each other. In a biological brain there are synapses between neurons that transmits signals from one neuron to the other. The artificial neurons are also modeled as such and they transmit and process signals that it receives from other neurons. The signals in case of these artificial neurons are data in the form of real numbers and the processing is done using some non linear function of the sum of inputs. The neurons and the corresponding connections have some weights that changes with time as learning progresses. By increasing the weights we define the increase in strength of the signals in a certain connection and by decreasing the weights we define the reduction in strength. Typically,

we define layers as collections of some neurons and the neurons in different layers usually perform different transformations on their inputs.

K-nearest neighbor (KNN):

KNN is a non parametric, lazy algorithm used to distribute data into different classes and predict the class of a new data entry. By the term non parametric, it means not making assumptions about the data distribution and the data model is instead determined from the data. This is helpful because data in the real world does not follow any theoretical assumptions. So KNN can be used to classify data when there is no prior knowledge of the data. By lazy, it means that KNN does not learn the data and make a generilation to predict the next sample. Instead, it explicitly keeps all the training data during the classification of a new sample. KNN classifies a new sample by figuring out how close the sample resembles each class in the training data.

Support vector machine (SVM):

SVMs are supervised learning algorithms used for classification and regression. Both linear and non linear classification can be performed using SVM. Non linear classification is performed using kernel function where the kernels are homogenous polynomial, complex polynomilal, Gaussian radial basis function, and hyperbolic tangent function. The Gaussian kernel has a single parameter and so is preferred the most. SVM is the most commonly used machine learning technique since it outperforms most other methods.

Dynamic time warping algorithm:

The dynamic time warping (DTW) is an algorithm for measuring similarity between two time series sequences which may be of different speed. DTW can be applied on time series sequence of audio, video, graphics data or any data that can be turned into a linear sequence. The main idea of the algorithm is to compare two arrays with different length and build one-to-many relations between elements of different arrays.

Naive Bayes

Naïve Bayes uses the Bayes Theorem to provide a probabilistic machine learning algorithm that can be used in a variety of classification tasks. It's family of algorithms instead of being a single algorithm and all these algorithms share a common principle, "every pair of features being classified is independent of each other"

2.3 Recognition approaches under camera vision based

2.3.1 Video-based recognition

This approach can be utilized for detection purposes. It extracts the object through a series of image frames.

2.3.2 Skeleton-based recognition

This improves detection of complex features by specifying model parameters. It can also use different representations of skeletal data for classification.Moreover, this easily translates features and correlations of data by describing geometric attributes and constraints in order to focus on geometric and statistical features. Some common features that are used are joint orientation, space between joints, joint location, degree of angle between joints, curvature of joints etc.Furthermore, skeleton based approaches have the advantage of having a smaller data size compared to RGB-D images. The smaller data size results in less computational cost and skeletal data is also robust to phenomena such as variable lighting effects and occlusions. For example, it is possible to collect hand poses such as “grasp” using depth sensors by recording the joint locations in 3D skeleton data. These poses consist of a fixed number of hand bones and joints that can be inferred from video frames.

2.3.3 Depth-based recognition

A depth camera provides 3D geometric information about the object. The 3D data directly reflects the depth field in comparison to a color image which contains only a projection. The advantages of using this approach is the lighting, shade, and that there is no effect of color on the final image. Among the limitations there is cost, size and availability of the depth camera which in turn limits its usage.

2.4 Point Cloud

A point cloud is a collection of individual points in 3D space. Each point is represented by its coordinates in the XYZ plane. Difference between point clouds and other data such as text or audio or images is that here we work with sets.

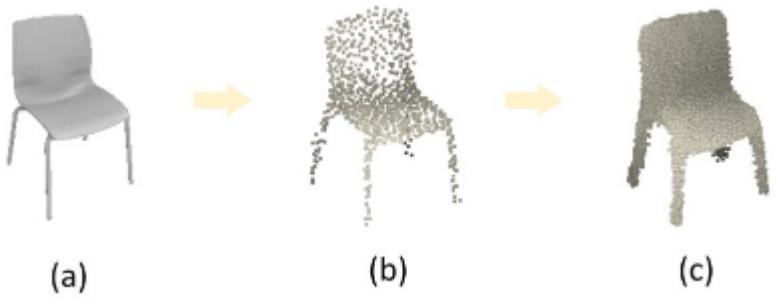

Figure 4: Point Cloud[7]

There are a few main properties of point clouds are [1]. Firstly, point clouds do not have any specific order. Therefore a network that takes in N input point clouds will have to be invariant to N! Permutations of the point clouds So, if the orders of the points are changed, we'll still obtain the same results. Secondly, the points in the point cloud are not secluded and neighboring points form a subset which carries a great weight hence the model has to be able to identify local structures from nearby points. Lastly, the different representations grasped of the point set should be invariant to rotation, translation and other forms of transformations.

There are generally three problems solved using point clouds [1].First one is from a given set of points, which is one point cloud, classifying as to what that object signifies. Second is breaking down one object into a number of objects. To differentiate between them, although the class is not determined here. This is known as part segmentation. Lastly, the semantic segmentation where points are given certain labels.

2.5 Point Net

A network that consists of linear layers which is also similar to transformation of each point from one dimension to another.
The Point Net Architecture has two networks, first the classification network and second the segmentation network. The classification network takes n cloud points as input and transforms the input points by an input transformer network.

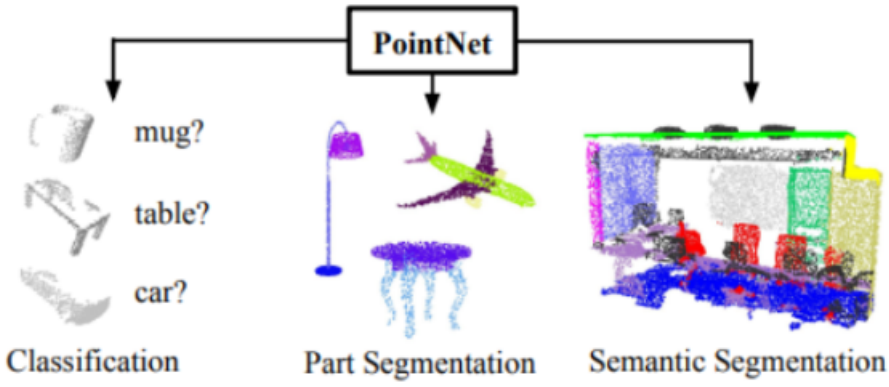

Figure 5: Problems solved using point cloud[1]

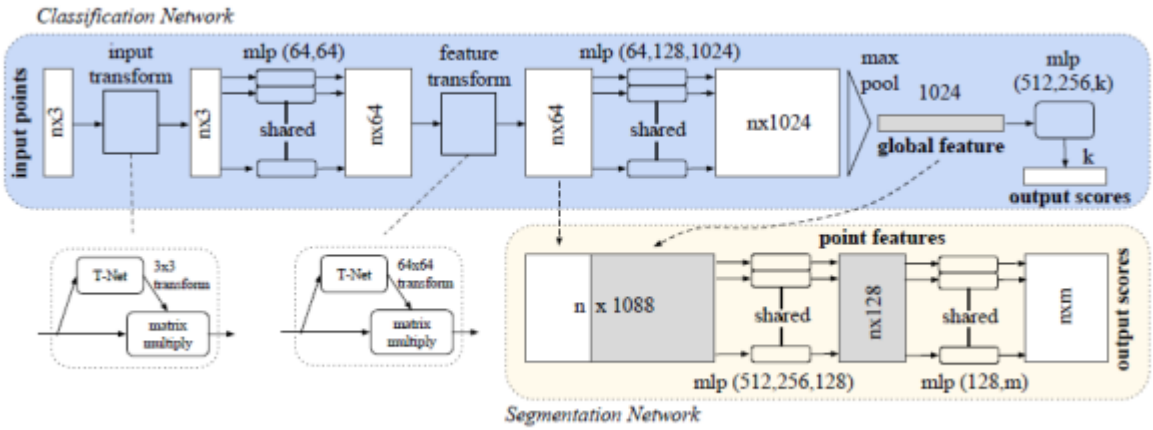

Figure 6: Point Net Architecture[1]

Layer 1 :

The main idea behind this layer is to align the input point cloud to a canonical space. The affine alignment is done because we want the point clouds to be invariant to certain geometric transformations such as rigid transformations, rotation, reflection so that when there is some rotation performed we don't want the coordinates and hence the results to be changed. So, to solve this some transformation matrix and T-net is used to predict the transformation. Depending on the input point cloud we have different transformation matrices. The T-net is a mini PointNet composed of basic modules of point independent features extraction, max pooling and fully connected layers and the T-net is trained with the rest of the network.[1]

Layer 2 :

After the input transformation, each point is then embedded by a shared multi-layer perceptron.[1]

Layer 3:

Subsequently a feature transformation is applied to align points in the embedding space. However, unlike the input transform, feature transformation is done in a much higher dimension which makes optimization difficult. So a regularization is added to the softmax training loss to make the optimization stable.[1]

Layer 4:

Each point is again transformed in another embedding space with 1024 dimensions.[1]

Layer 5:

A max pooling layer which aggregates all points in the high dimensional embedding space to output a global feature vector for making it permutation invariant.[1]

Layer 6:

Finally the global vector is updated by a multilayer perceptron to output the classification scores for k classes. The class with the highest probability is the class for the input point cloud.[1]

Adding to the classification network we get the segmentation network which joins global and local features and outputs per point scores. The global features are taken and expanded row by row where each row is 1024. Then a series of multilayer perceptrons are performed to get m scores at the end.

2.6 Point LSTM

A modified version of Long short-term memory (LSTM) for point cloud sequences which can capture long term relationships in a sequence. In the original architecture, each point in the point cloud sequence has independent hidden states and cell states. For each point in the current frame, relevant points in the previous frame are searched and state information from predecessors is collected. To reduce computational complexity, a simplified version of the PointLSTM was proposed with point shared states in the same frame. All points in the same frame have shared hidden states and cell states.

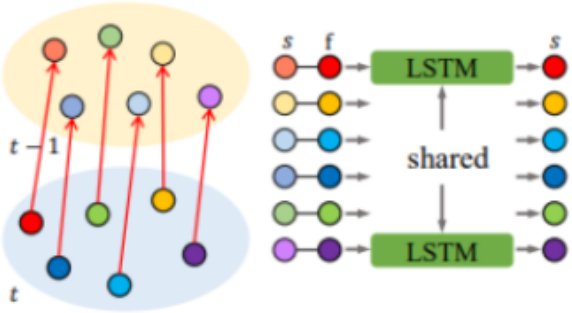

Figure 7: Point LSTM on Ordered Point Clouds[3]

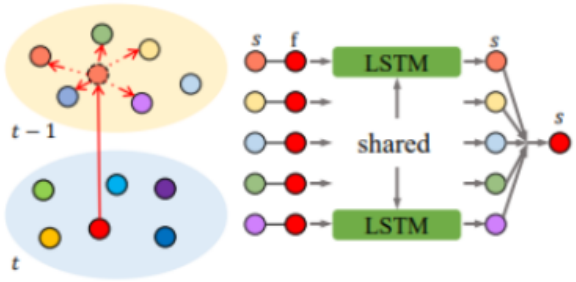

Figure 8: Point LSTM on Orderless Point Clouds[3]

In the figures above, s and f are states and features, respectively. In an ideal scenario, each point in the current frame can find the point in the previous frame which correlates with it.[3]

To further understand the point LSTM structure in a supplement paper of [3] they divided the network into different stages. Extraction of inter-frame features is done in stage one using spatial grouping. Then from stage two to stage four inter-frame features are extracted by spatio-temporal grouping and density based sampling. Finally, in the fifth and final stage all the information is gathered from all the timesteps to extract point wise features, and a max-pooling layer is followed to obtain global features.[3]

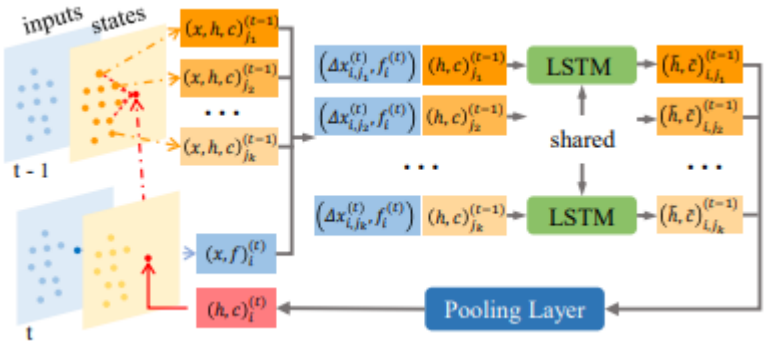

Figure 9: Point LSTM Architecture[3]

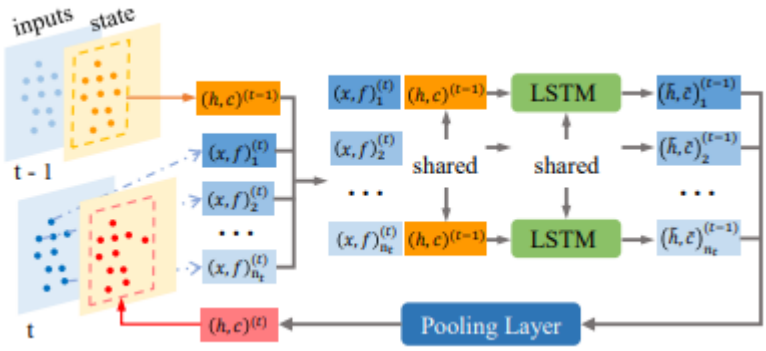

Figure 10: Point LSTM-PSS Architecture[3]

2.7 Depth Based CNN+RNN

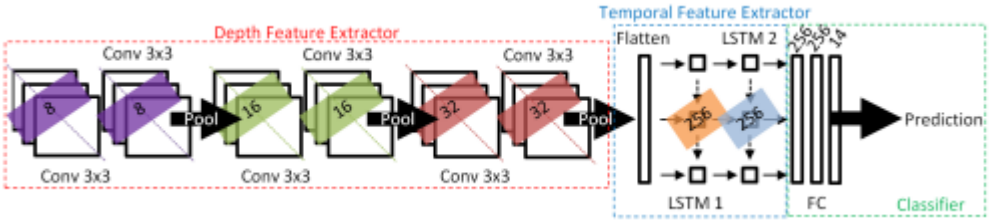

Figure 11: Depth Based CNN+RNN Architecture[2]

The power of two deep learning techniques, convolutional neural networks (CNN) and recurrent neural networks (RNN) is combined in [2] for automated hand gesture recognition using two modalities of input: depth and skeleton data. The two modalities of input are used to train two neural networks separately and later merged to get the final classification. This approach has achieve an overall accuracy of 85.46 on the dynamic hand gesture-14/28 dataset.

In this architecture features from a depth image sequence of a hand gesture is extracted using the CNN and since the gestures are dynamic and we have an ordered sequence of images, the RNN supplements the CNN to extract temporal features.

The architecture in [2] consists of three sections, the depth based feature extraction through CNN, the time series processing through RNN and lastly, the classification us-

ing a multilayer perceptron(MLP).

There are six 3x3 convolutional layers and a 2x2 max pooling layer between every two convolutional layers in the first CNN component. There are two LSTM layers in the RNN which is the second component and each of them consists of 256 LSTM units. Finally the MLP contains three fully connected layers which consist of 256, 256 and 14 units respectively.

2.8 Skeleton based LSTM

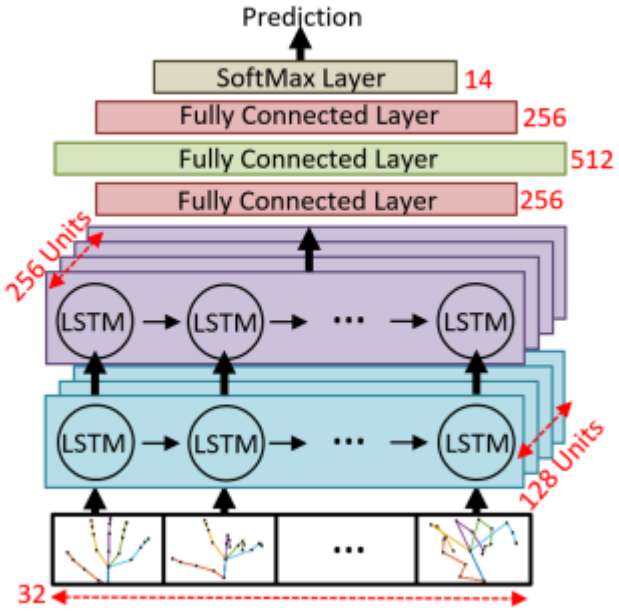

Figure 12: Skeleton-based LSTM network architecture[2]

RNN is used to extract the temporal patterns from the movements of skeleton points which are within an order. This is included in the second component of the system shown in [2]. The RNN structure is very similar to the one explained in the first component, except here there is an additional FC layer and multiple LSTM units under each layer. All-inclusive, in [2] we saw that the framework is composed of two network components that use skeleton and depth information for gesture recognition separately. Since each network has the ability of predicting a chosen gesture based on the chosen type of information, a higher performance is expected from the fusion of both the networks as

it will have the ability of extracting the best features from both the skeleton and depth-based spatial information and the fundamental temporal patterns between a collection of frames.

2.9 Multimodal fusion

In [2] we saw that the framework consists of two networks who independently have the potential of predicting hand gesture recognition based on the selected type of information. A higher performance is expected from the fusion of both the networks as it will have the ability of extracting the best features from both the depth and skeletal based data.

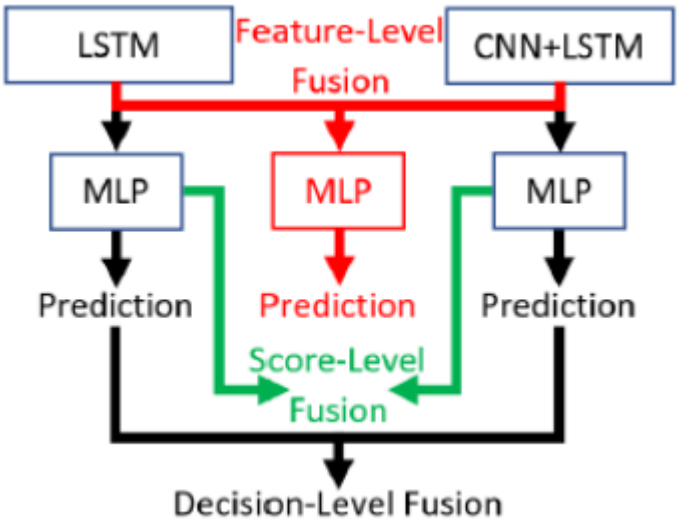

Figure 13: The three levels of fusion for the LSTM and CNN+LSTM networks.[2]

The three main fusion techniques considered in [2] :

2.9.1 Feature level fusion

After the input has been passed through a sequence of successive convolutional/LSTM layers then the feature-level fusion can be applied. After the fusion is applied, a feature-level fusion network needs to be made for which any kind of classifier like multi-layer perceptron or support vector machine can be linked and this process is carried out by the multi-layer perceptrons.

2.9.2 Score level fusion

A score-level fusion can be performed either after or between the fully-connected and soft-max layers. Here an assumption is made that the extraction of features from the input data has been successful and that has been passed through a selected classifier to get the score or probability. A fusion here provides a more reliable result since the scores obtained are heavily congruent to the network's prediction. Hence, any kind of combination of the scores from different networks will give a more reliable result.

2.9.3 Decision level fusion

The decision-level fusion is very similar to a score-level fusion except in the decision-level fusion the fusion is done after the network's prediction. Here the output that is predicted from the network is used in the fusion and the fusion is not related to any score or probability. The output from the soft-max layer gives us a probability or score on which the network's predicted output totally depends on. There are many methods using which a prediction can be generated from the network but the two most common are using decision threshold and ranking order. The first technique is where a random value is selected for each network as the threshold and any scores above the threshold are accepted and other scores are rejected. The second technique involves a grouping of the scores where the top of the list indicates a higher probability of getting accepted. A commonly used method following the second technique is the rank-1 where the top most prediction is taken as the network's predicted output. In [2] they have focused on only score-level and feature-level because decision-level is strongly related to the score-level and the number of networks in [2] which would've been used for the decision-level fusion

is only two.

Besides the feature-level and score-level fusion, some other fusions were applied in [2] which are concatenation, averaging and maximum. Under the feature-level fusion a set of newly generated features were obtained which contemplated the skeleton information and the depth which was extracted. So, here the concatenation fusion was used. Moreover, for the score-level fusion the scores obtained speak on behalf of the network's prediction. Hence, here the averaging and maximum fusion was applied. Amongst the three, the averaging one is expected to generate a more reliable result since it depends on more networks and hence more amount of information.

Chapter 3 Proposed Method

In [3] they have created the PointLSTM architecture that has achieved state of the art results in hand gesture recognition using point cloud sequences as their input modality. From [2] we can see that using multiple modalities of input to different models and later fusing the feature or score level predictions can give much better results than using unimodal input and features. So we propose an architecture that uses two modalities: depth image sequence and point cloud sequence. We used the state of the art model PointLSTM to work with the point cloud sequence and the DepthCRNN model to work with depth image sequence and we apply two types of fusion on the outputs of these two models: score level fusion and feature level fusion.

3.1 Score Level Fusion

Both the PointLSTM model and the DepthCRNN model are kept as the original ones and we feed point cloud sequences to the PointLSTM model and depth image sequences to the DepthCRNN model. Each model uses its respective MLP layers to give us a prediction score in each step. We apply the score fusion by merging these predicted scores using either max fusion or average fusion. In case of max fusion we take the maximum score between the two predictions for each class and in case of average fusion we take the average of the two scores predicted by the two models. In our experiment we see that average fusion performs much better than max fusion in accurately classifying hand gestures in our architecture.

3.2 Feature Level Fusion

In case of feature level fusion, we modify both the models slightly and discard the existing MLP layer of both models. Similar to score level fusion, we feed point cloud sequences to the PointLSTM model and depth image sequences to the DepthCRNN model but this time we take the features level scores only from both models instead of the class level prediction scores by removing the MLP layers. We concatenate the features found from the two models into a single feature vector and feed this to a single MLP layer which finally gives us the score level prediction that we use to classify the

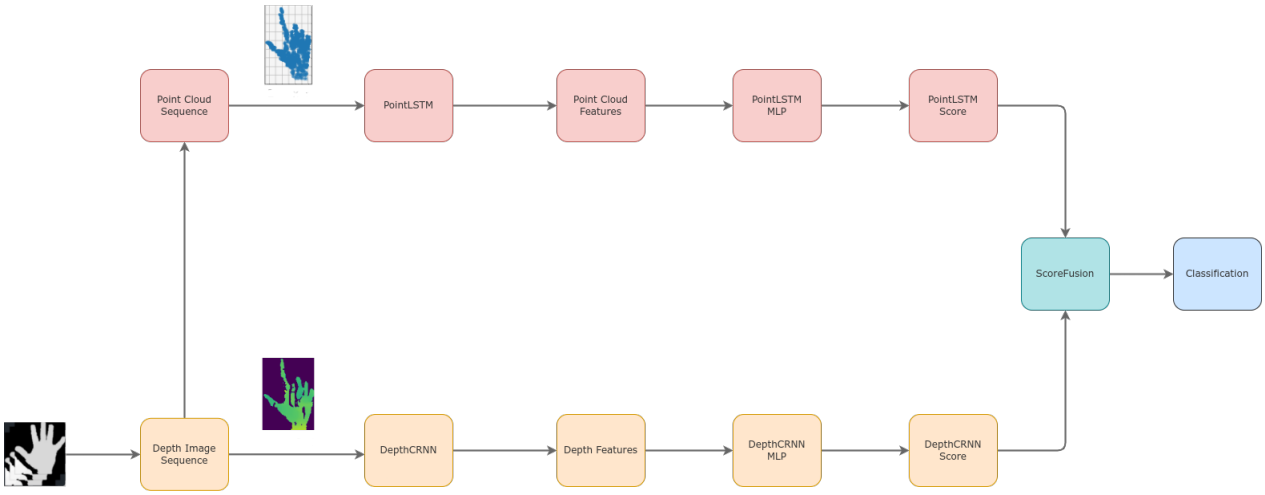

Figure 14: Score Fusion Architecture

gestures.

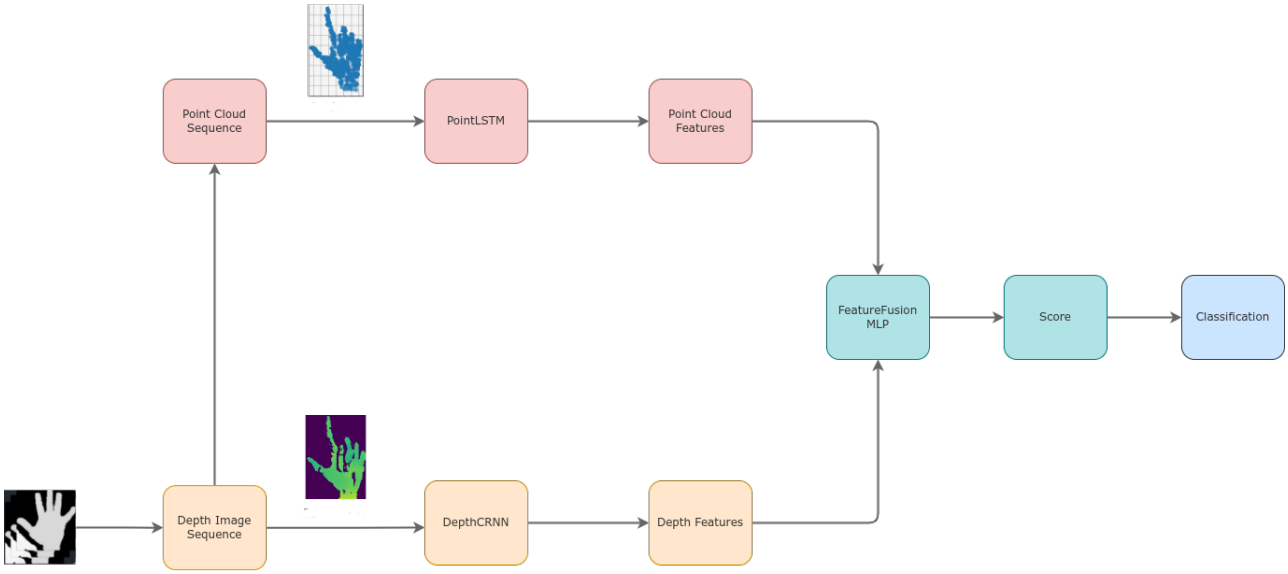

Figure 15: Feature Fusion Architecture

Chapter 4 Experimental Results & Discussion

4.1 Dataset

4.1.1 Dataset Description

We're using the SHREC2017 dataset [6] as our benchmark dataset. There are sequences of 14 gestures in the dataset which are performed in two ways: one finger and the whole hand. There are 2800 sequences resulted by performing each gesture between 1 and 10 times by 28 participants. These sequences are grouped by the gesture, participant, way of performing the gesture (one finger or whole hand). The sequence lengths vary from 20 to 50 frames and there is a depth image and the 3D coordinates of 22 joint points forming a hand skeleton in each frame. The depth images have a resolution of 640x480 and the sequences were captured in 30 frames per second.

Number of gestures	14
Number of participants	28
Total number of sequences	2800
Information in each frame	Depth image, Coordinates of 22 joint points
Dimension of each frame	640 x 480
Length of each sequence	20 to 50 frames

Figure 16: Dataset Information [6]

4.1.2 Dataset Preparation

The SHREC2017 dataset provides the coordinates required to segment out the hand region for every depth image. We used those provided coordinates to segment out the hand region of every frame of our depth image sequence and reshaped every frame to be a 50x50 image.

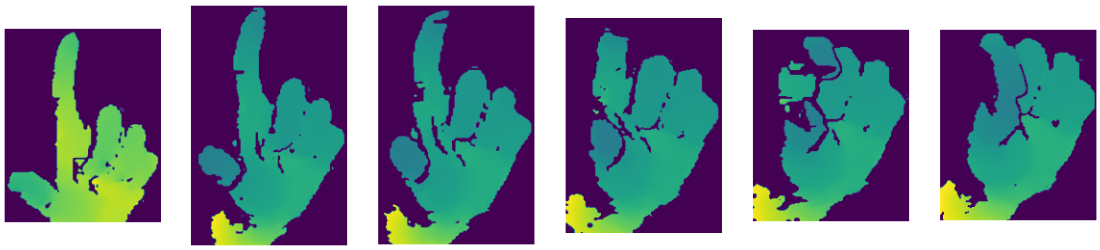

Figure 17: Depth Image Sequence

We used the depth image sequences from the SHREC2017 dataset to create the corresponding point cloud sequence for every sample gesture. In order to create point clouds from depth images we used the following formula:

$$x_{i,3D} = \frac{x_{i,2D} - c_x}{f_x} I_{(x_{i,2D}, y_{i,2D})} \tag{1}$$

$$y_{i,3D} = \frac{y_{i,2D} - c_y}{f_y} I_{(x_{i,2D}, y_{i,2D})} \tag{2}$$

$$z_{i,3D} = I_{(x_{i,2D}, y_{i,2D})} \tag{3}$$

Here $(x_i, 2D, y_i, 2D)$ defines a pixel of the depth image, $(x_i, 3D, y_i, 3D, z_i, 3D)$ defines the 3D coordinates of the point cloud for the corresponding pixel. (cx, cy) is the principal point offset and (fx, fy) is the focal length of the camera used to capture the depth images. These are intrinsic parameters of the camera.

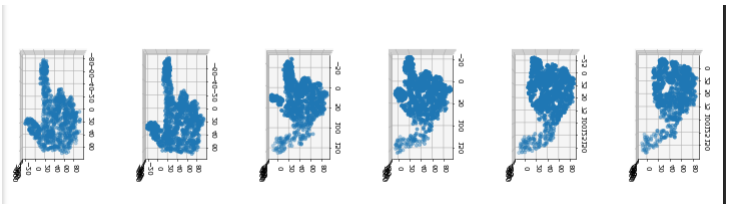

Figure 18: Point Cloud Sequence

We quantized the depth images in such a way that any depth value less than 200 is ignored (we took a threshold of 200 since anything behind this depth value is defined as the background) and the depth values above 200 are quantized to a range of 155-255 with 10 levels. We used the following formula for quantization as show in [12]:

$$Q(x,y) = DL_{min} + \left\lfloor \eta \times \frac{D(x,y) - D_{min}}{D_{max} - D_{min}} \right\rfloor \times \left\lfloor \frac{DL_{max} - DL_{min}}{\eta} \right\rfloor \tag{4}$$

Here $D(x,y)$ defines the depth value at the position (x,y) of the depth image and $Q(x,y)$ defines the corresponding quantized value. η is the number of levels being used (10 in our case). The range $DL_{min} - DL_{max}$ is the range of depth values that we are using $(155 - 255$ in our case).

Since there is not much emphasis on the actual hand region in depth images it can be seen that fingers and palm regions occupy similar depth values. This lack of contrast might hide significant information which can be helpful in gesture recognition. For example in the following figure it is not evident that fingers have overlapped with the palm in the original image but it can be seen in the quantized image. That is why we perform quantization on the depth images so that the contrast increase and our model can learn better using the extra information.

Figure 19: Original Image (left) Quantized Image (right) [13]

4.2 Training Details

The models were trained in kaggle using NVIDIA TESLA P100 GPUS. Each training took around 1 hour 34 minutes. The individual settings for each model stream can be seen as below :

1. PointLSTM

   32 frame clips were uniformly sampled and 256 points were generated for each frame. At training time, 128 points were randomly sampled from the preprocessed points and taken as input for the point lstm stream.The augmentations applied are randomly scaling(±20%), rotating(±15), and dropping input points(20%). [3]

2. DepthCRNN

   The framerate used is 32. The depth image sizes are 50 by 50. Initially no augmentation was used.

Table 1 shows the initial hyperparameters used in training.

Learning Rate	1e-3
Batch Size	4
Image Size	50*50
Epochs	30
Loss	Cross-Entropy
Optimizer	Adam
Scheduler Type	None

Table 1: Experimental Setup for Training

4.3 Initial Training of Average Score Fusion Model

The initial run was conducted by training the Score Fusion Model with initial hyperparameter settings and no augmentations on the depth images. The number of trainable parameters were 4078484, the optimizer used was Adam with no weight decay, the loss criterion was cross entropy loss and no schedulers were used.

From the initial run, we achieved an accuracy of 76% on the test dataset. The training curves can be seen as below :

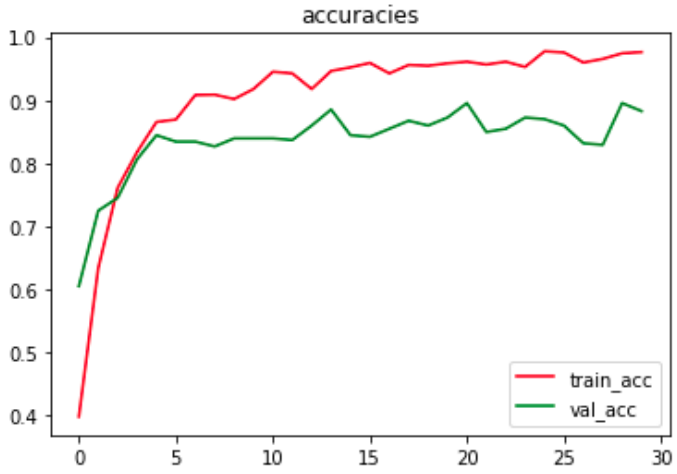

Figure 20: Initial Average Score Fusion Train Accuracy Curve

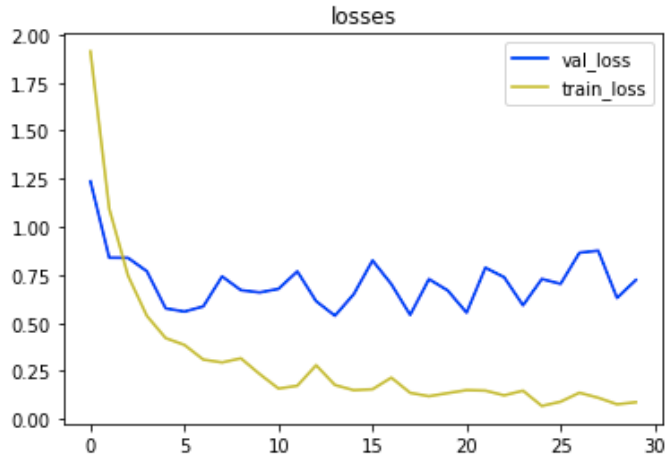

Figure 21: Initial Average Score Fusion Train Loss Curve

As we can see from the curves, significant overfitting exists which is why the test accuracy is so low. We need to perform augmentation on depth images and certain regularization techniques and hyperparameter tuning in order to reduce the overfitting.

4.4 Regularization techniques used to approach overfitting

Overfitting occurs when the model learns the training data very well but fails to generalize to new data. To overcome the problem of overfitting, different regularization techniques are used for our training.

4.4.1 Augmentations on depth images

Image augmentation is a technique used to artificially expand the existing dataset by slightly changing the original image through applying different geometric transformations. There are two types of augmentation, online and offline.[11] In offline augmentation, all the transformations on the images of the dataset are applied beforehand, thereby increasing the size of the dataset. This method is appropriate when the dataset is small as the size would increase in proportion to the number of transformations applied. In online augmentation[11], transformations are performed on mini batches before feeding it to the model. For our training, we use the albumentation library for online augmentation and apply the following geometric transformations to our depth images with a probability of 0.5 :

- Shift : The pixels of the image will be shifted horizontally or vertically by the desired factor without changing the dimensions of the image.
- Scale : The image will either be zoomed in or zoomed out depending on the value of the scale factor, a value less than one means zoom out and more than one means zoom in.
- Rotate : The image is rotated either clockwise or anti-clockwise depending on the rotation angle.

We tried different values of the augmentations for hyperparameter tuning. At first we trained using a shift and scale factors of 0.05 and a rotate factor of 10. From the training curves, we can see that overfitting has reduced drastically after applying the augmentations and we achieved a test accuracy of 85.83%.

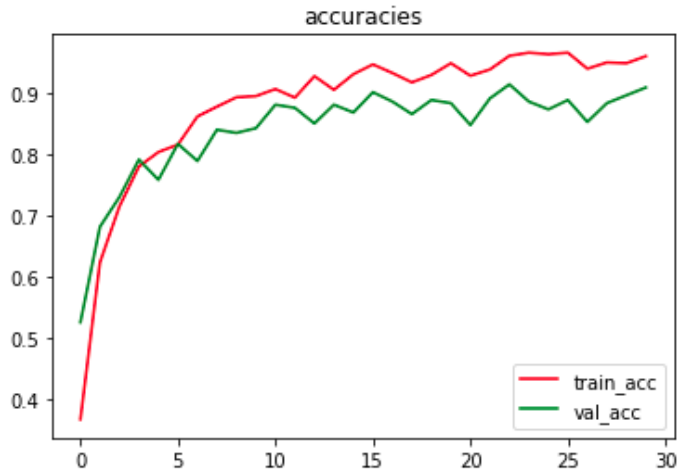

Figure 22: Average Score Fusion Train Accuracy Curve After Adding Depth Augmentation

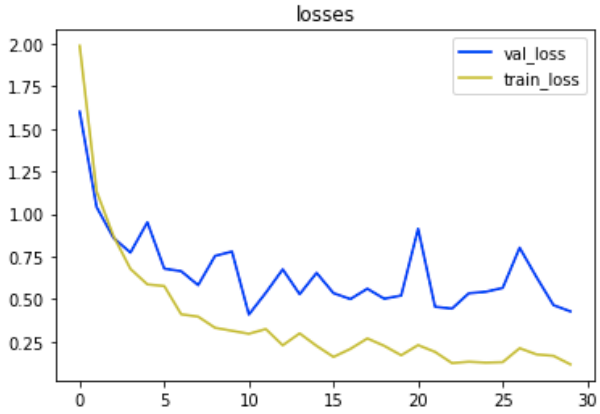

Figure 23: Average Score Fusion Train Loss Curve After Adding Depth Augmentation

The other factors of shift-scale-rotate we tried are 0.1-0.1-20 and 0.2-0.2-20 respectively. It was seen that the model performs the best with the settings 0.2 factor for shift and scale and 20 factor for rotation.

4.4.2 Time Shift Augmentation

We randomly crop 10% of the frames, i.e. 3 frames for our dataset as we are using a framerate of 32, either from the front or from the back. To keep consistency, after removing the frames, we have to pad the removed frames with zero so that the total number of frames remain constant.

4.4.3 Optimizer

A loss function lets us quantify how well the weights are at predicting the expected outcome. The goal of an optimization function is to find the set of weights that will minimize our loss function. The initial optimizer that we used was the adam optimizer. Adam optimizer uses a combination of RMSProp and Stochastic Gradient descent with momentum. Adam uses the estimations of the first and second moments of gradient to adapt the learning rate of each weight of the neural network. The first moment is the mean value while the second moment is the variance value. The advantage of Adam was to work well with sparse gradients and non-stationary objects [8] . However models trained with Adam optimizer do not always generalize well. Hence, a regularization technique, weight decay, is used with Adam optimizer so that models can generalize better. After changing the optimizer to AdamW with a weight decay of value 0.1, overfitting was slightly reduced and we achieved a test accuracy of 86.43%.
The training curves can be seen as below.

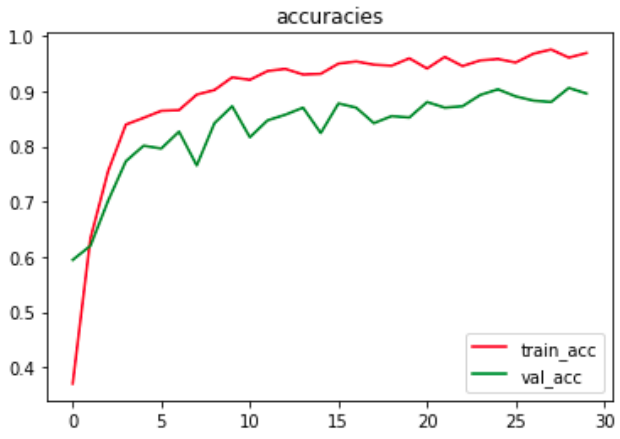

Figure 24: Average Score Fusion Train Accuracy Curve After Adding AdamW Optimizer

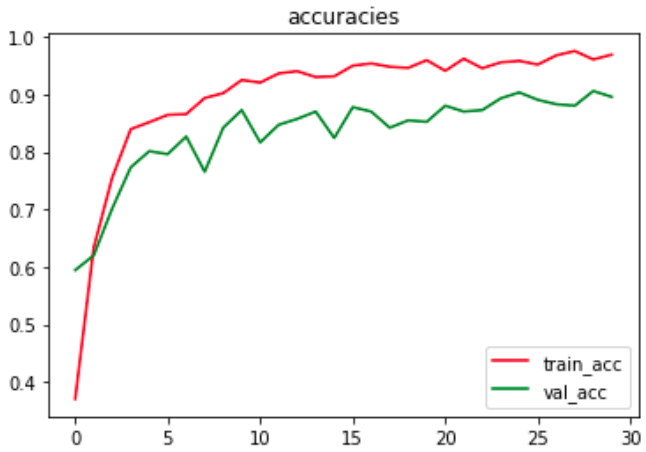

Figure 25: Average Score Fusion Train Loss Curve After Adding AdamW Optimizer

4.4.4 Loss Criterion : Label Smoothing

Label Smoothing is a regularization technique that can be used when the loss function is cross entropy and the model uses softmax function in its final layers to compute the logit[9] . When we perform classification, the labels are considered as hard, binary assignments and the softmax outputs one-hot encoded label vector y. The purpose of label smoothing is to turn the hard class label assignments to soft label assignments. Therefore instead of assigning a binary label to the classes with only one class having label 1 and the others label 0, the soft label assignment maximizes the probability of positive classes while assigning very low probability to other classes. This prevents the model from becoming too confident in its predictions, hence prevents overconfidence. After changing the loss criterion to label smoothing with a factor of 0.1, overfitting was drastically reduced, giving a test accuracy of 90%. The training curves can be seen as below.

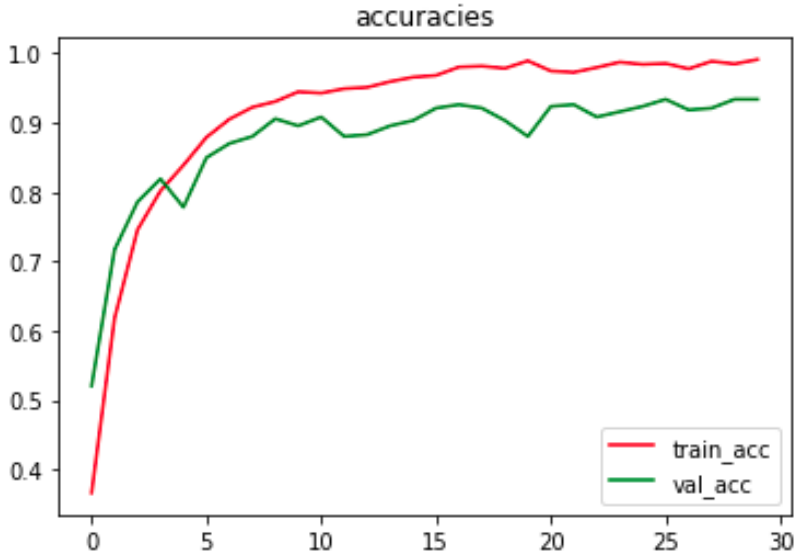

Figure 26: Average Score Fusion Train Accuracy Curve After Adding Label Smoothing of 0.1

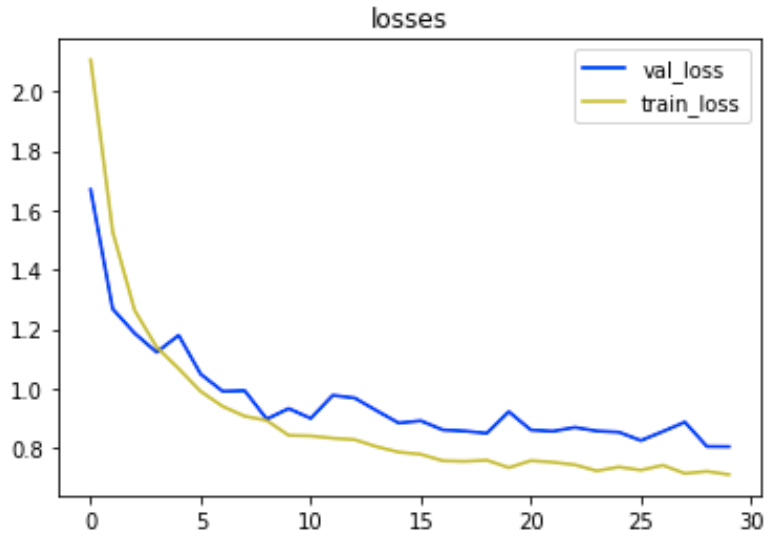

Figure 27: Average Score Fusion Train Loss Curve After Adding Label Smoothing of 0.1

4.4.5 Schedulers

All optimizers have a learning rate which allows them to define the size of gradient step,and this learning rate stays fixed over the training process. Learning rate schedulers modify the learning rate hyperparameter over time according to a scheduler instead of keeping it fixed.

We used WarmUp Learning rate in the first 10 epochs of training. The learning rate warmup uses a small step at the beginning of the training. The learning rate increases linearly or non linearly to a specific value at beginning and then shrinks to zero. The purpose of using warmup scheduling is that, at first, a model is initialized randomly and so it is far from being the ideal one, therefore using a large learning rate would make the training unstable. Using a small learning rate initially will enable us to apply larger learning rates later towards the training. [10]. So WarmUpLR increases the learning rate from minimum learning rate to maximum learning rate over the number of warm steps and then fixes it at maximum learning rate. As we can see in the image below, the first row shows the conventional annealing methods, which starts from a high value and either stays constant or decreases according to some function while the second row shows the warmup methods, which starts from zero and slow increases and then decreases again.

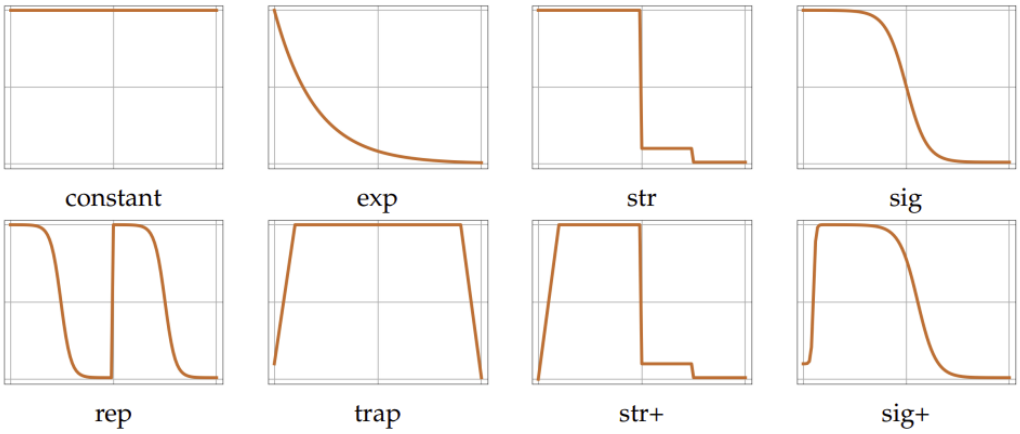

Figure 28: Different Annealing methods [10]

For the rest 20 epochs, a cosine annealing scheduler was used. Cosine AnnealingLR is a scheduling technique that starts with a very large learning rate which then aggressively decreases to zero before increasing again[10]. Mathematically, the cosine annealing can be formalized as follows [10] :

$$\eta_t = \eta_{min} + 1/2(\eta_{max} - \eta min)(1 + \cos(T_{cur}/T_{max} * \pi)) \tag{5}$$

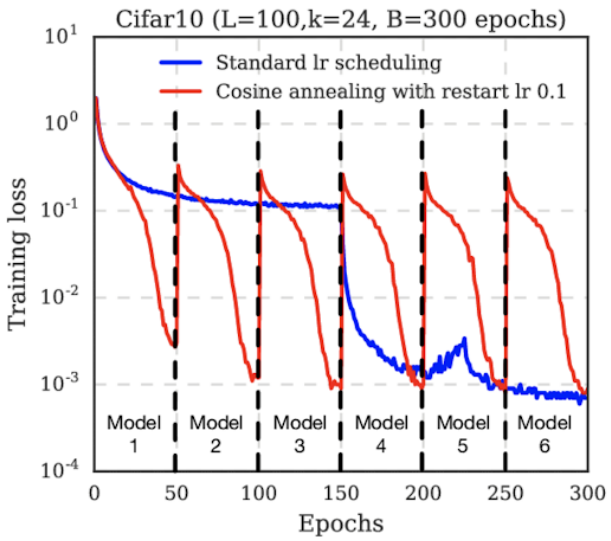

Figure 29: Cosine Annealing [10]

The purpose of using this scheduling technique is to use appropriate weights as the starting point for subsequent learning rate cycles, but it allows the learning algorithm to converge to another solution.

After adding the schedulers, overfitting was further reduced and the test accuracy rose to 90.83% as can be seen in the curves below.

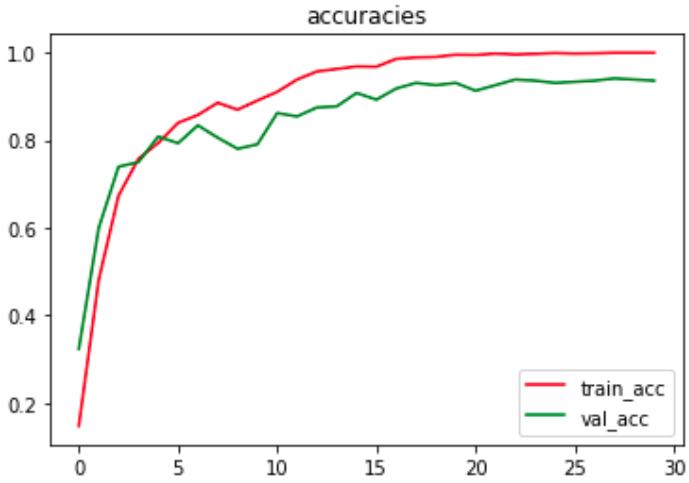

Figure 30: Average Score Fusion Train Accuracy Curve After Adding Schedulers

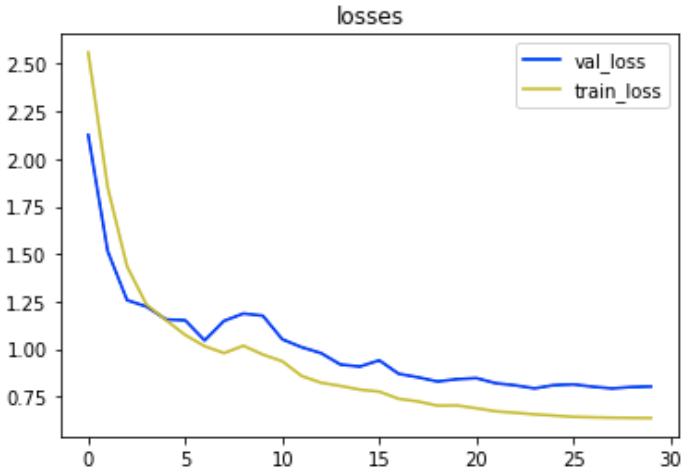

Figure 31: Average Score Fusion Train Loss Curve After Adding Schedulers

4.5 Score Fusion Train After All Regularization and Hyperparamter Tuning

After adding all the regularizations : depth image normalization and quantization, augmentation on depth images, adding AdamW as optimzer, adding Label Smoothing as loss criterion, adding two types of schedulers and tuning the dropout probability to 0.5, we run our final training of the score fusion model on 14 gestures and achieve an accuracy of **96.2%**. The training curves can be seen as below.

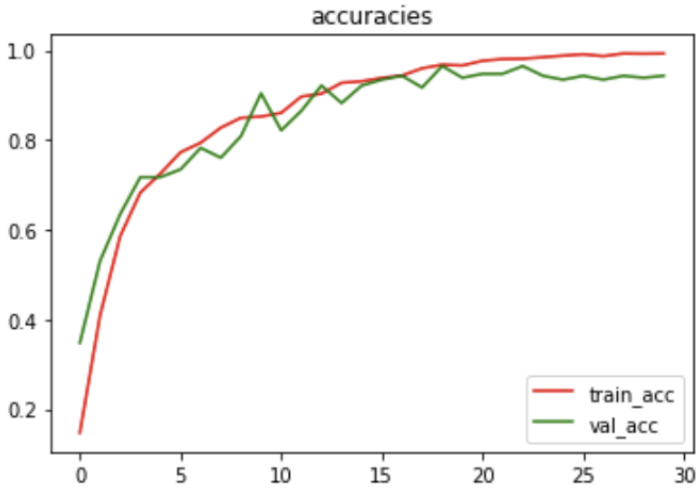

Figure 32: Score Fusion Train Accuracy Curve After Final training

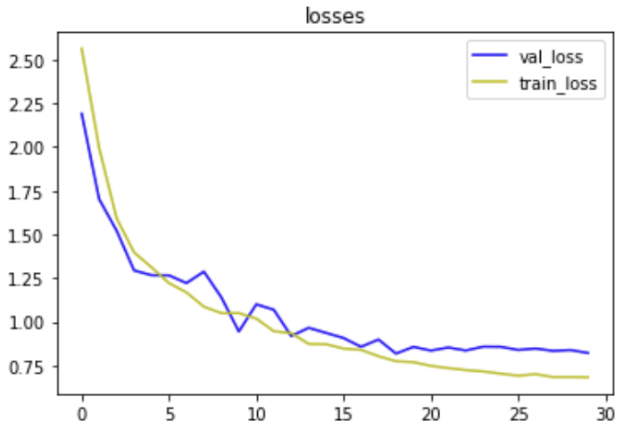

Figure 33: Score Fusion Train Loss Curve After Final training

4.6 Max Fusion for 14 gestures

Keeping all the hyperparameters for which we got the best result for average fusion, we train our model again, this time taking the maximum of the logits instead of the average of the logits. It does not seem to perform as well as average fusion. We acheive an accuracy of 90.6%. The training curves can be seen below.

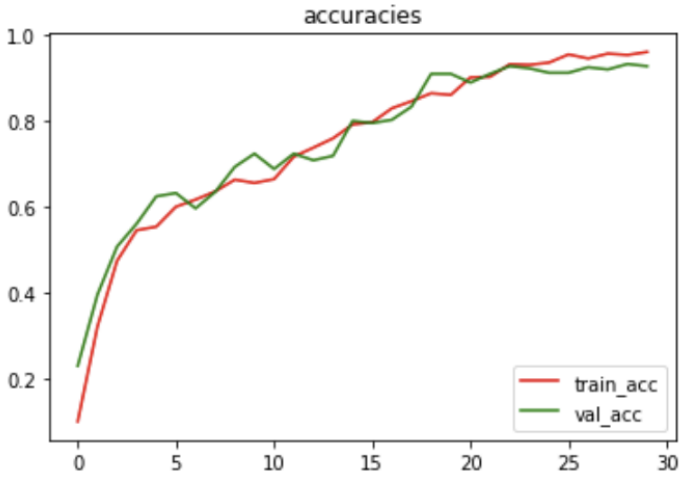

Figure 34: Max Score Fusion Train Accuracy Curve

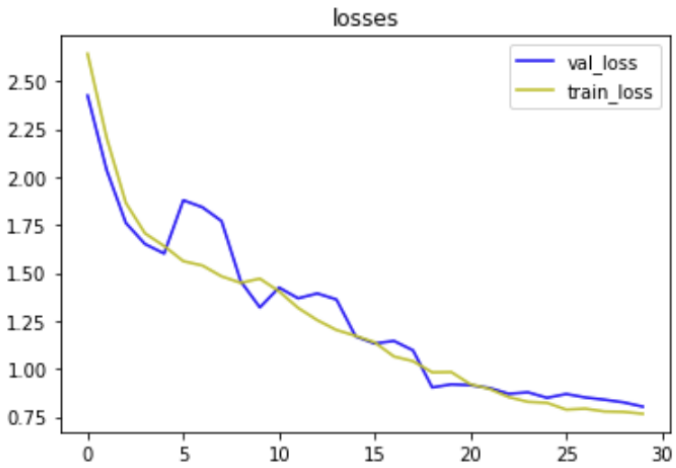

Figure 35: Max Fusion Train Loss Curve

4.7 Feature fusion for 14 gestures

Keeping all the hyperparameters for which we got the best result for average fusion, we train our model again, this time using feature fusion. We achieve a test accuracy of 94.6%. The score is very close to average fusion even though average fusion outperforms feature fusion. The training curves can be seen below.

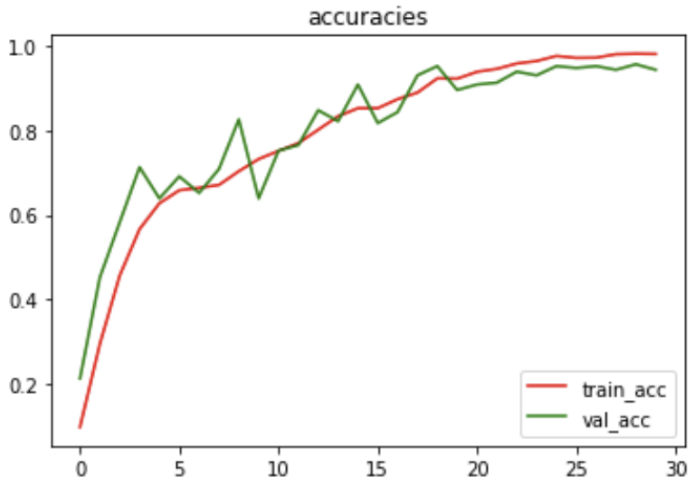

Figure 36: Feature Score Fusion Train Accuracy Curve

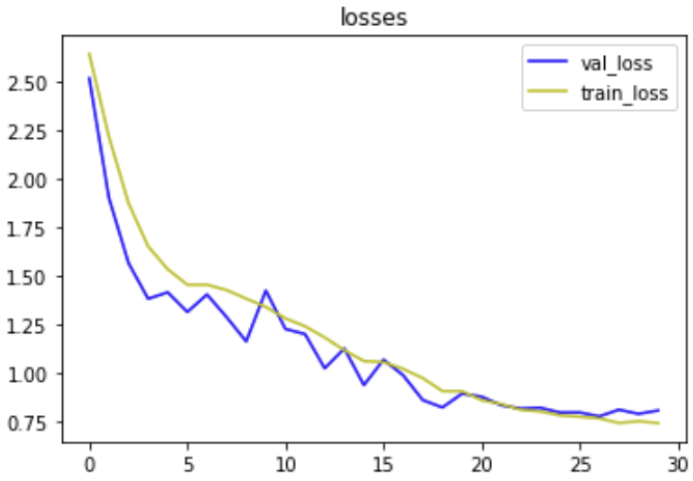

Figure 37: Feature Fusion Train Loss Curve

4.8 Comparison Analysis

The training results using different hyperparameters and fusion techniques is compared. The evaluation metric used is accuracy, which is the percentage of gestures correctly identified. The comparison of our best performing model is also compared against other models. The resutls can be seen in Table1, Table 3 and Table 4

4.8.1 Training Results Comparison

The test accuracy comparison across the different regularization techniques can be seen in the table below.

	Initial Run	**Augmentations on Depth Images**	**Optimizer AdamW**	**label smoothing loss**	**Schdulers**	**Time Shift Augmentation and Depth Quantization**
Average Fusion Model on 14 Gestures	76.31%	85.83%	86.43%	90%	90.83%	96.2%

Table 2: Accuracy Comparison Table Across Different Regularization techniques

The comparison of the three types of models we trained, Avg-Score-Fusion, Max-Score-Fusion and Feature-Fusion on 14 and 28 gestures can be seen as below. It can be concluded that Avg-Score-Fusion performs the best while Max-Score-Fusion performs the worst. It is also to be noted that feature fusion model works better for 28 gestures than avg-score-fusion according to our experimental results.

	Average Score Fusion	**max Score Fusion**	**Feature Fusion**
14 Gestures	96.2%	90.6%	94.6%
28 Gestures	92.8%	87.1%	93.8%

Table 3: Accuracy Comparison Table Across Different Fusion Models

4.8.2 Performance Comparison with other models

The comparison of our model results to other models can be seen as follows. We can see that our avg-score-fusion model outperforms the state-of-the-art model, PointLSTM for 14 gestures.

Method	**Modality**	**14**	**28**
Key frames	depth sequence	82.9%	71.9%
SoCJ+HoHD+HoWR	skeleton	88.2%	81.9%
Res-TCN	skeleton	91.1%	87.3%
STA-Res-TCN	skeleton	93.6%	90.7%
ST-GCN	skeleton	92.7%	87.7%
DG-STA	skeleton	94.4%	90.7%
Point LSTM	point clouds	95.9%	94.7%
Average Score Fusion	point clouds and depth sequence	**96.2%**	92.8%
Max Score Fusion	point clouds and depth sequence	90.6%	87.1%
Feature Fusion	point clouds and depth sequence	94.6%	93.8%

Table 4: Performance Comparison on SHREC'17 Dataset [3]

Chapter 5 Conclusion and Future Work

Our work is primarily focused on studying the multi-modal fusion approach to dynamic hand gesture recognition. We focus on a few points :

1. We explore the different regularization techniques and hyper-parameter tuning in order to reduce the effect of overfitting, enabling the model to learn better.
2. We explore different fusion techniques to enable the model to learn from the different modalities of input for hand gestures.
3. We try to improve efficiency by achieving similar results using smaller size depth images, smaller size point clouds and training the model for a smaller number of steps.
4. We have shown that adding grayscale variations improves the overall accuracy of the multimodal fusion architecture.

Some challenges we faced are :

1. Exploring ways to reduce overfitting
2. Randomised results due to the random seed values.

Our list of future work can be listed as follows :

1. Explore more regularization techniques so that overfitting can be further reduced.
2. Add more layers to the final MLP in feature fusion and explore the impacts.
3. Tune hyper parameters for the feature fusion model
4. Try a different modality of data
5. Train the other benchmark hand gesture recognition datasets using score fusion and feature fusion models.

References

[1] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "PointNet: Deep Learning on point sets for 3D classification and segmentation," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[2] K. Lai and S. N. Yanushkevich, "CNN+RNN depth and skeleton based dynamic hand gesture recognition," arXiv [cs.CV], 2020.

[3] Y. Min, Y. Zhang, X. Chai, and X. Chen, "An efficient PointLSTM for point clouds based gesture recognition," in 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[4] M. Oudah, A. Al-Naji, and J. Chahl, "Hand gesture recognition based on computer vision: A review of techniques," J. Imaging, vol. 6, no. 8, p. 73, 2020.

[5] M. Yasen and S. Jusoh, "A systematic review on hand gesture recognition techniques, challenges and applications," PeerJ Comput. Sci., vol. 5, p. e218, 2019.

[6] "DHG - 14/28," Telecom-lille.fr. [Online]. Available: http://wwwrech.telecom-lille.fr/DHGdataset/. [Accessed: 23-Nov-2021].

[7] Lu, Qiang Xiao, Mingjie Lu, Yiyang Yuan, Xiaohui Yu, Ye. (2019). Attention-Based Dense Point Cloud Reconstruction From a Single Image. IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2943235.

[8] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).

[9] Müller, Rafael, Simon Kornblith, and Geoffrey E. Hinton. "When does label smoothing help?." Advances in neural information processing systems 32 (2019). 10. Nakamura, Kensuke, et al. "Learning-Rate Annealing Methods for Deep Neural Networks." Electronics 10.16 (2021): 2029.

[10] Loshchilov, Ilya, and Frank Hutter. "Sgdr: Stochastic gradient descent with warm restarts." arXiv preprint arXiv:1608.03983 (2016).

[11] Gandhi, A. (2021, May 20). Data augmentation: How to use deep learning when you have limited data. AI amp; Machine Learning Blog. Retrieved

April 24, 2022, from https://nanonets.com/blog/data-augmentation-how-to-use-deep-learning-when-you-have-limited-data-part-2/

[12] Mahmud, H., Hasan, M., Kabir, M., Mottalib, M.A.: Recognition of symbolic gestures using depth information. Adv. Hum. Comput. Interact. (2018).

[13] Mahmud, Hasan Morshed, Mashrur Hasan, Md Kamrul. (2021). "A deep-learning–based multimodal depth-aware dynamic hand gesture recognition system".

[14] Islam, Robiul Mahmud, Hasan Hasan, Md Kamrul Rubaiyeat, Husne. (2016). Alphabet Recognition in Air Writing Using Depth Information.

[15] Amma, Christoph, Dirk Gehrig, and Tanja Schultz. "Airwriting recognition using wearable motion sensors." Proceedings of the 1st Augmented Human international Conference. 2010.

[16] Xie, Lei Wang, Chuyu Bu, Yanling Sun, Jianqiang Cai, Qingliang Wu, Jie Lu, Sanglu. (2018). TaggedAR: An RFID-based Approach for Recognition of Multiple Tagged Objects in Augmented Reality Systems. IEEE Transactions on Mobile Computing. PP. 1-1. 10.1109/TMC.2018.2857812.

[17] Zhou, B., Wan, J., Liang, Y., Guo, G. (2021). Adaptive cross-fusion learning for multi-modal gesture recognition. Virtual Reality Intelligent Hardware, 3(3), 235-247.

[18] Špakov, Oleg, Howell Istance, Kari-Jouko Räihä, Tiia Viitanen, and Harri Siirtola. "Eye gaze and head gaze in collaborative games." In Proceedings of the 11th ACM Symposium on Eye Tracking Research Applications, pp. 1-9. 2019.

[19] Mahmud, Hasan Islam, Robiul Hasan, Md Kamrul. (2022). On-air English Capital Alphabet (ECA) recognition using depth information. The Visual Computer. 38. 10.1007/s00371-021-02065-x.

[20] Alam, Md, Ki-Chul Kwon, Mohammed Y. Abbass, Shariar Md Imtiaz, and Nam Kim. "Trajectory-based air-writing recognition using deep neural network and depth sensor." Sensors 20, no. 2 (2020): 376.

[21] Qi, Charles Ruizhongtai, Li Yi, Hao Su, and Leonidas J. Guibas. "Pointnet++: Deep hierarchical feature learning on point sets in a metric space." Advances in neural information processing systems 30 (2017).