

Dynamic BDSL Digit Recognition Using Hybrid 3DCNN+BiLSTM Model

Ishat Salsabil Silvia¹, Kamrul Hasan Talukder²

¹MSc. Student, Computer Science and Engineering Discipline, Khulna University

²Professor and Former Head, Computer Science and Engineering Discipline, Khulna University

Abstract

Sign language is used as a way of communication by deaf and mute individuals. However, due to the limited number of people who understand sign language, integrating them into society is challenging. Approximately 6.9% of Bangladesh's population and 5% of the world's population suffer from speech impediments. Individuals with this condition cannot hear what others are saying or communicate verbally, thus sign language must be relied upon. In recent years, sign language recognition has gained attention due to its necessity. However, a scarcity of publicly available dynamic gesture datasets for Bangladeshi Sign Language (BdSL) exists. Dynamic gestures, which contain both spatial and temporal information, are more useful in real-life applications. The classification of dynamic gestures requires more data than static gestures. In this research, 11 numeral Bengali digit signs are aimed to be classified. Data has been collected from the "SignBD-Word" dataset, which contains extracted RGB human body pose keypoints skeleton data. When compared to other dynamic sign language datasets and data requirements for training deep neural networks for dynamic gestures, the data per class in this dataset is found to be insufficient. A hybrid model architecture is proposed in this research to recognize dynamic gestures using lightweight 3DCNN and bidirectional LSTM layers for classifying gestures from the motion patterns of human body pose keypoints skeleton data after experimenting on various models. It has been observed that combining 3DCNN with the pre-trained DenseNet-201 and the BiLSTM model increases real-time accuracy by 4.54%. To the best of our knowledge, this is the first approach to combine 3DCNN with DenseNet-201 for action recognition. Also, one of the earliest investigations on dynamic BdSL hand gesture digit recognition using dynamic data. Additionally, different pre-trained models as base feature extractors have been evaluated.

Keywords: BdSL, Sign language, Deep learning, Transfer learning, Body joints skeleton, Action recognition

1. Introduction

Hand gestures are a nonverbal technique to communicate emotion, emphasize and structure a conversation, and point to persons and objects. Sign language is a gesture-based language used mostly by disabled individuals to communicate more successfully with one another and with normal people. Despite the ambiguity of hand gesturing meanings, which largely depend on the current working situation and geographical and cultural background, some army, navy, and air force motions can express precise information, allowing Human-Robot Interaction (HRI) via hand gestures. So, adopting a more reliable and precise system for identifying HGR's necessity cannot be ignored. Bangla Sign Language (BdSL) is the

sign language used in Bangladesh. The units of the numeric system, called Bengali numerals, come from the Indian subcontinent and are extensively used in many languages. More than 350 million people (or over 5% of the world’s population) use them [13]. Offline sign language detection methods normally use image or video datasets. However, very few works have been done to recognize Bengali sign language from video datasets due to the scarcity of video datasets for BdSL. Images record one still instant; they are unable to represent the important temporal dynamics that are present in gestures. Videos provide the model with a constant stream of frames from which it may extract motion features such as direction changes, acceleration, and velocity. For dynamic motion gesture recognition (DMGR) tasks, a deep learning model trained on video data performs better than one trained on static images [2]. Sign languages for Bangla base numerals are ০, ১, ২, ৩, ৪, ৫, ৬, ৭, ৮, ৯ which are described in below,

Figure 1: Bengali Base Numeral Hand Gestures.

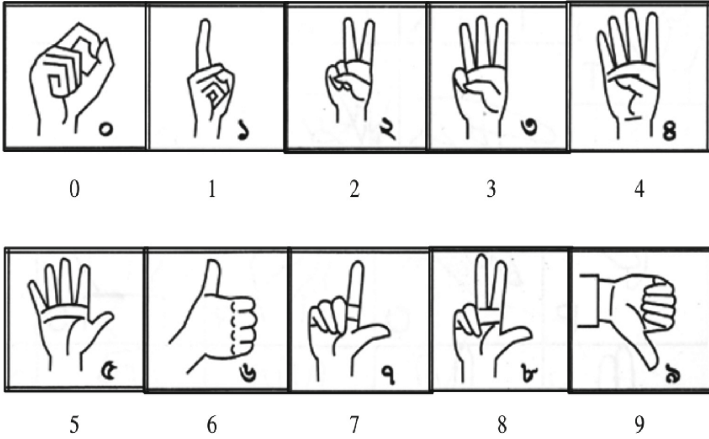


Table 1: Class Labels with Corresponding Bengali And English Translations.

Class labels	Bengali	English
shunno	০	0
ek	১	1
dui	২	2
tin	৩	3
char	৪	4
pach	৫	5
choy	৬	6
saat	৭	7
aat	৮	8
noy	৯	9
dos	১০	10

In this research, 11 numeral BdSL signs were classified including base numeral along with number 10. Since video-based data has a second (temporal) dimension, classifying it is more difficult than images. However, some CNN video domain extensions have been investigated. Several methods are used to recognize human gestures to date. The two-stream CNN [31], two-stream 3D CNN [32], 3DCNN+ConvLSTM [33], 3D CNN [13], and 3D CNN + LSTM [34] are among the technologies that are frequently used for action recognition. Deep learning has thus been successfully applied in many recent projects. Recognizing gestures from fewer sample videos is a challenging task because traditional 3DCNN requires more data than 2DCNN to classify. Human body joints store incredibly accurate information regarding human positions, making them a considerably more convenient and efficient way to describe activities and, consequently, how they are carried out [8]. Also, Human skeletons are easily obtained using sensors or pose recognition tools, and they are robust to changes in background and illumination [6]. This research intended to classify the motion of the human body keypoint joints skeleton containing frames extracted from the human body which requires less spatial features than human poses. For this research, a hybrid 3DCNN+BiLSTM structure with a base pre-trained model Densenet-201 as a feature extractor is proposed for classifying body joint skeleton motion patterns. Also, a few augmentation techniques considering real-

life pose variations and data variations on the dataset were applied and used different pre-trained models as feature extractors. Out of them, DenseNet-201[1] outperformed other pre-trained models.

The following is a summary of this paper's key contributions:

1. One of the earliest investigations on dynamic BdSL hand gesture digit recognition using motion pattern of performed dynamic gesture.
2. A comparative study on the effectiveness of using different pre-trained models as base feature extractors.
3. Proposed a hybrid 3DCNN+ BiLSTM model using DenseNet-201 as a base feature extractor after evaluating the efficiency of using pre-trained models combined with 3DCNN to assess model performance.
4. This study pioneers the implementation of DenseNet-201 with 3DCNN and BiLSTM for enhanced action recognition, presenting a novel approach not extensively explored in existing literature.

2. Related Works

Guo, Z. et al. (2022) [3] suggested a method for capturing detailed body information using Whole-Body Keypoint and Skeleton (WKS) labels. They developed an architecture that achieves superior performance by utilizing the Swin transformer when combined with three-dimensional (3D) convolutional neural networks (CNNs) to extract spatiotemporal characteristics. Using the UCF-101 dataset, they evaluated their performance. Hachiuma, R. et al. (2023) suggested a method [4] in which a Structured Keypoint Pooling network was proposed. The suggested approach offers resilience against input errors by sparsely aggregating keypoint features in a cascaded fashion based on previous knowledge of the data structure, such as the instances and frames to which each keypoint belongs. They evaluated their method by comparing RGB input frames skeleton, skeleton+object. Both skeleton and skeleton+object for classifying different actions outperformed the RGB input frames for different action recognition.

Altaf, Y. et al. (2023) proposed an approach for Indian Sign Language Recognition [9] while they used DenseNet-201 with Transfer Learning and gained very high accuracy for image-based classification. Al-Hammadi et al. (2020) [5] stated an approach utilizing 3DCNN for learning region-based spatiotemporal features of hand gestures. Two different approaches, a single 3DCNN structure, and a fusion of parallel 3DCNN structures were used for both signer-dependent and signer-independent modes. Sharma, S. et al. (2021) [22] proposed a system for recognizing American sign language (ASL) from motion gesture videos using video data as input. They used 3DCNN for processing video data. In their system, Frames were extracted from RGB videos and converted to grayscale, and different preprocessing steps like noise and spot reduction were done. The frames were recombined later for passing to cascaded 3DCNN. The system is implemented for recognizing 100s of different ASL words. The first method to use 3DCNN for keypoints skeleton-based classification was proposed by Liu, H. et al. (2017) [32]. They proposed a two-stream 3DCNN method for human skeleton-based action recognition.

Several static BdSL datasets are available publicly like BdSL-D1500[26], Ishara Lipi [27], KU-BdSL [28], Ishara-Bochon [30], etc. But to the best of our knowledge, only one dynamic BdSL hand gesture has been published publicly which is SignBD-Word [10]. Various image-based BdSL digit and alphabet classification works have been done till now. Among them, Islam, S. et al. (2018) [20] proposed a CNN model to recognize BdSL base numeral digits and trained the model on Ishara Lipi [27]. Another research was conducted on recognizing BdSL digits by Rayeed, S. M. et al. (2022) [29], where they presented a

dataset for the base BdSI numerals and used an SVM model for depth information containing image class classification purposes.

3. Baseline Models Overview

This research evaluated the performance of several models, including both pre-trained and custom-built architectures or layers. Pretrained models like I3D and DenseNet (DenseNet-201, DenseNet-169) were analyzed for their efficiency and accuracy. Additionally, models developed from scratch, such as 3DCNN, LSTM, and BiLSTM were also reviewed. The study provides a comprehensive comparison and insight into how each model performs under different conditions.

3.1 3DCNN

A 3DCNN [13] proposed by Tran, D. et al. is similar to a 2DCNN except for a few things. 2D convolutions produce a single image by using the same weights over the entire depth of the stack of frames (many channels). To preserve the temporal information of the frame stack, 3D convolutions employ 3D filters and result in a 3D volume. The input layer of 3DCNN takes sequences of frames as input where the images have width and height like 2DCNN and another spatiotemporal feature like depth or time. The hidden layers consist of 3D convolutional operations and pooling layers. Several typical hyperparameters for defining convolutional operations include stride, padding, filter size, and 3DCNN layers, which enhance the recognition of moving and 3D images. A three-dimensional filter that moves in three directions is present in each layer (x, y, z). A convolutional map is produced during the 3D convolution process, which is required for data analysis as well as time and volumetric context.

3.2 DenseNet

DenseNet-201, a variant of the Dense Convolutional Network (DenseNet) architecture, is known for its efficiency in feature extraction through the dense connectivity pattern among its layers [1]. Each layer in DenseNet-201 receives additional inputs from all preceding layers and passes on its feature maps to all subsequent layers, which facilitates a highly efficient flow of information throughout the network. This architecture proves particularly beneficial when working with small datasets, as it inherently includes regularizing characteristics that help reduce overfitting—a common challenge in training deep learning models on limited data. Moreover, the network's ability to leverage deep supervision ensures that even the earliest layers contribute directly to the final output, enhancing the model's sensitivity to fine details in dynamic motion patterns. DenseNet-169 which is another variant of DenseNet, adopts a less complex architecture. This depth reduction in comparison to the 201 layers of DenseNet-201 results in a trade-off between processing efficiency and accuracy.

3.3 LSTM

For handling sequential data, long short-term memory (LSTM) networks proposed by Hochreiter, S. et al. have become a potent tool for dynamic motion pattern recognition [14]. In contrast to conventional feedforward neural networks, long-range relationships within motion sequences can be learned by LSTMs thanks to an internal memory structure. This is important for applications like video action identification, where correct classification depends on a grasp of the temporal relationships between frames. Compared to models that just use spatial variables, LSTMs perform better because they can capture the finer details of human motion, such as the rhythm of stride or the flow of gestures [17].

3.4 BiLSTM

Bidirectional LSTMs (BiLSTMs) provide an even more sophisticated method, building on the success of LSTMs in dynamic motion pattern recognition [18]. Standard LSTMs process input exclusively in one

direction, even though they are quite good at identifying temporal connections within a sequence. To overcome this, BiLSTMs combine two LSTMs, one of which processes the sequence ahead and the other backward. This enables the model to simultaneously utilize data from previous and subsequent frames in the sequence [19]. This is especially helpful for applications like complicated action recognition, where proper categorization depends on the context of the frames that come before and after. Research has demonstrated that in dynamic motion pattern recognition tasks, BiLSTMs can outperform normal LSTMs, resulting in enhanced accuracy and robustness in action recognition applications [24].

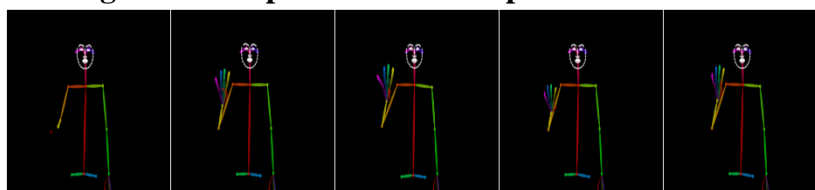
3.5 I3D

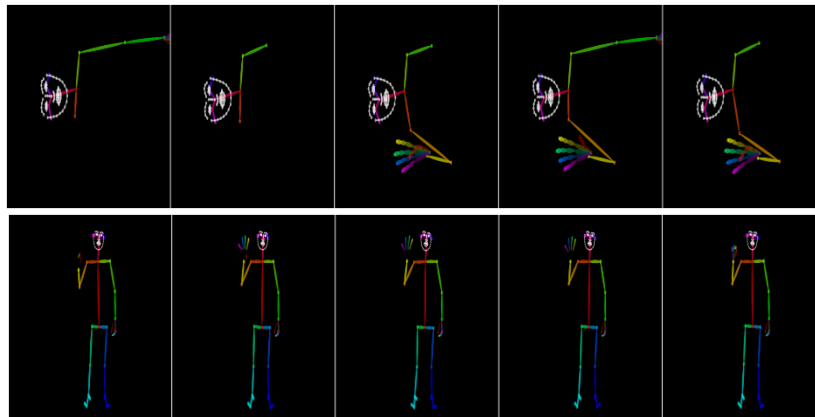
The I3D (Inflated 3D ConvNet) model is a pivotal development in action recognition, originally introduced by Carreira, J. et al. [25] in 2017. It adapts 2D ConvNet architectures into 3D, enabling simultaneous capture of spatial and temporal features in video data which significantly improves action recognition accuracy. Subsequent enhancements, including integration with attention mechanisms and graph-based techniques, have further boosted its performance, demonstrating its effectiveness and adaptability in the field of video analysis. Recent studies have also explored hybrid approaches, combining I3D with other deep learning models to further enhance its ability to handle complex motion patterns.

4. Dataset Description

Although there are video datasets for widely used sign languages of different languages, currently, we are aware of just one dataset for word-level BdSL. The data is collected from the publicly available SignBD-Word [10] dataset which consists of 6000 sign videos representing 200 unique words. The dataset contains 2D body pose data in addition to full and upper body shots of the signers. 16 participants performed the gestures. Of them, thirteen are men and the remaining individuals are women, ages 18 to 27. Signers were positioned 5–6 feet away. The subjects are filmed twice: once in an upper-body view and again in a full-body shot. Every sign class that represents a word has thirty different video clips. 11-digit gesture classes from the dataset were collected. The collection contains 2D body keypoint joint skeleton information containing frames extracted from each video, including key points for the hands, face, and body. To efficiently obtain particular key points for every frame of the upper-body and full-body image, OpenPose [7] along with Pixie2HD was employed for extracting 2D body keypoint joints from each video frame. Among the 16 performer's gesture performances, 2 signers' gesture performances contain only one view which is mostly left rotated upper body view. Each of the gesture classes was split into two sets which are train and test. The dataset contains 24 data for training in the train folder and 6 data per test folder for each class. And for class 'dui' there are 23 train data. Thus, training a deep neural network requires huge data comparatively to perform better in real life. In this case, several approaches were applied to classify the gestures with this limited amount of data and tried to increase the model's performance. Here are some sample frames from the dataset of class "char",

Figure 2: Sample Frames Example of The Dataset.





5. Pre-Processing

5.1 Pre-Processing Frames

Training a neural network generally requires a large amount of data. The aim is to classify gestures from a limited amount of data with various diversity. To minimize complex issues like trimming less important frames, applying data augmentation for increasing amounts of data and variation in this dataset, etc. for simplicity purposes were applied. For this purpose, the frames were resized from (512, 512, 3) to (224, 224, 3) which is a standard size and applied largely to various video-based datasets classification. Also, the normalization of pixels is applied to reduce complexity.

In the original SignBD dataset, from each video, 30 frames were extracted, while the 30 frames included starting off performing the gestures till the end. After evaluating the whole dataset, it is seen that the starting or ending few frames contain mostly neutral or repeated poses. So, 25 frames out of 30 were selected truncating the first 3 and last 2 frames so that the sequences contain important features.

5.2 Cross-Validation

Within digit data in the dataset, each of the gesture classes contains 24 folders of frames except one gesture class 'dui', which contains 23 folders. The validation dataset isn't available separately. So, a cross-validation technique was used along with data augmentation to increase the capability of model learning. For this purpose, the training dataset was split with a split rate of 0.75 for the train set and the rest for the validation set. Each time all folders per class randomly shuffle and create different batches for training and validation datasets.

5.3 Sequence Generation

Generating sequences of frames includes stacking sorted frames. For training a CNN model, the shape of each data has to remain the same. While stacking the frames, 25 frames were selected as described above. Then normalized each image for efficient data loading and generating sequences, tf.data libraries offered by TensorFlow for pipeline data generation were used, and the prefetching technique for faster data generation and feeding was also used. While generating batches of sequences, label mapping was assigned corresponding to each sequence. Using the data pipelining technique leverages the use of memory efficiency by loading batches of data in memory rather than loading the whole data and allowing the data pipeline to fetch batches of data in the background, thus speeding up training.

5.4 Augmentation

In this study on deep learning for dynamic sequence recognition, data augmentation is crucial for enhancing model robustness and generalization. A consistent augmentation strategy across video sequences, involving central cropping, zoom, and controlled rotation was applied. Specifically, each video

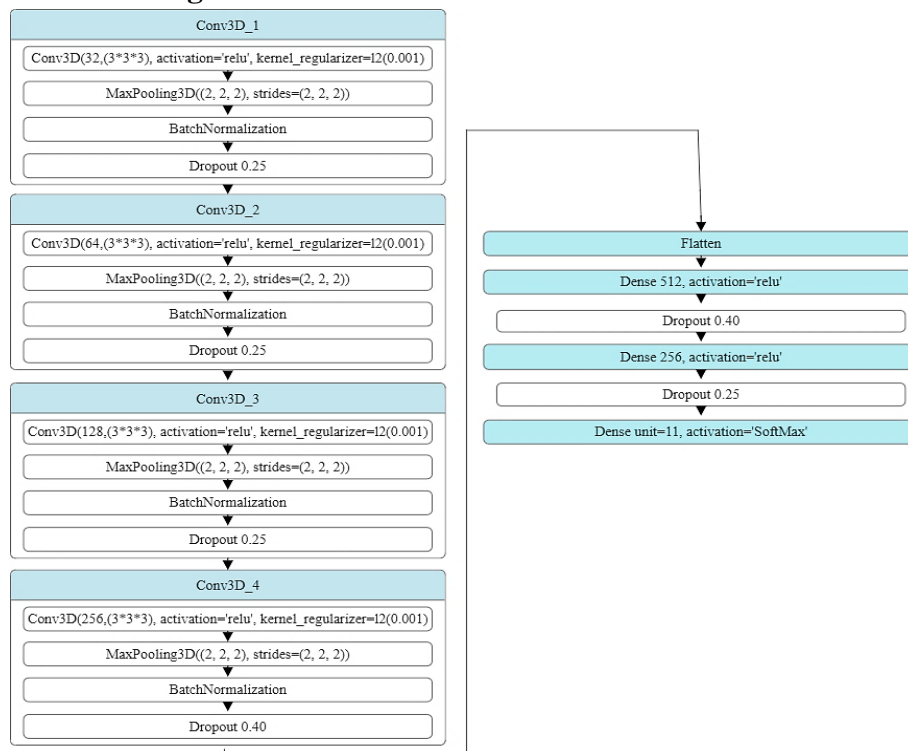
frame sequence is centrally cropped to retain 85% to 95% of the area, simulating partial object visibility. Frames are also zoomed between 1.1x and 1.3x to emulate varying camera-subject distances. Additionally, frames undergo rotations of 3 to 5 degrees to mimic real-world object orientations. This systematic application of transformations ensures temporal consistency across frames, critical for maintaining the continuity essential for learning from video data.

6. Proposed Model Evaluation

6.1 Experimental 3DCNN Model

As there are no pre-trained models available for similar datasets like ours, initially a lightweight 3DCNN model was built where 3DCNN layers were inspired by a sign language feature extractor embedding 3DCNN model proposed by Jirathampradub, H. et al. [15] and added dense layers the upon performance of recognition, modified the model accordingly. Initially, the model performed well in recognizing the training dataset but recognition accuracy for the test dataset was poor. The initial model structure was specifically designed for video data, incorporating multiple 3D convolutional layers to capture both spatial and temporal features effectively. The input shape here is (25, 224, 224, 3) which represents (the number of frames per sequence, image height, image width, and color channel).

Figure 3: Architecture of the 3D CNN Model



The network begins with a 3D convolutional layer with 32 filters of size $3 \times 3 \times 3$, including regularization via an L2 regularization to mitigate overfitting and applying batch normalization and dropout for better generalization. The model then escalates the complexity by progressively doubling the number of filters in subsequent layers—64, 128, and finally 256—each followed by max pooling for dimensionality reduction and additional batch normalization and dropout layers of value 0.25 after conv3D layers 32, 64, and 128. A dropout value of 0.40 after conv3D layer 256 was added to enhance training stability and model robustness and prevent overfitting. The top of the network flattens the output from the 3D

convolutional layers and passes it through a dense layer with 256 units, culminating in a SoftMax output layer that categorizes the input into multiple classes based on the training data.

This architecture is particularly suited for tasks requiring analysis of sequential video frames, such as action recognition or gesture detection, where capturing the dynamics within the video is crucial. Though considering the number of training data, the model was able to classify all the training data very accurately, and upon training more epochs it increased more training accuracy without dropping test accuracy values. At a certain point, it could predict all the training datasets. But test accuracy was 16%. So, different augmentation techniques described above were implemented and increased variation which helped in gaining test accuracy of 19.7%. However, the training accuracy was high so the model was capable of distinguishing different gestures accurately without decreasing the test accuracy. Then various available pre-trained models as base feature extractors were applied to improve performance and then compared the performances.

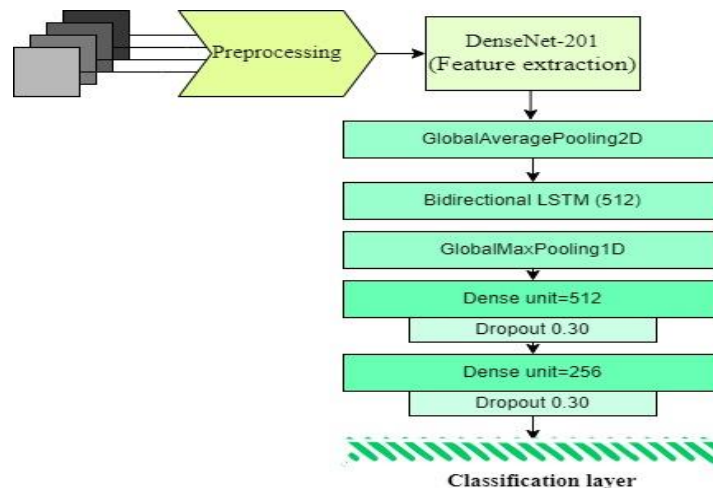
6.2 Transfer Learning

Transfer learning has emerged as a powerful strategy in machine learning, especially when dealing with datasets of limited size. It involves transferring knowledge from one model that has been trained on a large dataset to another model to be trained on a smaller dataset. This approach is particularly beneficial in domains where data collection is expensive or difficult. We used different available models like DenseNet-201, DenseNet-169, I3D, etc. DenseNet is known for its efficient training process, as each layer receives additional inputs from all preceding layers and passes on its feature maps to all subsequent layers. This characteristic makes DenseNet particularly effective for transfer learning because it ensures maximum information flow between layers, which can be very beneficial when training data is limited. I3D (Inflated 3D ConvNet) was Originally designed for video action recognition, I3D transfers the knowledge from 2D models (like Inception) trained on large image datasets to 3D tasks. It expands 2D convolutional networks into 3D, allowing it to leverage temporal information. This makes I3D a good choice for transfer learning in video and motion analysis, where both spatial and temporal dimensions are crucial.

6.3 Experimental DenseNet-201 with BiLSTM Model

Combining DenseNet-201 with BiLSTM has been explored in many classification tasks like [16, 21, 23] where combining DenseNet-201 with BiLSTM helped gain higher recognition accuracies. The advantages of this technique were explored and experimented on a custom model where DenseNet-201 was used as the base feature extractor and later some BiLSTM and Dense layers were added. The detailed architecture is given below,

Figure 4: Architecture of DenseNet-201 with BiLSTM Model.



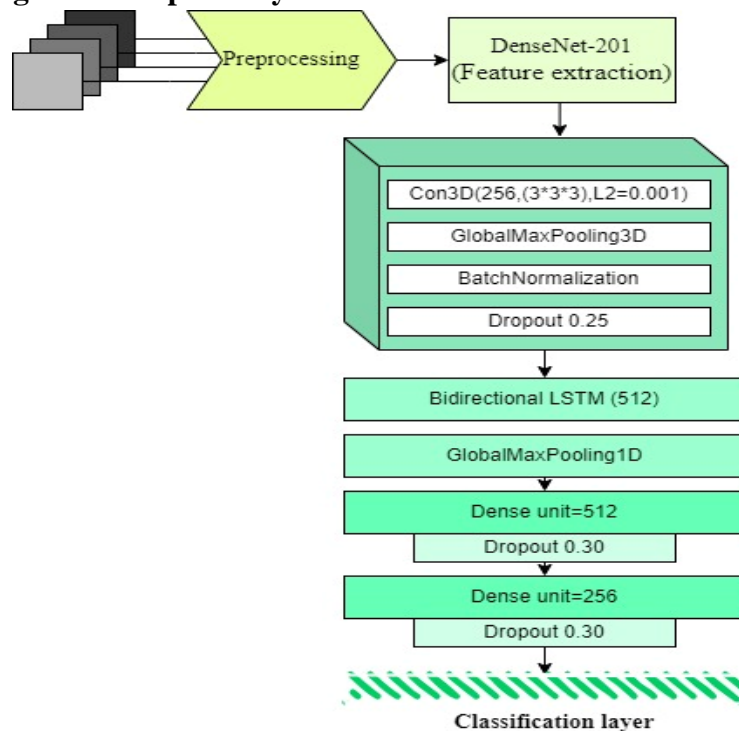
In the development of a model for classifying dynamic hand gestures based on 2D RGB skeleton motion frames, an innovative architecture leveraging the DenseNet-201 convolutional network is proposed. The model is structured to process sequences of 25 frames, each with a resolution of 224x224 pixels. The DenseNet-201, pre-trained on the ImageNet dataset, serves as the foundational feature extractor. To accommodate the sequential input, a TimeDistributed layer wraps the DenseNet-201, allowing the model to apply the convolutional base to each frame independently. To further harness temporal information across the frames, the outputs from the TimeDistributed DenseNet-201 layers are passed through a TimeDistributed global average pooling layer to reduce dimensionality while preserving important spatial features. This is followed by a BiLSTM layer with 512 units, which captures dynamic changes in gesture sequences by processing data in both forward and reverse directions. This layer is crucial for understanding the temporal dependencies between frames. A Global Average Pooling operation across the time dimension condenses the temporal information into a single vector per sequence after that. Subsequently, the model includes multiple Dense layers with ReLU activation to enhance feature learning, interspersed with Dropout layers set at 0.3 to prevent overfitting, which feeds into a SoftMax classifier that outputs the probabilities for the 11 gesture classes. This architecture not only efficiently processes spatial features through DenseNet-201 but also effectively models temporal relationships, making it well-suited for dynamic gesture recognition in real-time applications.

This model demonstrates a robust approach by combining the strengths of deep convolutional networks for spatial feature extraction and recurrent networks for sequence learning, which is essential for accurately classifying sequences of dynamic gestures.

6.4 Proposed Hybrid 3DCNN+ LSTM Model

After conducting several experiments, a hybrid 3DCNN+BiLSTM model with Densenet-201 as the base feature extractor was built. The architecture of the proposed model is given below,

Figure 5: Proposed hybrid 3DCNN+BiLSTM architecture



In the proposed architecture for classifying 2D RGB skeleton motion frames for dynamic hand gestures, the model effectively leverages a combination of convolutional and recurrent neural network components to address the classification of 11 distinct classes. At the core of this architecture is the DenseNet-201 model, utilized as a feature extractor over individual frames of the gesture sequences. This base model is pre-trained on the ImageNet dataset, ensuring robust initial feature detection, and is set to be non-trainable to preserve its learned characteristics.

The input layer accepts sequences of 25 RGB frames, each with dimensions 224x224. These frames pass through a TimeDistributed wrapper that applies the DenseNet-201 model to each frame independently, extracting high-level spatial features. Subsequently, a 3D convolution layer with a kernel size of 3x3x3 processes these features to integrate temporal dynamics, a critical aspect of understanding motion in gestures. Following the convolutional stage, the model incorporates a GlobalMaxPooling3D layer to reduce dimensionality and a Batch Normalization (BN) layer to stabilize and accelerate training. A Dropout layer follows, set at 25%, to mitigate overfitting. The output is then reshaped and fed into a BiLSTM layer, which further refines the model's ability to capture temporal dependencies across frames. The sequence output from the BiLSTM undergoes global max pooling to summarize the essential features before entering a series of fully connected layers. These dense layers, interspersed with dropout layers at a rate of 30%, serve to finalize the feature learning and perform the classification. The final layer employs a SoftMax activation function to output probabilities across the 11 gesture classes. This integrated DenseNet-201 and BiLSTM architecture is designed not only to extract detailed spatial features from individual frames but also to effectively model the temporal sequences inherent in dynamic hand gestures, making it highly suitable for real-time gesture recognition tasks.

7. Overview of Optimizers, and Loss Functions

7.1 Loss Function

A loss function compares the target and anticipated output values to determine how well the neural network mimics the training data. During training, it is attempted to minimize this output difference between the target and expected value by monitoring the loss function.

7.1.1 Sparse Categorical Cross Entropy (SCCE)

In deep learning frameworks such as Keras and TensorFlow, a fundamental loss function used for multi-class classification problems is sparse categorical cross entropy (SCCE). The SCCE calculates the difference between the true labels and the model's anticipated class probabilities, which are usually acquired via SoftMax activation for multi-class classification. The SCCE loss is calculated using the following formula:

$$\text{Loss} = -\sum q(x) * \log(p(x)) \quad (1)$$

Here, Σ (sigma) represents summation across all classes. $q(x)$ denotes the true label for a data point. $p(x)$ signifies the predicted probability for each class from the SoftMax output. And \log represents the natural logarithm (base e). SCCE is computationally efficient for multi-class classification tasks and provides straightforward results with a simple implementation.

7.2 Optimizer

An optimizer is a procedure or function that adjusts learning rates and weights in neural networks. It therefore aids in lowering overall loss and raising precision. In this study, the Adam optimizer was employed.

7.2.1 Adam

Adam optimizer was first introduced by Kingma, D. P. et al. [11]. It is an optimization technique that may be used to iteratively update network weights depending on training data. From adaptive moment estimation comes the name, Adam. To adjust network weights during training, this optimization approach is a further extension of stochastic gradient descent. Adam optimizer modifies the learning rate for each network weight separately, unlike SGD training, which maintains a single learning rate. The inventors of the Adam optimization algorithm are aware of the advantages of the AdaGrad and RMSProp algorithms, two other stochastic gradient descent extensions. As a result, both the AdaGrad and RMSProp algorithms' features are inherited by the Adam optimizers. Adam uses both the first and second moments of the gradients to adjust learning rates rather than just the first moment (mean) as it does in RMS Prop. By the second moment of the gradients, we refer to the uncentered variance. Compared to other optimization algorithms, this one is easier to implement, runs more quickly, uses less memory, and needs less adjusting.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \left[\frac{\delta L}{\delta w_t} \right] \quad v_t = \beta_2 v_{t-1} + (1 - \beta_2) \left[\frac{\delta L}{\delta w_t} \right]^2 \quad (2)$$

The Adam optimizer's operation is represented by the formula above. The decay rate of the average of the gradients is shown here by β_1 and β_2 . Here, m_t is the aggregate of gradients at time t . This research employs an Adam optimizer with an exponential decay learning rate schedule, initially set at 0.0001 and decreasing by a factor of 0.9 every 10,000 steps to optimize convergence. The Adam optimizer, a robust approach, employs an exponential decay learning rate to train deep learning models on dynamic sequences of frames for pattern recognition. This method enhances convergence speed and efficiency, reducing the learning rate exponentially over epochs [11]. This method is particularly effective in complex tasks like dynamic sequence recognition, promoting a more controlled and precise update path, and leading to better generalization and performance on unseen data [12].

8. Training

Initially, a few pre-trained models were used as base feature extractors. These pre-trained models were trained separately to visualize their performance on this dataset. Using pre-trained models as base feature extractors increased real-time accuracy. As multiple layers of Conv3D were employed, running a higher number of epochs resulted in more accuracy, but it was computationally expensive. Therefore, a few previous layers were truncated, and training with simpler 3DCNN+LSTM layers performed better than training solely on the base pre-trained model, achieving similar accuracy. The simpler model was trained for 50 epochs, showing almost the same accuracy as using the additional Conv3D layers.

The same Conv3D 256 layers and LSTM layers were used, maintaining the same structure, and different pre-trained models were replaced as base feature extractors. Among these, DenseNet-201 outperformed other pre-trained models. Different pre-trained models were used to visualize performances. Initially, training was conducted with the unmodified training and validation datasets, using early stopping and model checkpoints to save progress. Later, training with the augmented training dataset increased model robustness. While training with DenseNet-201, it is common to use LSTM or BiLSTM layers for action recognition classification tasks. However, combining 3DCNN with LSTM layers is a well-known technique for classifying dynamic action recognition. DenseNet-201 was trained with added 3DCNN+LSTM layers, two Dense layers with ReLu activation, and one final classification Dense layer with SoftMax activation. This method increased overall test and training accuracy. Replacing LSTM with Bidirectional LSTM resulted in better performance.

Adam optimizer with an exponential decay rate was used throughout the training process, aiding in recognizing complex data. Each model with pre-trained models as the base feature extractor was executed for 50 epochs on the main training and validation dataset. A smaller number of epochs were then executed, considering model complexity, until state-of-the-art performance was reached. After training on the unmodified training dataset, the model was trained again on the augmented training dataset similarly. Different augmentations helped achieve more real-time and training accuracies. Sparse Categorical Cross Entropy (SCCE) was used as the loss function, ReLu as the activation function in all dense layers, and SoftMax as the activation function in the classification layer. Additionally, callback functions like early stopping, ReduceLRonPlateau, and ModelCheckPoints were used with a small patience rate to prevent overfitting and save model progress and the best results.

9. Result Comparison

9.1 Performance Comparison of Different Models

Each DenseNet models except DenseNet-201 with BiLSTM are used with the same structured 3DCNN+BiLSTM layers after. To extract spatial characteristics from the input sequence, this convolutional-LSTM architecture uses a 3D convolutional layer with 256 filters. Then, to capture the most notable features across the spatial dimensions, global max pooling 3D(GMP3D) is used. Batch normalization aids in internal covariate shift, whereas dropout (25%) was used to prevent overfitting. After that, the features are formatted so that they can be processed by LSTM. To find the most informative representation of the sequence, a 512-unit recurrent LSTM layer first analyses the temporal dependencies within the data. Global max pooling is then applied along the temporal dimension. For additional feature transformation, the output is routed through two fully connected layers (512 and 256 units) with ReLU activations. Once more, dropout rates of 30% and 30% are utilized.

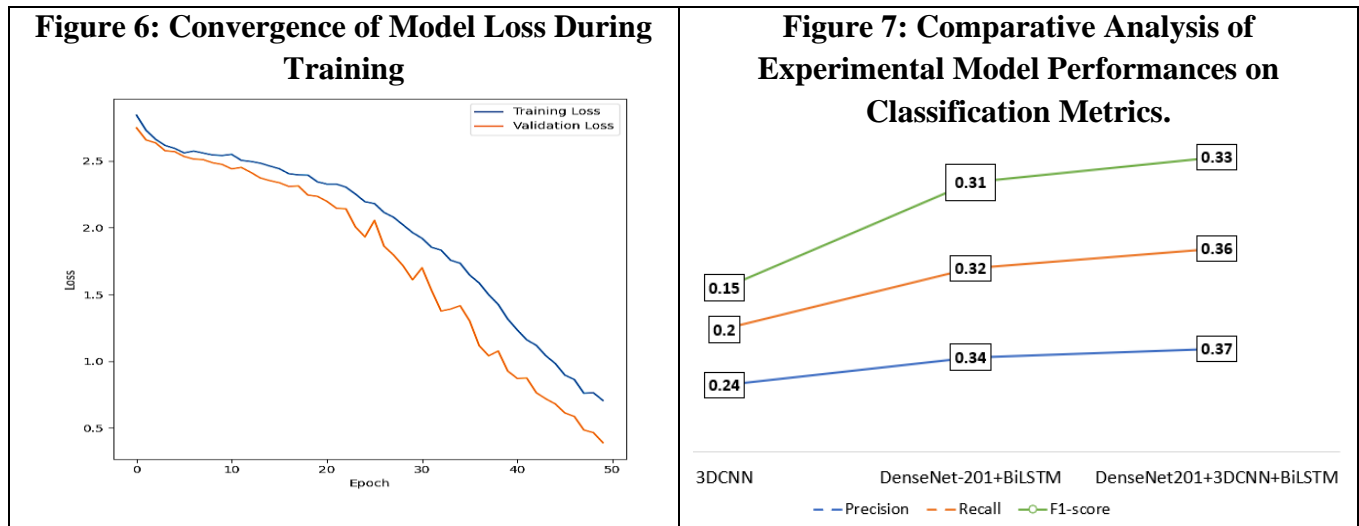
While evaluating the performance of I3D as a feature extractor, it didn't perform well with either 3DCNN or BiLSTM or a combined 3DCNN+BiLSTM structure. So, the I3D model pre-trained on the Kinetics-400 dataset was evaluated separately by adding two Dense layers of unit 512 with a dropout value of 0.4, and later another Dense layer of unit 256 with a dropout value of 0.3 was utilized with a final classification layer with SoftMax activation function. Based on using different pre-trained models wrapped in TimeDistributed function as base feature extractor, here is the detailed comparison below to visualize their performances.

Table 2: Comparison of the Accuracy Levels of Various Models.

Models	Training accuracy	Test accuracy
3DCNN	100	19.70
I3D	94.62	21.21
DenseNet-169+3DCNN+BiLSM	98.93	27.27
DenseNet-201+BiLSTM	99.62	31.82
DenseNet-201+3DCNN+LSTM	98.98	33.30
DenseNet-201+3DCNN+BiLSTM	100	36.36

From the above figure, it can be seen that DenseNet-201 works best among the pre-trained models. Combining 3DCNN+ BiLSTM layers with DenseNet-201 provides a better generalization for unseen data

and training data accuracy. Also proposed model achieved 4.54% more accuracy than the experimental model DenseNet-201 with BiLSTM. In the SignBD-Word dataset, from where we collected our data, they evaluated different pre-trained models and the I3D model outperformed the other pre-trained models for bodypose data [10]. Here, comparing the performances, it is seen that the DenseNet models outperformed the I3D model.



Upon examining the training and validation loss curves in Figure 6, several salient features surface that serve as indicators of the learning behavior of the model. The monotonous drop shown in both graphs indicates that the model is successfully picking up on the underlying patterns in the dataset during training. Interestingly, with just minor variances, the validation loss nearly matches the training loss, which usually denotes strong generalization without appreciable overfitting.

In the Figure 7 graph, the efficacy of three distinct experimental models was evaluated, highlighting their macro average Precision, Recall, and F1-score. The architectures compared the classification matrices among 3DCNN, hybrid DenseNet-201+BiLSTM model, and proposed hybrid DenseNet-201+3DCNN+BiLSTM models. Our results depict a clear superiority of the ensemble model, achieving the highest scores across all metrics: Precision (0.37), Recall (0.36), and F1-Score (0.33).

9.2 Discussion

Although DenseNet-201+BiLSTM is included in experimental model 2, which has a different structure than the proposed model, two same-structured models were also evaluated using a global average pooling layer after DenseNet-201. One model excluded the Conv3D block, while the other included it with the same type of pooling layers. Both computational expense and recognition performance were evaluated. Including the Conv3D block increased the computational expense and required more epochs to achieve state-of-the-art performance. In terms of accuracy, adding the Conv3D layer performed better, whereas experimental model 2 was computationally less expensive. For evaluation purposes, both models were run until the highest training accuracy was reached. The first model without the Conv3D layer was run for 50 epochs, and the other model with the Conv3D block was executed for 100 epochs to achieve the highest training accuracy. After execution, it was observed that adding a Convolutional block resulted in a 3.03% increase in accuracy. However, dynamic action recognition still requires further exploration to reduce the additional computational expense.

Significant limitations were identified due to the dataset's size and quality. Each of the 11 classes had a minimal number of training samples, with ten classes containing only 24 samples each and one class

having 23 samples, complicating the effective training of complex models comprising DenseNet-201, 3DCNN, and BiLSTM layers. Additionally, the extraction process resulted in incomplete keypoints for many data, further challenging the model's learning capability. These data constraints were reflected in the substantial disparity between training accuracy and test accuracy. Reflecting on the limitations and the model's learning capability, it was concluded that adding more data and ensuring a more accurate extraction of keypoints or performing accurately would help in generalizing unseen data more accurately.

9.3 Tools

In the experiments conducted, TensorFlow version 2.15.0 as a deep learning framework was utilized. All models were trained and tested using Google Colab, which provided access to a T4 GPU. The initial setup included a standard environment with 12 GB of RAM. For more intensive computational tasks, Google Colab's high RAM option was accessed, which provided 51 GB of RAM, ensuring sufficient resources for handling larger datasets and more complex model architectures if needed.

10. Conclusion

Although the recent progress in classifying dynamic gestures from dynamic data and creating large datasets for dynamic Human Action Recognition has been notable, BdSL still faces a lack of data and approaches for these purposes. The use of 3DCNN has not yet been extensively explored due to computational expenses. For BdSL, this study is the first to investigate dynamic BdSL classification from keypoint skeleton-based data using a 3DCNN-based approach. It has been shown that using 3DCNN combined with popular pre-trained feature extractor models can achieve better accuracy. Additionally, BiLSTM and GlobalMaxPooling3D contribute to better spatiotemporal data generalization. In the future, collecting more data will allow for better real-time accuracy by exploring more approaches. By developing a more robust architecture and increasing the dataset size, a more effective approach for BdSL will be developed, which will be beneficial for disabled individuals to express themselves using sign language.

References

1. Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708).
2. Li, G., Wu, H., Jiang, G., Xu, S., & Liu, H. (2018, July). A Deep Learning Framework for Recognizing Both Static and Dynamic Gestures. In 2018 IEEE Access (Vol. 7, pp. 23713-23724). Institute of Electrical and Electronics Engineers (IEEE)
3. Guo, Z., & Ying, S. (2022). Whole-Body Keypoint and Skeleton Augmented RGB Networks for Video Action Recognition. Applied Sciences, 12(12), 6215.
4. Hachiuma, R., Sato, F., & Sekii, T. (2023). Unified keypoint-based action recognition framework via structured keypoint pooling. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 22962-22971).
5. Al-Hammadi, Muneer; Muhammad, Ghulam; Abdul, Wadood; Alsulaiman, Mansour; Bencherif, Mohamed A.; Mekhtiche, Mohamed Amine (2020). Hand Gesture Recognition for Sign Language Using 3DCNN. IEEE
6. Duan, H., Xu, M., Shuai, B., Modolo, D., Tu, Z., Tighe, J., & Bergamo, A. (2023). SkeleTR: Towards Skeleton-based Action Recognition in the Wild. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 13634-13644).

7. Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2D pose estimation using part affinity fields," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, Hawaii, USA, 2017, pp. 7291–7299.
8. Pucci, D., Becattini, F., & Del Bimbo, A. (2023). Joint-based action progress prediction. *Sensors*, 23(1), 520.
9. Altaf, Y., Wahid, A., & Kirmani, M. M. (2023, February). Deep Learning Approach for Sign Language Recognition Using DenseNet201 with Transfer Learning. In 2023 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS) (pp. 1-6). IEEE.
10. Sams, A., Akash, A. H., & Rahman, S. M. (2023, July). SignBD-Word: Video-Based Bangla Word-Level Sign Language and Pose Translation. In 2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT) (pp. 1-7). IEEE.
11. Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.
12. Loshchilov, I., & Hutter, F. (2016). Sgdr: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983.
13. Tran, D., Bourdev, L., Fergus, R., Torresani, L., & Paluri, M. (2015). Learning spatiotemporal features with 3d convolutional networks. In Proceedings of the IEEE international conference on computer vision (pp. 4489-4497).
14. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8), 1735-1780.
15. Jirathampradub, H., Nukoolkit, C., Suriyathumrongkul, K., & Watanapa, B. (2020, July). A 3d-cnn siamese network for motion gesture sign language alphabets recognition. In Proceedings of the 11th International Conference on Advances in Information Technology (pp. 1-6).
16. Phong, N. H., & Ribeiro, B. (2022). Action Recognition for American Sign Language. arXiv preprint arXiv:2205.12261.
17. Bulling, A., Blanke, U., & Schiele, B. (2014). A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys (CSUR)*, 46(3), 1-33.
18. Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11), 2673-2681.
19. Shi, X., Chen, Z., Wang, H., Yeung, D. Y., Wong, W. K., & Woo, W. C. (2015). Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, 28.
20. Islam, S., Mousumi, S. S. S., Rabby, A. S. A., Hossain, S. A., & Abujar, S. (2018). A potent model to recognize bangla sign language digits using convolutional neural network. *Procedia computer science*, 143, 611-618.
21. Yang, Z., Li, L., Rao, Z., Meng, W., & Wan, S. (2024). A short-term PV resource assessment method with parallel DenseNet201 and BiLSTM under multiple data features. *Energy Reports*, 11, 2841-2852.
22. Sharma, S., & Kumar, K. (2021). ASL-3DCNN: American sign language recognition technique using 3-D convolutional neural networks. *Multimedia Tools and Applications*, 80(17), 26319-26331.
23. Karacı, A., & Akyol, K. (2023). YoDenBi-NET: YOLO+ DenseNet+ Bi-LSTM-based hybrid deep learning model for brain tumor classification. *Neural Computing and Applications*, 35(17), 12583-12598.

24. Siami-Namini, S., Tavakoli, N., & Namin, A. S. (2019, December). The performance of LSTM and BiLSTM in forecasting time series. In 2019 IEEE International conference on big data (Big Data) (pp. 3285-3292). IEEE.
25. Carreira, J., & Zisserman, A. (2017). Quo vadis, action recognition? a new model and the kinetics dataset. In proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 6299-6308).
26. Kanchon Kanti Podder, Muhammad Abdul Kadir, & Muhammad E. H. Chowdhury. (2021). BdSL-D1500[Data set]. Kaggle. <https://doi.org/10.34740/KAGGLE/DS/1238004>
27. Islam, M. S., Mousumi, S. S. S., Jessan, N. A., Rabby, A. S. A., & Hossain, S. A. (2018, September). Ishara-lipi: The first complete multipurpose open access dataset of isolated characters for Bangla sign language. In 2018 International Conference on Bangla Speech and Language Processing (ICBSLP) (pp. 1-4). IEEE.
28. Jaid Jim, Abdullah Al; Rafi, Ibrahim; Akon, Md. Zahid; Nahid, Abdullah-Al (2021), "KU-BdSL: Khulna University Bengali Sign Language dataset", Mendeley Data, V1, doi:10.17632/scpvm2nbkm.1
29. Rayeed, S. M., Akram, G. W., Tuba, S. T., Zilani, G. S., Mahmud, H., & Hasan, M. K. (2022, March). Bangla sign digits recognition using depth information. In Fourteenth International Conference on Machine Vision (ICMV 2021) (Vol. 12084, pp. 190-199). SPIE.
30. Islam, M. S., Mousumi, S. S. S., Jessan, N. A., Rabby, A. S. A., Abujar, S., & Hossain, S. A. (2019). Ishara-Bochon: the first multipurpose open access dataset for Bangla sign language isolated digits. In Recent Trends in Image Processing and Pattern Recognition: Second International Conference, RTIP2R 2018, Solapur, India, December 21–22, 2018, Revised Selected Papers, Part I 2 (pp. 420-428). Springer Singapore.
31. Ye, H., Wu, Z., Zhao, R. W., Wang, X., Jiang, Y. G., & Xue, X. (2015, June). Evaluating two-stream CNN for video classification. In Proceedings of the 5th ACM on International Conference on Multimedia Retrieval (pp. 435-442).
32. Liu, H., Tu, J., & Liu, M. (2017). Two-stream 3d convolutional neural network for skeleton-based action recognition. arXiv preprint arXiv:1705.08106.
33. Zhu, G., Zhang, L., Shen, P., Song, J., Shah, S. A. A., & Bennamoun, M. (2018). Continuous gesture segmentation and recognition using 3DCNN and convolutional LSTM. IEEE Transactions on Multimedia, 21(4), 1011-1021.
34. Ouyang, X., Xu, S., Zhang, C., Zhou, P., Yang, Y., Liu, G., & Li, X. (2019). A 3D-CNN and LSTM based multi-task learning architecture for action recognition. IEEE Access, 7, 40757-40770.