information

Article

LiST: A Lightweight Framework for Continuous Indian Sign Language Translation

Amrutha K, Prabu P and Ramesh Chandra Poonia *

Department of Computer Science, Christ (Deemed to be University), Bengaluru 560029, Karnataka, India
* Correspondence: rameshcpoonia@gmail.com

Abstract: Sign language is a natural, structured, and complete form of communication to exchange information. Non-verbal communicators, also referred to as hearing impaired and hard of hearing (HI&HH), consider sign language an elemental mode of communication to convey information. As this language is less familiar among a large percentage of the human population, an automatic sign language translator that can act as an interpreter and remove the language barrier is mandatory. The advent of deep learning has resulted in the availability of several sign language translation (SLT) models. However, SLT models are complex, resulting in increased latency in language translation. Furthermore, SLT models consider only hand gestures for further processing, which might lead to the misinterpretation of ambiguous sign language words. In this paper, we propose a lightweight SLT framework, LiST (Lightweight Sign language Translation), that simultaneously considers multiple modalities, such as hand gestures, facial expressions, and hand orientation, from an Indian sign video. The Inception V3 architecture handles the features associated with different signer modalities, resulting in the generation of a feature map, which is processed by a two-layered (long short-term memory) (LSTM) architecture. This sequence helps in sentence-by-sentence recognition and in the translation of sign language into text and audio. The model was tested with continuous Indian Sign Language (ISL) sentences taken from the INCLUDE dataset. The experimental results show that the LiST framework achieved a high translation accuracy of 91.2% and a prediction accuracy of 95.9% while maintaining a low word-level translation error compared to other existing models.

Keywords: Indian Sign Language; Inception V3; INCLUDE dataset; LSTM; artificial intelligence; human–computer interface; natural language processing

Citation: K, A.; P, P.; Poonia, R.C. LiST: A Lightweight Framework for Continuous Indian Sign Language Translation. *Information* **2023**, *14*, 79. https://doi.org/10.3390/info14020079

Academic Editor: Ivan Dunđer

Received: 15 December 2022
Revised: 25 January 2023
Accepted: 26 January 2023
Published: 29 January 2023

1. Introduction

Sign language has always been used by the hearing impaired and hard of hearing (HI&HH), who form an integral part of society. Both verbal and non-verbal communicators use signs and gestures at various levels. The difference between these two communicators is that non-verbal is the only mode for the HI&HH, who use sign language through combinations of finger movements, hand movements, facial expressions, and body postures. These gestures are helpful, as they can convey information and express emotions every day. The outstanding feature of sign language is that it has a dataset based on it, just as each country has its national language. A few examples are Chinese Sign Language (CSL), American Sign Language (ASL), and Arabian Sign Language (ArSL). Recently, the government of India developed a structured and annotated sign language called Indian Sign Language (ISL). Hence, the phonology of ISL is quite new. Phonology can be made evident in spoken language, but hand movements and hand orientations [1] are more subtle in sign language. Furthermore, the ambiguities in sign gestures can be recognised only if the context is understood.

In [2], the authors mentioned the attitudes and responses of individuals to the HI&HH. Earlier, the lack of cross-discipline education made it difficult for HI&HH society to improve itself and become more independent. Based on the census taken in 2011 and updated in

2016, 2.68 crore people out of a population of 121 crores were found to be disabled [3]. The census report revealed that disabilities are more prominent in rural areas and are most prevalent in the age group of 10–19 years. Negligence and ignorance has resulted in little or no effort being taken for the skill development and education of differently abled populations.

Recently, India's government has taken steps to enhance the skills of the HI&HH population. The Indian Sign Language Research and Training Centre (ISLRTC) [4] developed a standard dictionary of 3000 sign words, which was later upgraded to 5000 words. The dictionary acts as a base to form a nationalised syllabus that can be followed by schools and educational institutes that train teachers for special education. Online platforms [5] provide learning opportunities for speech and hearing-challenged students. Educational institutes teach ISL from the basic to the professional level and educate the differently abled students and their parents.

The standardised form and easy availability of ISL vastly increased the demand for its usage. In addition, different training institutes promoted higher education and employability among the HI&HH, which has necessitated their communication with verbal communicators. Human translators are currently appointed to remove the language barrier between the HI&HH and other people. Such dependence can be avoided by introducing an automatic translator that can learn and understand the signs and then translate them into a recognisable form. Automatic translators using deep sequential models are available, but these have not performed outstandingly. As a result, an economical, user-friendly, and simple translator model is lacking for ISL.

SLT can be modelled using various architectures. If the dataset is simple and small, image processing methods such as skin segmentation and convex hull algorithm detect and recognise the signs [6]. As the complexity increases, the SLT can be modelled using AI models. Deep learning models can understand structured information from multiple modalities and have been shown to outperform many state-of-the-art techniques [7,8]. Enormous sign language datasets such as the Phoenix RWTH dataset require much more efficient and attention-based SLT models such as BERT to process them [9,10].

Several factors affect the performance of automatic translator models. As sign language is a combination of hand gestures, orientation, facial expressions, and body pose, overlooking any of these modalities might result in misinterpretation of gestures. Factors such as lighting, speed, and the signer's movements affect the translator's performance. Most Indian SLT models can only recognise gestures if the signer is static and directly facing the camera. The distance between the signer and the camera greatly impacts performance. An SLT framework addressing multiple modalities is either bulky or requires a long learning time. Another parameter affecting model performance is sign language ambiguity, which can be recognised through the previous or succeeding word. In sentences containing "What time" and "How big", the term "what" and "how" are represented by the same hand gesture. The terms are distinguished based on the next word, "time", which is shown by tapping two fingers on your wrist, or "big", which is shown by moving both hands in opposite directions [1]. In [11], the authors mentioned the importance of mouth shape in sign language. The tracking of mouth movements can help understand semantically similar sentences. However, hardly any research has focused on mouth movements explicitly, as they tend to disregard the face during model training.

Much research has been conducted on the translation of ISL into a different understandable form. However, works such as [12–14] show that more than 80% of research still focuses only on isolated sign language translation. Furthermore, less than 10% of studies have used commercially available datasets such as INCLUDE for analysis. In [15,16], the authors proposed a different approach for sign language translation. The work involved converting each signed sentence into separate units or subunits before processing. However, this process increases the complexity, and the sentence may lose its contextual meaning.

This work aims to resolve the existing models' shortcomings by proposing a fast and lightweight deep learning SLT that can process a sign sentence as a whole. The proposed

model uses the entire frame to simultaneously process manual and non-manual signs simultaneously. This was achieved by integrating Inception V3 for feature extraction and a two-layered LSTM architecture for classification and prediction. A set of five different signers sitting facing the camera from the INCLUDE dataset was used to train the model. Complexities such as moving signers and dynamic backgrounds were not taken into consideration during model training.

The contributions of this paper are as follows.

- The dataset is a multi-signer corpus containing 101 ISL sentences and seven signers. The majority of the research in ISL translation has been conducted using isolated or dynamic ISL words.
- The proposed model considers the entire frame instead of cropping the hand area. Hence, both manual and non-manual signs in ISL were considered for processing. Overfitting of the model was avoided by using one of the best combinations of deep learning models: InceptionV3 and LSTM.
- The videos in the INCLUDE dataset are taken in natural settings and consist of frames with blurred images and frames with little or no movements. These frames can be removed, as training the model with these may increase the learning time and also lead to overfitting of the model.
- The smaller convolutions and the asymmetric convolutions of the Inception V3 architecture helped extract the maximum features in a relatively short amount of time and with fewer parameters.
- ISL sentences were classified and predicted using a two-layered LSTM architecture. The apt size of the classifier helped reduce the translation time without compromising accuracy.

The rest of the paper is organised as follows. First, a review of related studies and sign language translator modelling is presented, followed by the introduction of the proposed sign language translation (SLT) model, presentation and analysis of results, and our conclusions.

2. Related Study

In [17], the authors proposed an ANN-DP algorithmic model to recognise invariant signs of the signer's position. Dynamic programming (DP) helps to remove the movement epenthesis and automatically groups words from continuous sign sentences. The model was tested on a large vocabulary CSL dataset. The model was constructed in such a way that the current word was dependent only on the immediate next word. This language model is also known as the bigram model and achieved an improvement of 17% in the recognition rate and translation speed compared to other existing models. In [18], the authors proposed 3D-based sign detection models using graph matching (GM). The proposed model uses the interframe GM method to process both spatial and temporal information. The model was tested using the HDM05 dataset.

In [16], while training a large dataset, the authors introduced the need for subunit modelling to remove existing issues, such as movement epenthesis and performance decline. The model was evaluated using different sign language corpora. The model proved to be efficient even when the dataset size was increased. In [19], the authors introduced 3D-based large SLT systems. The benchmark dataset ChaLearn was used to test the proposed model, which achieved an accuracy of 88.7%.

In [20], the authors proposed a recurrent neural network (RNN) model that processes skeletal features extracted using an external device called LeapMotion. The model was tested using both ASL and SHREC datasets, outperforming existing models by achieving an accuracy of 85.13% and 96%, respectively. In [21], the author proposed a bidirectional long short-term memory (BiLSTM) model with spatiotemporal fusion. Single-word processing, alignment issues, and manual ground truth labelling can be avoided through this method. The model performance was tested using large datasets, i.e., CSL and RWTH-PHOENIX-Weather-2014T, to obtain an accuracy of 81.22% and 75.32%, respectively.

In [22], the authors proposed a hybrid model combining 3D ResNet and the encoder–decoder classification model. The model was trained with the massive RWTH-PHOENIX and CSL benchmark datasets and obtained a precision of 93.9%. In [23], the authors introduced a model that can capture spatial values and temporal information with the help of a graph convolutional network (GCN) and Bidirectional Encoder Representations from Transformers (BERT), respectively. In [24], the authors introduced a rapid method for sign language translation using wearable inertial measurement unit (IMU) sensors on both hands. The model works with the help of two BiLSTM layers and connectionist temporal classification (CTC). CTC helps to understand the difference between each word without knowing sign boundaries. The proposed model reduced the training error by 11%.

In [20], the authors introduced a model that can translate a huge dataset by correctly interpreting each sign's content. The proposed model consists of a generator that identifies the gloss from the sign video based on spatial and temporal information. A discriminator constantly checks the predictions of the generator at both the gloss and sentence levels

As ISL is relatively new, few experiments have been conducted, and an adequate number of models is not available for comparative studies. Vision-based ISL recognition models have shown more economical and real-time application value than sensor-based models. In [25], the authors proposed an ML-based continuous ISL recognition system. This background-invariant model extracts frames with maximum information from video. The features are extracted using discrete wavelet transform (DWT). Finally, the features are processed using HMM [26]. In [27], the authors introduced a selfie-based ISL translation model that uses minimum distance and an artificial neural network (ANN) to classify sign language. Sobel edge detector helped to increase the word-matching score (WMS) to 85%.

A study showed that much work had been done in the fields of isolated ISL and dynamic ISL. In [28], the authors introduced a recognition system that translates emergency sign words. The model extracts hand sign gestures (HSGs) and classifies them using the support vector machine (SVM) model. In [29], the author proposed a 3D convolutional neural network (CNN) model that extracts 3D features from images that are then classified by a multilayer perceptron. The dataset consists of 20 different signs with varying backgrounds and signer speeds. The proposed model achieved an average accuracy of 88.24%.

3. Sign Language Translator Modelling

The proposed model works in three basic steps. The first step is the preprocessing step. The raw data need to be cleaned and manipulated to extract maximum features from each frame. The second step is feature extraction, in which convolution filters move along with each frame to create a feature map. The third step is the recognition and translation step, in which the model, when trained with the data and the ground truth, predicts the best possible output during validation. Several methods of SLT are available. In isolated SLT, the model recognises several independent words that are immediately translated. In dynamic sign word translation, words are individually translated to form a logical construct. The proposed method—Lightweight Sign language Translator (LiST)—takes the input sentence and translates it in the same order. This helps maintain the contextual meaning of the sign language and increases the translator's robustness. Figure 1 shows the overall flow of the translation process, and in the following section, we describe each step in detail.

3.1. Data Preprocessing

The raw data are analysed in data preprocessing to understand the most appropriate information and irrelevant information. The INCLUDE dataset consists of multiple signers with different signing speeds. Accordingly, several frames provide considerably less information due to one or all of the following reasons.

The probability of duplicate frames is higher during the conversion of high-resolution videos. Each signer has a different signing style, which may create motion and out-of-focus blur. The first and last few frames may not contain much information in a sign language

video, and these frames can be excluded before feeding the video to the feature extractor. The following section reviews different approaches for cleaning up the raw sign dataset.

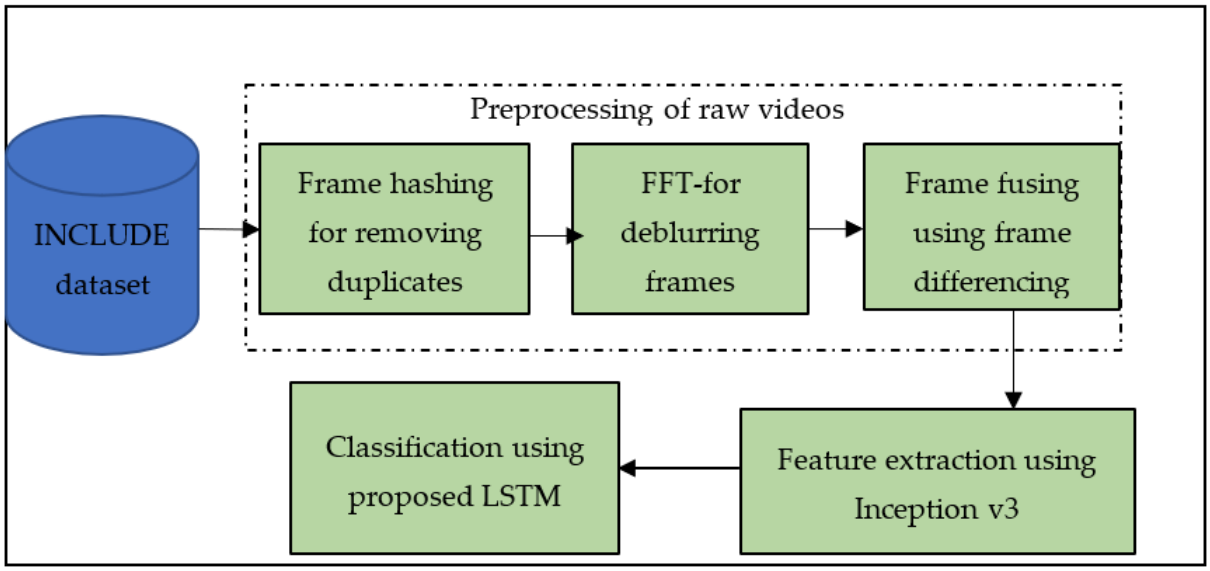


Figure 1. Workflow of the proposed sign language translator model.

3.1.1. Keyframe Extraction

Keyframes provide a maximum amount of information. In a continuous sign language dataset, duplicate frames can be identified by comparison. A unique number can be created for each video frame, forming a signature. Such signatures can be compared to identify and understand whether two frames share a common value, and such frames can be identified as duplicates. The unique value for each frame can be calculated using the MD5 hashing technique [30]. The hash function learns the contents of each image and constructs a unique value for it. The hash values are stored in a hash base, and the hash value of a new entry is compared with the rest of the stack to identify any similar values. If the values are similar, then that frame is considered a duplicate. The frame is labelled with its sequence number and can be easily identified and removed.

The hashed values or the fingerprints are 64 hexadecimal values created for each frame. The hamming distance between the two frames is matched for similarities. The probability of the two compared frames being similar is high if the hamming distance between them is lower than the threshold value. On the contrary, the images are different if the hamming distance between the frames is higher than the threshold value. The hamming distance is not zero in most cases, although the frames appear similar. Hence, a lower threshold value can be set during comparison. The proposed model considers five as the lower threshold value. The hamming distance between two frames is calculated using the XOR function (1). The unique frames are then stored in the vector ($\overline{v}$). Algorithm 1 explains the various steps involved.

$$f'_{n-2} \oplus f'_{n-3} \approx 1 \tag{1}$$

where f_{n-2} and f_{n-3} are two consecutive frames.

Initially, the input, i.e., ISL videos from the dataset ($\overline{V}$), is split into frames, i.e. $f_0, f_1, f_3 \ldots\ldots f_{n-1} \subseteq of \overline{V}$, and a vector hash function is performed on each frame, which helps to compute the hamming distance.

Algorithm 1: Detection and Removal of Duplicate Frames

Input: Sign language sentence videos $\overline{V}$ \\ISL videos from the INCLUDE dataset
Output: $\overline{\overline{v}}$ \\ set of unique frames for each sign sentence
Extract frames from the sign language videos $f_0, f_1, f_3 \ldots\ldots f_{n-1} \subseteq of \overline{V}$
$for(f_0 = 0; f_0 \leq f_{n-1}; f_i++)$
$\overline{\overline{v}} \leftarrow f'_0, f'_1, \ldots\ldots f'_{n-1}$: compute framewise_vectored_hashed_function
$\forall \overline{\overline{v}}$ compute $f'_{n-2} \oplus f'_{n-3} \approx 1$ \\ hamming distance between the consecutive frames
If $f'_{n-2} \oplus f'_{n-3} \approx 1$
delete(f'_{n-3})
$\overline{\overline{v}} \leftarrow$ unique frames // after the removal of duplicate frames
end if
end for

3.1.2. Blur Detection

Motion blur and out-of-focus blur are the two most common issues in moving object videos. These two problems reduce the clarity of the image, thereby affecting the amount of information provided by each frame. Blur detection in each frame can be carried out using the fast Fourier transformation (FFT) function. FFT converts images from the spatial domain to the frequency domain. This is possible by breaking the image into its sine and cosine transforms. The image clarity in the spatial domain determines the number of frequencies in the frequency domain. A sudden change in frequency highlights the area in which blurriness is affected. Once the FFT is calculated, the image can be converted back to the spatial domain using inverse Fourier transformation $(FFT)^{-1}$. FFT and $(FFT)^{-1}$ are computed for a 2D square image using Equations (2) and (3), respectively:

$$F(x,y) = \sum_{a=0}^{N-1} \sum_{b=0}^{N-1} f(a,b) e^{-i2\pi(\frac{x_a}{N} + \frac{y_b}{N})} \tag{2}$$

where, for each image in the special domain, the exponential function corresponds to each point in the Fourier space-$f(x,y)$. Therefore, the sum of the product between the image in the special domain and the base functions yields the image representation in the Fourier space. The $N-1$ function represents the highest frequency.

$$f(a,b) = \frac{1}{N^2} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) e^{i2\pi(\frac{x_a}{N} + b\frac{y_b}{N})} \tag{3}$$

Similarly, in Equation (3), the image is retransformed to the spatial domain using the normalisation function $\frac{1}{N^2}$.

A kernel in image processing is a small matrix that can change the appearance of the original image. It can create effects such as blurring, sharpening, or embossing. Deep learning models such as CNN extract the required features using these kernels. The proposed model uses a 3×3 sharpening kernel that can remove blur detected using FFT.

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

The kernel passes through the entire image expansion, each pixel value is multiplied by the corresponding kernel entry, and the sum is calculated. The sharpening kernel compares the adjacent pixels and enhances the image. Algorithm 2 represents the process conducted using FFT and a Laplacian filter. The deblurred and sharpened images are then stored in $\overline{v}$.

Algorithm 2: Removal Of Blurred Frames And Sharpening

Input: $\overline{\overline{v}}$ \\ Collection of unique frames for each sign sentence
Output: $\overline{v}$ \\ Collection of deblurred frames for each sign sentence
$\forall(\overline{\overline{v}})$ \\ on all the frames in $\overline{\overline{v}}$
Compute FFT \\ detect blurriness using FFT
If$\left(f'_{n-2} > 5\right)$
delete f'_{n-2} \\ if the blurriness is greater than five, then discard
Else
Apply Laplacian filter$(\overline{v})$ \\$3 \times 3$ filter
End if
$\overline{v} \leftarrow$ deblurred frames

3.1.3. Fusing of Frames

The frame representation of the same sentence by different signers showed considerable changes. The speed variations among the signers created a few frames without motion, which can be fused to form a single frame. The frame fusion method must be lossless, i.e., little or no information is lost during the process [31]. The authors of [32] introduced a method for detecting motion in videos with a constant background. This motion detection algorithm is used on the proposed dataset. The absolute difference between the neighbouring cells is calculated to understand whether any objects have moved. The changes are measured using the frame differencing method at the pixel level. The frame difference can be represented by formula (4).

$$\Delta_n = |F_n - F_{n-1}| \tag{4}$$

where Δ_n is the difference between the frames in the n^{th} position and the $n-1^{\text{th}}$ position. A threshold value determines the presence or absence of a motion. Here, the threshold is set as 15% of the pixel intensity of the image, as the slightest movement needs to be detected. The background subtraction process can further increase the accuracy. Once identified, frames with less or no movements can be fused to form a single frame. This process helps to reduce the number of unwanted frames fed into the Inception V3 architecture for feature extraction. Each frame's rapid and subtle motion is detected with the help of the absolute frame differencing function. The processed frames are then temporarily stored ($\hat{v}$) for further analysis. Algorithm 3 represents the steps in the frame-fusing process.

Algorithm 3: Identification Of The First Frame With Motion

Input: $\overline{v}$ \\ deblurred frames
Output: $\hat{v}$ \\ frames with motions
$\forall(\overline{v})$ \\ for every frame in $\overline{v}$
compute $\left|f''_{n-1} - f''_{n-2}\right|$ \\ absolute frame difference between adjacent frames
if $\left|f''_{n-1} - f''_{n-2}\right| = 0$
no movement \\no new information when compared to the previous discard current frame
else
movement detected
end if
$\hat{v} \leftarrow$ frames with motions

3.2. Feature Extraction

Inception Architecture and Specialities

Convolutional neural networks (CNNs) are very popular in computer vision and in other fields and tasks. The model was created to learn complex images by convolving them into smaller portions. The size of the filters present in the network decides the information to be processed and discarded. Another important feature of the CNN model is the stride, which is the number of pixels covered during each iteration. However, if the model is very

large, then the extracted features are not always sufficient. The key solution for feature extraction was perceived to increase the depth of the models until the Inception models were introduced. Added features can improve the model's performance and efficiency at a low cost [33]. Different coevolution models competed in the annual ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) to facilitate the comprehension of their performance [34]. All models were made to train with the ImageNet dataset [35], and the model had to classify the images and scenes. A few CNN-based feature extraction models that gained popularity during the ImageNet Challenge are described below. In most scenarios, CNN was considered one of the best methods for feature extraction. The authors of [36] utilised the CNN architecture to identify and track the movement of the human body. The authors of [37] used the CNN model to identify small objects from an aerial view.

In [38,39], the authors introduced a stacked 3 × 3 convolution architecture, i.e., VGG16 and VGG19, where the values 16 and 19 represent the number of weight layers in the network. This deep architecture can capture many features and easily classify them. The model consists of two fully connected layers, each with 4096 nodes connected to a SoftMax classifier. However, the model has two major disadvantages: it takes a lot of time to train, and its size is large in terms of disc space and bandwidth. Both models, VGG16 and VGG19, are deep and have a size of more than 500 MB.

In [40], the authors introduced a 'Network in Network' architecture called ResNet. The network is a combination of many pooling and convolutional layers. The authors demonstrated that extremely deep networks, such as ResNet, can be trained using residual modules with standard stochastic gradient descent (SGD).

In [40], the authors mentioned the need for better filters to learn complex image features. A model in which all linear filters are replaced with 1 × 1 non-linear convolutional filters was proposed. The authors mentioned that a fully connected layer was not required at the end of the CNN model because the same functionality could be achieved through global average pooling. In [41], the authors introduced a more robust and accurate state-of-the-art model called Inception V2. This model, which is pretrained with the large-scale ImageNet dataset, can detect and classify objects better than classic CNN architecture. With this model, the performance of the sparse network increased in image-processing-related tasks. Combining the additional losses with the classification layer reduced convergence during training. In [33], the authors showed how large convolution filters could be separated into smaller convolutions, such as 1 × 7 and 7 × 1. Improved normalisation and factorisation of convolutions helped the Inception V2 model to perform better than its predecessors.

Figure 2 represents the classic form of the Inception model, which was created to form a single vector of the extracted features. Filters with different shapes analyse a region and learn different patterns. Later on, it was realised that creating a pooling layer in parallel would create a greater impact and increase the model's performance [41]. Another major issue with the classic model is that the 5 × 5 convolutions cost the top layers a huge amount of processing time. Hence, the authors suggested a better-performing model with dimensionality reduction. The modified architecture is depicted in Figure 3.

Inception V3 is a CNN with more feed-forward, convolution, and pooling layers [42]. The convolutions between the first layer (L_1) and the last layer (L_n) with an input of size (H × W), x,y iterators, as well as a kernel (κ) of size $K_n \times K_m$ with m,n iterators and bias b in the first layer, can be expressed as (5) and (6), respectively.

$$I_{x,y} = \sum_{m} \sum_{n} K^{l}_{m,n} O^{l}_{i+m,j+n} + b^{l} \tag{5}$$

$$O^{l}{}_{x,y} = pool(f \sum_{m} \sum_{n} K^{l}_{m,n} O^{l}_{x+m,y+n} + b^{l})) \tag{6}$$

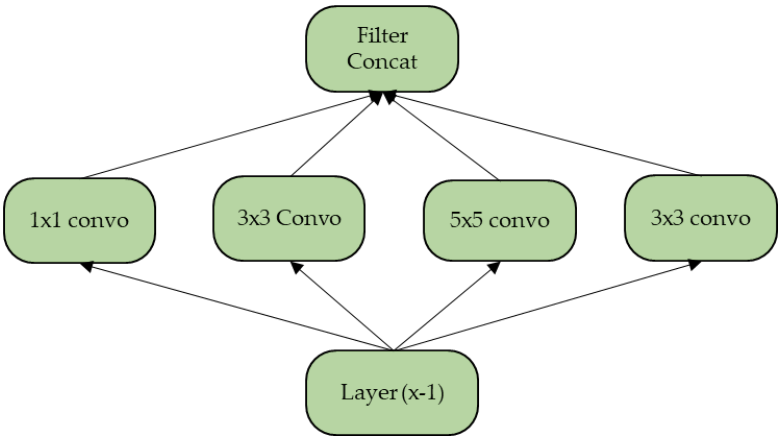


Figure 2. Classic Inception model with filters of different values.

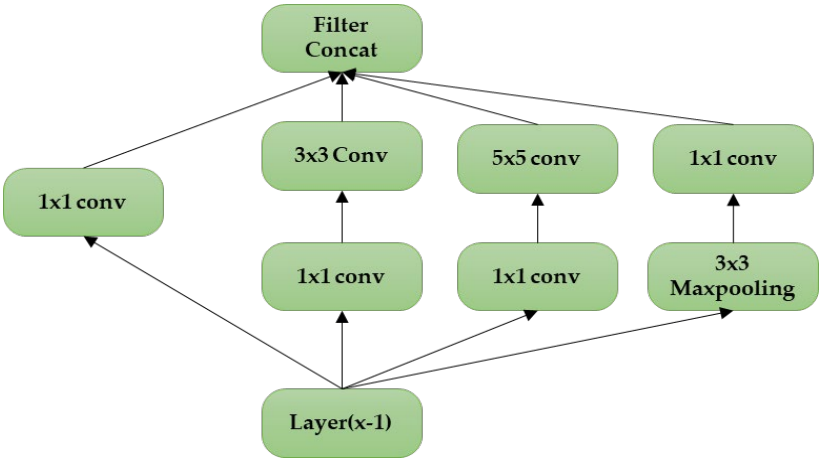


Figure 3. The Inception model modified by adding a MaxPooling layer.

Here, the weight of iterators m and n connects the nodes from the layer $L-1$ to the first layer. The dimensionality reduction and the feed-forward connection of the Inception V3 architecture resulted in high performance with a smaller number of parameters. Table 1 compares different filter-based feature extraction models with Inception V3.

Table 1. Different convolution filter-based feature extractors and classification models.

Sl. No.	Model	Size	Top-1 Accuracy	Top-5 Accuracy	No. of Parameters Considered During Processing	Topological Depth	CPU Time/Inference Step
1	VGG16	528	0.713	0.901	13,83,57,544	23	69.5
2	VGG19	549	0.713	0.9	14,36,67,280	26	84.75
3	ResNet50	98	0.749	0.921	2,56,36,712	-	58.2
4	InceptionV3	92	0.779	0.937	2,38,51,784	159	42.25

3.3. Long Short-Term Memory(LSTM)

LSTM is an architecture that can retain the information passed to it. The memory cells present in the model can decide which information is to be retained and which is to be discarded. Therefore, the cell can help to form an output that depends on the historical input rather than the immediate past. The forget gate constantly modifies the cell state placed directly below. The forget gate multiplies the zero value by all the information that is to be removed. As a result, the relationship can be maintained among words in a long sequence.

The cell state consists of a sigmoid activation function that can shrink the values between 0 and 1. The input gate, also known as the save vector, includes tanh activation, which can regulate the value between -1 and 1. The output gate decides which memory should be fed to the next layer. This gate is also controlled using sigmoid activation. A significant hidden layer is placed between the input and output layers. The classic LSTM model consists of a single hidden layer before the feed-forward output layer.

At each timestamp ($\hat{t}$), the information update in each cell can be expressed as Equations (7)–(12) [43].

$$\text{Input gate}: \ I_{\hat{t}} = \sigma(W_i X_{\hat{t}} + U_i h_{\hat{t}-1} + b_i) \tag{7}$$

$$\text{Output gate}: \ O_{\hat{t}} = \sigma(W_o X_{\hat{t}} + U_o H_{\hat{t}-1} + b^o) \tag{8}$$

$$\text{Forget gate}: \ f_{\hat{t}} = \sigma(W_f X_{\hat{t}} + U_f H_{\hat{t}-1} + b^f) \tag{9}$$

$$\text{Memory Cell}: \ C_{\hat{t}} = f_{\hat{t}} \odot C_{\hat{t}-1} + I_{\hat{t}} \odot \tanh(W_c X_{\hat{t}} + U_c H_{\hat{t}-1} + b_c) \tag{10}$$

The activation function of the entire model is calculated using

$$H_{\hat{t}} = O_{\hat{t}} \odot \tanh(C_{\hat{t}}) \tag{11}$$

where W_i, W_o, W_f, and W_c are the weights in the input, output, forget gate, and memory cell, respectively. σ represents the activation function sigmoid, and $\odot$ represents the element-wise product between the terms.

The loss function is computed during model training using cross entropy between the ground truth $G(X)$ and the model-predicted values ($P(X)$).

$$\text{Loss}H(G_x P) = -\sum_{X} G(X) \log P(X) \tag{12}$$

The standard LSTM model uses sigmoid and tanh as gating and output activation functions, respectively [44]. The rectified linear unit (ReLU) activator of the LSTM contributes to fast convergence and the creation of a non-linear output by learning the non-linear dependencies. Furthermore, the activator plays a key role in creating a sparse network that can process the parameters in a faster mode [45]. The performance of the model can be improved by modifying the activation function. The authors of [34], introduced parameterised exponential linear units (PELUs), which rectify the disadvantages of LeakyReLU. The minimum values are multiplied by a small value, which does not allow the values to go below zero. The authors of [46] introduced yet another version of ReLU, Exponential ReLU. The non-linear exponential curve can reduce the vanishing gradient problems. The right activation function can reduce the limitations (vanishing and exploding gradients) of classic RNN models. Figure 4 shows a pictorial representation of the model fitting.

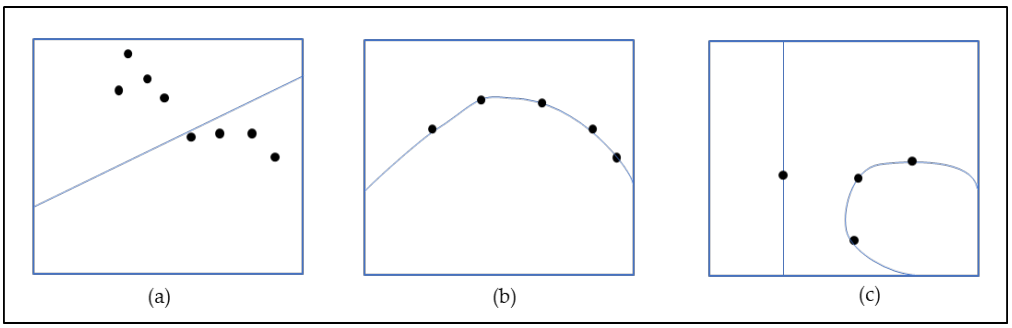


Figure 4. (**a**) Underfitting, where the model cannot capture the curve of the data. (**b**) The model fits perfectly to the curve of the data points; this curve can also train the unseen data. (**c**) The exact structure of the data cannot be captured, i.e., overfitting, where the points are covered but not in the correct shape.

The ReLU activation function can be represented mathematically as follows [39]. The activation function of ReLU is considered as $\sigma_{x,y}$ and is expressed as (13):

$$\sigma_{x,y}(z) = Max(0, z) \tag{13}$$

In addition, for the i^{th} unit of the hidden layer, n hidden layers and m samples are positioned such that $i_n \in \{1, 2, 3, \ldots, m\}$, with the hidden layers $\varpi \in \{h_1, h_2, h_3 \ldots, h_m\}$.

The matrix with diagonal values that correlate with the hidden layers can be expressed as:

$$\Psi_{i_n}^{1,\varpi} = 1if(M\bar{Wi})_{i,\varpi} > 0\&0, OW \tag{14}$$

where the M input matrix and $\bar{Wi}$ are the weights of the input matrix.

3.3.1. Vanishing and Exploding Gradients

The loss function is calculated in each epoch during model training based on a particular weight. The weights are adjusted slowly until the required accuracy is obtained. This process is also known as backtracking. The gradients of each layer are multiplied by the weights; if the weights are low, the product is too small, and the model cannot learn anything. This effect is called the vanishing gradient.

On the contrary, if the gradient values are too high, the product is very high, and the model skips a few essential features. This effect is known as the exploding gradient. The LSTM model does not suffer from either of these two effects. The gradients in LSTM have activators from the forget gate. Hence, the weights at each time step are modified accordingly [43]. Figure 5 represents a single cell of the LSTM model.

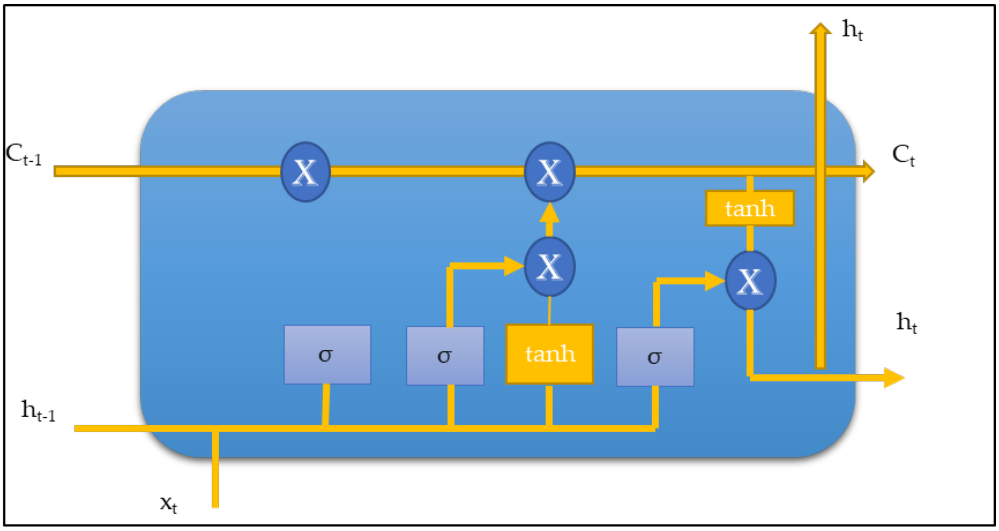


Figure 5. Basic LSTM block with memory gates.

3.3.2. Depth of the Model

The best way to control the operation of a deep learning model is to adjust its number of layers and nodes. This also determines the model's learning capacity. The classic LSTM model consists of one hidden layer and input and output layers. This vanilla model has less learning capacity and a high probability of the model becoming underfit. Ideally, a single hidden layer and many nodes can learn information from the dataset, but it entirely depends on the type of dataset used. The best solution is to gradually increase the number of layers so that the model understands that it has a high capacity to learn the dataset [47]. Increasing the hidden layer more than required would decrease accuracy during the testing phase. The model can learn information provided in the training data, but it cannot use those data when two similar but unknown types of errors are encountered during learning. Training errors can be reduced by cleaning the data.

Similarly, testing error is another predominant issue. When new information is fed into the system, it cannot understand it. Measures should be taken to decrease both training and testing errors.

4. Sign Language Translation

4.1. Dataset Description

ISL is a rich, complete language that follows a particular grammar and word sequence. The authors of [48] described an ISL dataset that consists of 0.27 million frames with more than 4000 videos and approximately 200 sign words. Experienced signers provide a real-time scenario, as the dataset has includes multiple signers. The model can learn the subtle changes in signers' manual and non-manual gestures. The dataset was collected with the help of seven senior students, who covered almost all real-life sentences. The signers stood approximately five to seven feet from the camera and were made to sign facing it. The dataset includes videos in which all manual and non-manual modalities are incorporated [48]. English annotations were available for each video and the isolated words. The seven signers were in both sitting and standing positions.

Each sentence is a combination of a minimum of two words with varying complexities. All signers were right-hand dominant and represented signs using both manual and non-manual gestures in each sentence. Figure 6 shows a snippet of the dataset that explains the diversity and complexity. The significant difference between a verbal sentence and a sign sentence is that sign sentences are devoid of connectives and sign only the important terms, and the sentence format may not be grammatically correct when translated into text or audio. This scenario occurs especially when signing information that consists of tense and plural values. The dataset also includes annotations known as glosses that help avoid such ambiguities. When classifying and predicting the sentence, the classifier is mapped to the complete English sentence or ground truth to create the exact text. Table 2 shows the gloss representation of the signs, along with the ground truth. These annotations were created with the help of senior ISL experts.

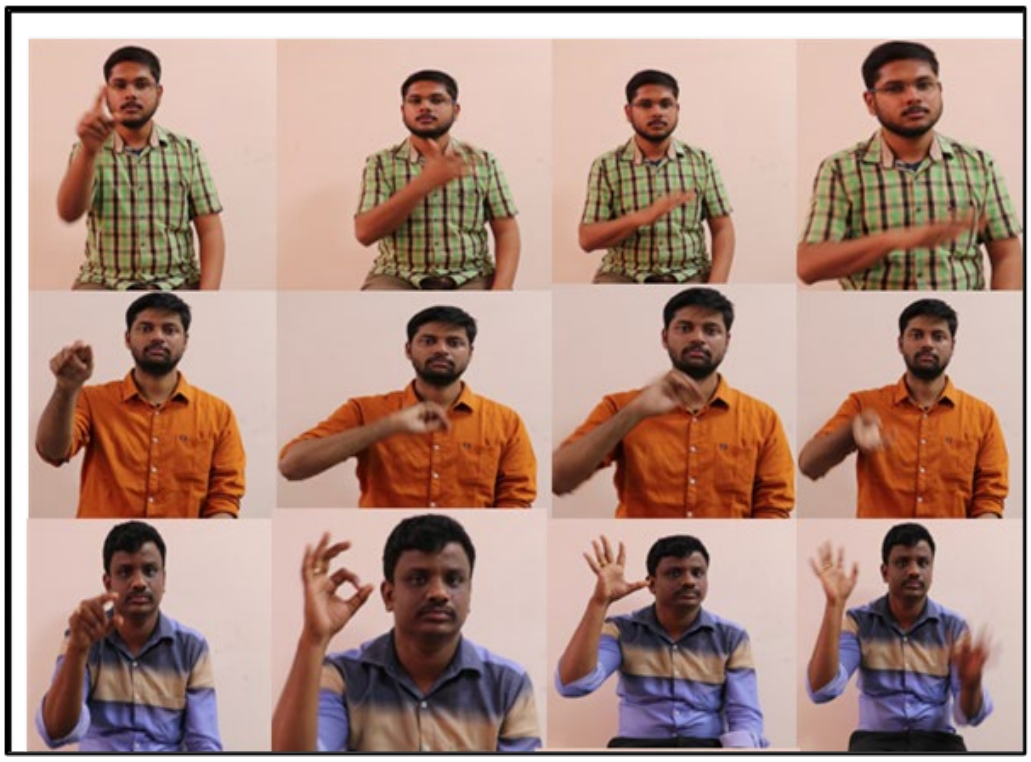


Figure 6. INCLUDE dataset snippet: dataset with multiple signers and signing speeds.

Table 2. Ground truth with sign gloss representation.

Sentence	Sign Gloss
Are you free today	YOU FREE TODAY
Are you hiding something	YOU HIDE SOMETHING
Bring water for me	BRING WATER ME
Can I help you	I HELP YOU
Can you repeat that, please	YOU REPEAT PLEASE
Comb your hair	COMB YOUR HAIR
Congratulations	CONGRATULATIONS
Could you please talk slower	YOU PLEASE TALK SLOWER
Please do me a favour	DO ME FAVOUR
Do not abuse him	DO NOT ABUSE HIM
Do not be stubborn	DO NOT BE STUBBORN

4.2. Feature Extraction

The Inception network takes in the image and collects its features using the kernels. Parallelly, the dimension of the image is reduced. The first convolution filter of size 1×1 performs a point-wise product of every pixel in an image. The resulting output is a 3D vector with dimensions of $1 \times 1 \times K$, where the number of filters is the third dimension. This filter learns more details related to the depth of the image rather than any patterns. In addition, the 1×1 filter reduces the number of channels in the input image and creates a bottleneck. The pooling layers placed along with these filters downsample the image. A network with more depth can perform better downsampling. Here, the 3D (RGB) frames are fed with slight illumination variations. Figure 7 explains the different modules present in the Inception V3 architecture that help extract maximum features.

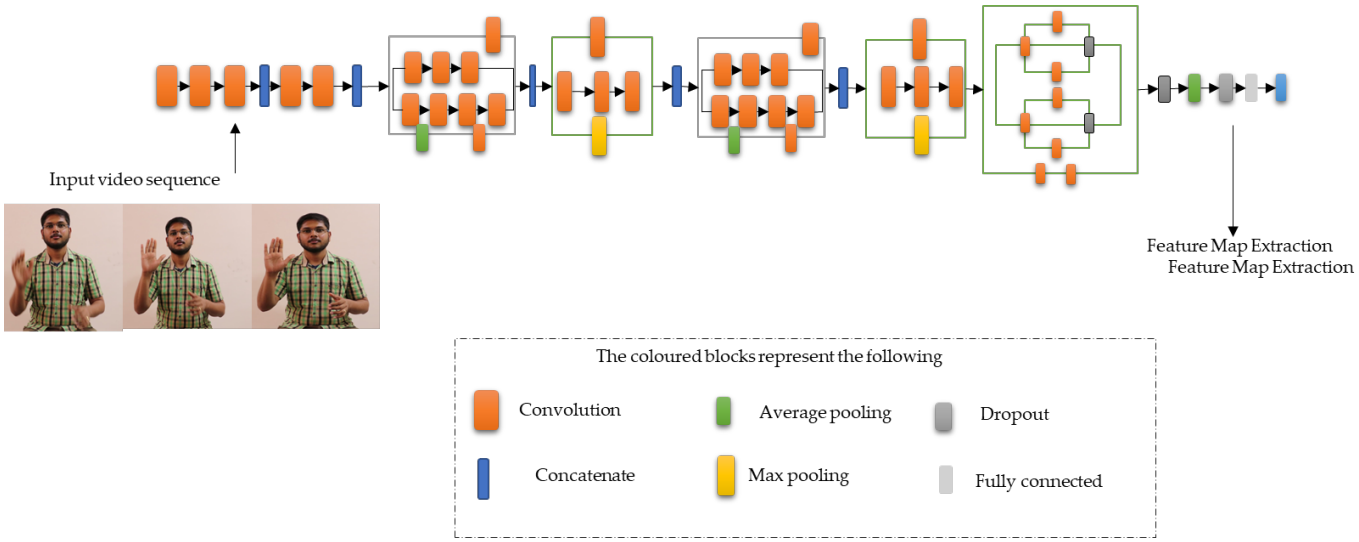


Figure 7. Inception V3 architecture extraction of features from each frame.

The model's 3×3 and 5×5 filters learn the spatial patterns and features of varying shapes and sizes. Both filters are fed 3D values containing height, width, and depth ($H \times W \times D$). The filters have an added padding feature, which helps the image to remain the same size throughout processing. The padded values are masked before feeding into the sequential classifier. A normal Inception architecture consists of 1283×3 filters and 325×5 filters. The ISL dataset consists of both manual movements (hand gestures) and non-manual movements (facial expression, head movements, shoulder movements, and body pose). These filters must detect and learn manual and non-manual movements to translate the sign sentence precisely. The availability of multiple signers helps the model learn the features in different ways. The average pooling and the max-pooling layers after

a set of convolutions always help maintain a balance between the size of the feature map and the frame size. This feature map achieved satisfactory results during translation. The steps involved in feature extraction are explained in Algorithm 4.

Algorithm 4: Extracting Feature From The Frames

Input: $\hat{v}$ \\ cleaned frames of sign sentences
Output: Extracted features
1 Inception V3← $\forall(\hat{v})$ \\ feed the frames
2 tf. Keras. applications. Inception V3(
include_top = false,
weights = "imagenet",
input_shape = (IMG_SIZE, IMG_SIZE, 3), \\ (299,299,3)
pooling = avg,
)
3 Extract general features

The model weights are set to the pretrained ImageNet weights to extract the feature from each frame. The input image shape is $299 \times 299 \times 3$, where 3 represents the number of channels. Average pooling is applied to the last convolution layer to convert the extracted features into a 2D tensor.

4.3. Proposed Lightweight Classifier

The Inception architecture passes 2048 features to the classifier. The bulky dataset provides more features and more information. The hidden layers in the network study the feature map and the ground truth of each video. The features and the ground truth are fed into the model, which learns and predicts the translated audio after validation. The performance of the model was tested using metrices like Accuracy, Learning speed and the translation rate.

The step-by-step operation of the proposed model is depicted in Figure 8.

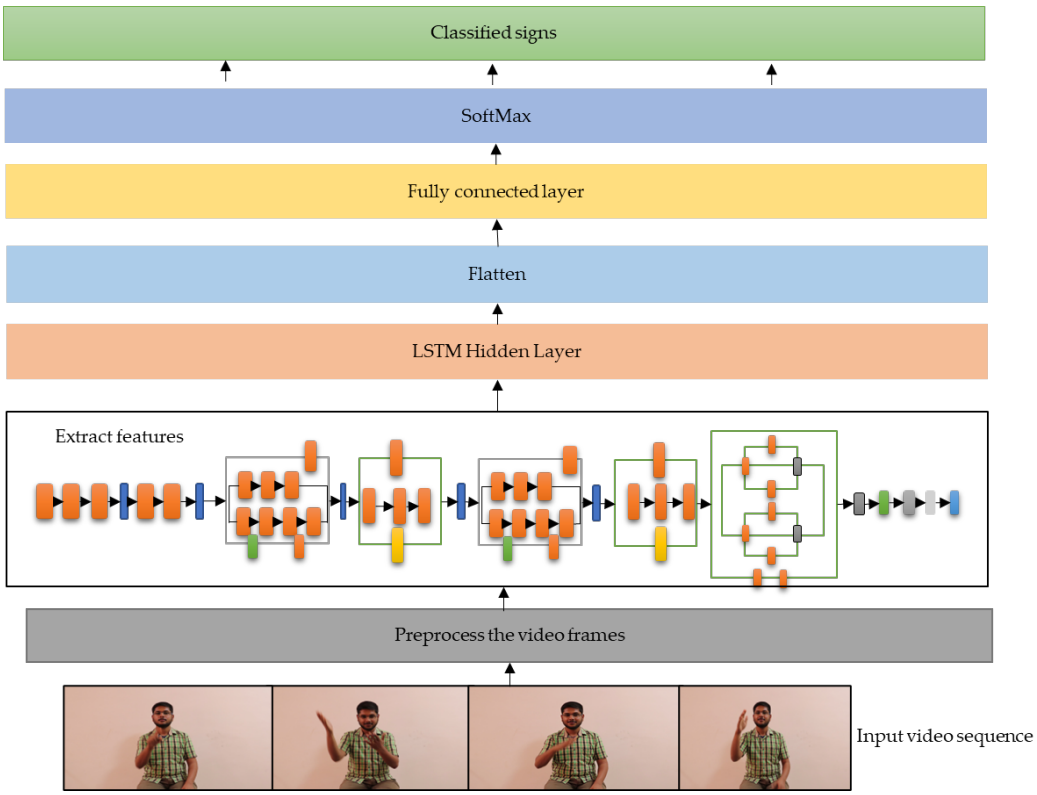


Figure 8. Various steps of the proposed Lightweight Sign language Translator.

The model weights are set to the pretrained ImageNet weights to extract the features from each frame. The input image shape is 299 × 299 × 3, where 3 represents the number of channels. Average pooling is applied on the last convolution layer to convert the extracted features into a 2D tensor. The final classification and prediction of the sign sentences are explained in Algorithm 5.

Algorithm 5: Classification Of The Sign Sentences Using LiST

Input: Feature vectors extracted from the Inception V3 architecture
Output: Translation accuracy, loss function, translated text and translated audio
1 ∀ extracted features using LSTM
2 tf.keras.layers.LSTM(
dropout = 0.5
activation = "relu"
optimizer = "adam"
loss = "sparse_categorical_crossentropy"
metrics = ["accuracy"])
execute for 500 epochs \\ until maximum accuracy is achieved
compute loss function and translation accuracy

5. Results and Analysis

The model recognises gestures and translates signs into audio based on the features present in each video. Maximum features from the dominant hand, non-dominant hand, and face are extracted from each frame using the Inception V3 architecture. These features are then converted to a 2D vector known as the vector map and fed into the classifier. The LSTM classifier model learns these features during training epochs and translates them into corresponding text. The experiment included 2048 parameters for classification.

In the parameter map, let $\alpha_1, \alpha_2, \ldots \alpha_n$ be the set of parameters for a sign sentence (S1). Then, $\alpha_1, \alpha_2, \ldots \alpha_n$ is a combination of dominant hand features and non-dominant hand features, along with facial features. The LSTM predicts the sign based on the presence or absence of these features using (15).

$$\sum_{1}^{K} P(S_1 | \alpha_{n1} \ldots\ldots\ldots \alpha_{nk}) \tag{15}$$

The model was fed small batches of 10 with 0.5 dropouts in the dense layer. The Adam optimiser with a 0.001 learning rate and ReLU activation layer helped increase the translation accuracy. The cross-entropy method calculated the loss during each epoch and reduced it to 0.2129.

The INCLUDE dataset was tested against other benchmark models, i.e., CNN, RNN, and LSTM. Primary performance evaluation metrics such as translation accuracy, learning time, memory usage, and prediction probability helped with model evaluation. The feature extraction and the classification method are different for each model. The proposed model's analysis is summarised in the following section. Finally, the model's performance in word-by-word translation was also analysed using WER (word error rate) metrics

The first sequential translation model tested was the CNN-RNN model. The CNN-RNN module extracted spatial and temporal information from the sign video. However, the model performance was not efficient in identifying the sign sentences. In addition, the training time exceeded the acceptable duration.

Secondly, a pose-based feature extractor was used to reduce the limitations of the CNN-RNN translator model. MediaPipe, an open-source library provided by TensorFlow, can detect human movements with the help of predetermined key points. The pose- estimator tracked the signer's hand movements and facial expressions. As a result, the feature map was easily created and fed into the LSTM classifier. The LSTM classifier captured the temporal information from the sign videos. However, the model misinterpreted gestures as the movements of the key points, and the video speed was not synchronised. The model

captured only major hand movements and failed to capture the different signer speeds. Hence, the pose-based feature extractor model was also considered incompetent for the INCLUDE dataset.

Next, Inception architecture was chosen to extract the features from the video. The Inception architecture is smaller and more efficient compared with VGG16 and VGG19. A three-layered LSTM model was employed for classification. The model performed better than its predecessors, but its accuracy was not satisfactory, and the training time was longer than the acceptable period. CPU usage also increased considerably.

The proposed model showed improved performance for all the parameters, such as accuracy, learning time, conversion speed, and CPU usage. Table 3 shows a comparison of the experimental results.

Table 3. Performance comparison of different models trained using the INCLUDE dataset.

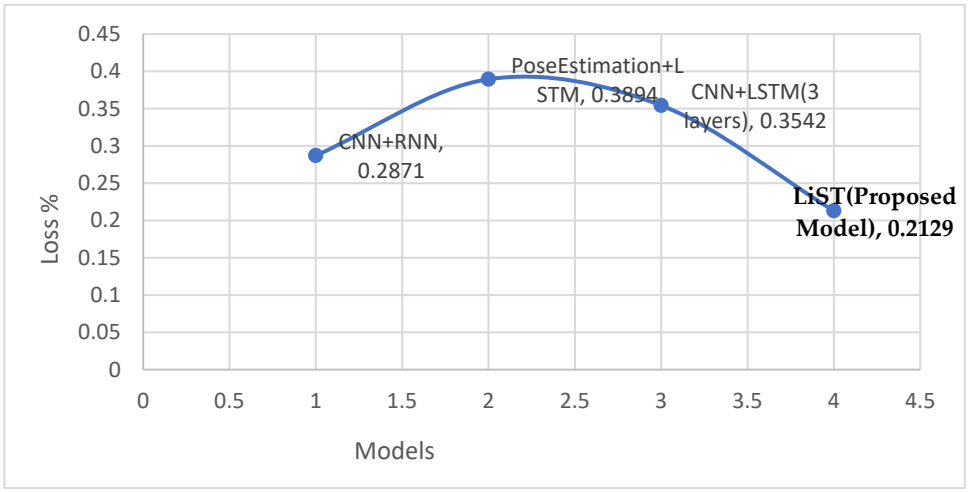
Model Type	No. of Signers	Epochs	Modalities Considered			Accuracy %	Loss %	Prediction Probability %	Conversion Rate
			Full Frame	Face	Hands				
CNN + RNN	5	425	✖	✖	✓	65.1	0.287	60.00	High
PoseEstimation + LSTM	5	500	✖	✓	✓	66.7	0.389	70.00	Low
Incepton V3 + LSTM (3 layers)	5	530	✓	✓	✓	75	0.354	85.00	High
LiST (Proposed Model)	5	500	✓	✓	✓	92.5	0.212	95.90	Medium

The proposed model was also compared with other deep learning SLT models, which obtained considerably high accuracy and prediction probability. A different dataset was also used depending on the type of ISL; instead of using complete ISL sentences, the authors attempted to form a sentence using dynamic sign words. Table 4 compares the proposed model with other deep learning frameworks used to convert ISL words and sentences. Given the modalities considered and the size of the dataset used to train the model, the proposed framework outperformed the other models in terms of accuracy.

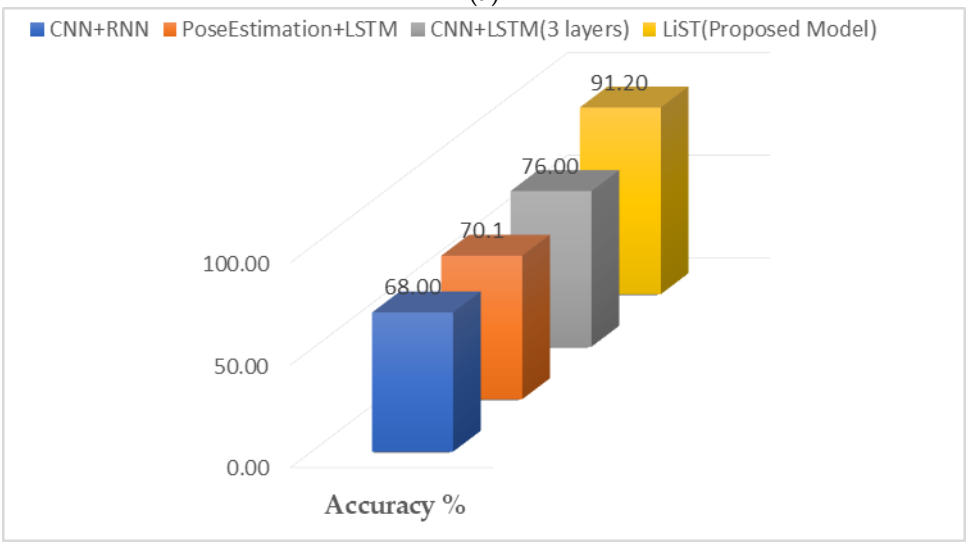
Table 4. Accuracy comparison between LiST and other benchmark models.

Authors	Year	Model	Accuracy	Dataset	External Components	Modalities Considered		
						Full Frame	Face	Hands
G. Ananth Rao and PVV Kishore	2017	Minimum Distance + ANN	85.58	Collection of words from ISL dictionary	✖	✖	✖	✓
Anshul Mittal et al.	2019	LeapMotion + Modified LSTM	72.3	ISL sentences	✓	✓	✓	✓
S. Sharma et al.	2021	IMU + CNN + LSTM	89.9	Isolated words from ISL dictionary	✓	✖	✓	✓
		LiST	91.2	ISL sentences	✖	✓	✓	✓

Performance evaluation metrics such as accuracy and loss function were computed and are compared in Figure 9a,b. The loss functions were calculated using the sparse categorical cross-entropy function. Because the classification predicts a sign sentence based on the maximum probability function computed based on the features present in the video, an accurate loss function can be calculated using the sparse function. The loss function was calculated during each epoch, and a final value 0.2129 was recorded when the maximum accuracy was achieved. Similarly, other sequential models were computed, and the analysis shows that the proposed model achieved the lowest loss function. Figure 9b represents the performance evaluation of the proposed model compared to other existing sequential models. The proposed model achieved outstanding performance with respect to prediction probability, and the model's training time was relatively short, making it more efficient.



(**a**)



(**b**)

Figure 9. Performance comparison between LiST and existing models. (**a**) Loss function comparison; (**b**) accuracy comparison with other existing models.

The proposed model was also tested for word-by-word accuracy using the *WER* (word error rate) ratio [49,50]. The *WER* metrics help to calculate the similarity between the predicted sign and the ground truth. Operations such as substitution, deletion, and insertion are used to align the output sequence with the input sequence.

$$WER = \frac{\sum IDS}{Num} \tag{16}$$

Here, in Equation (16), I represents the total number of insertions, D represents the total number of deletions, S represents the total number of substitutions, and Num represents the number of words in the ground truth [51]. The *WER* was also calculated and compared with the SLT models, such as CNN-RNN and PoseEstimation with LSTM, using the same dataset. The experimental results show that the proposed model performed better by lowering the error rate during translation. Table 5 and Figure 10 compare the *WER* between the proposed model and existing models. During the training time, the *WER* of the proposed model was 15.1%, which is 9.6% less than the three-layered LSTM architecture. During testing, the model's error rate was reduced to 13.5%, which is 8.37% less than three-layered LSTM. The proposed model showed less accuracy in complex sentences such as "I am in

a dilemma, what to do?" and in sentences that included highly complex and fast hand movements.

Table 5. WER comparison of different sign language translators using the INCLUDE dataset.

Model Name	Training Data WER%	Testing Data WER %
CNN + RNN	34.1	31.25
PoseEstimation + LSTM	32.9	25.7
Inception V3 + 3layer LSTM	28.7	21.87
LiST(Proposed)	15.1	13.5

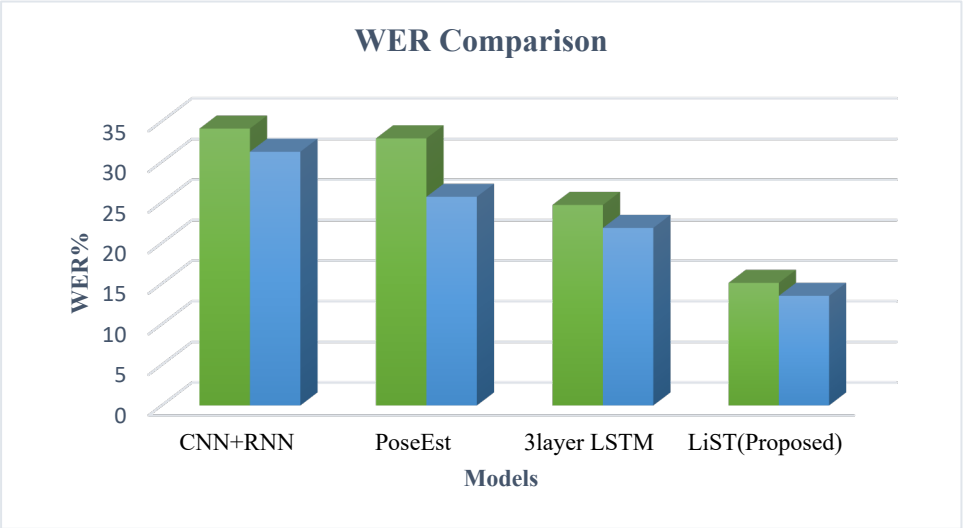


Figure 10. Training and testing WER comparison.

The experimental analysis of accuracy, learning rate, loss function, and prediction probability shows that the proposed model is comparatively better than other sequential SLT models.

6. Conclusions

A lightweight computer-vision-based deep learning framework is proposed to detect and translate Indian Sign Language sentences. The model's efficacy was evaluated using translation accuracy, prediction probability, and word error rate (WER). Hyperparameters such as the size of the model, the speed of execution, and memory usage were also compared. Analysis was performed using a dynamic and multi-signer dataset called INCLUDE, which consists of more than 100 sign language sentences of varying complexity. The experimental study showed that the proposed model achieved an average translation accuracy of 91.2% and an average prediction probability of 95.9%, and the WER was reduced to 13.5%. The model is also faster and more memory-efficient than other models. Comparative analysis revealed that the proposed framework is more robust than most of existing models.

In future iterations, the proposed translation framework can be further trained by having signers use different poses and orientations, such as standing or being mobile. Furthermore, the model can also be modified to recognise and translate regional Indian sign languages.

Author Contributions: Conceptualization, A.K. and P.P.; methodology, A.K. and P.P.; software, A.K. and P.P.; validation, R.C.P.; formal analysis, R.C.P. and P.P.; investigation, R.C.P. and P.P.; resources, A.K. and R.C.P.; data curation, A.K. and P.P.; writing—original draft preparation, A.K. and P.P.; writing—review and editing, A.K., P.P. and R.C.P.; visualization, A.K. and P.P.; supervision, P.P. and R.C.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available upon request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Verma, V.K.; Srivastava, S. A perspective analysis of phonological structure in Indian sign language. *Smart Innov. Syst. Technol.* **2018**, *79*, 175–180. [CrossRef]
2. Miles, M. Studying Responses to Disability in South Asian Histories: Approaches personal, prakrital and pragmatical. *Disabil. Soc.* **2011**, *16*, 143–160. [CrossRef]
3. Disabled Population in India as per Census 2011 (2016 updated)—Enabled.in. Available online: https://enabled.in/wp/disabled-population-in-india-as-per-census-2011-2016-updated (accessed on 29 March 2022).
4. ISL Dictionary Launch | Indian Sign Language Research and Training Center (ISLRTC), Government of India. Available online: http://www.islrtc.nic.in/isl-dictionary-launch. (accessed on 29 March 2022).
5. Raghuveera, T.; Deepthi, R.; Mangalashri, R.; Akshaya, R. A depth-based Indian Sign Language recognition using Microsoft Kinect. *Sādhanā* **2020**, *45*, 34. [CrossRef]
6. Taskiran, M.; Killioglu, M.; Kahraman, N. A real-time system for recognition of American sign language by using deep learning. In Proceedings of the 2018 41st International Conference on Telecommunications and Signal Processing (TSP), Athens, Greece, 4–6 July 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–5.
7. Voulodimos, A.; Doulamis, N.; Doulamis, A.; Protopapadakis, E. Deep learning for computer vision: A brief review. *Comput. Intell. Neurosci.* **2018**, *2018*, 7068349. [CrossRef]
8. Bianchini, M.; Dimitri, G.M.; Maggini, M.; Scarselli, F. Deep neural networks for structured data. In *Computational Intelligence for Pattern Recognition*; Springer: Cham, Switzerland, 2018; pp. 29–51.
9. Chowdhury, R.; Bouatta, N.; Biswas, S.; Floristean, C.; Kharkar, A.; Roy, K.; Rochereau, C.; Ahdritz, G.; Zhang, J.; Church, G.M.; et al. Single-sequence protein structure prediction using a language model and deep learning. *Nat. Biotechnol.* **2022**, *40*, 1617–1623. [CrossRef]
10. Zhou, Z.; Tam, V.W.; Lam, E.Y. SignBERT: A BERT-Based Deep Learning Framework for Continuous Sign Language Recognition. *IEEE Access* **2021**, *9*, 161669–161682. [CrossRef]
11. Koller, O.; Ney, H.; Bowden, R. Deep learning of mouth shapes for sign language. In Proceedings of the IEEE International Conference on Computer Vision Workshops, Santiago, Chile, 7–13 December 2015; pp. 85–91.
12. Saggio, G.; Cavallo, P.; Ricci, M.; Errico, V.; Zea, J.; Benalcázar, M.E. Sign language recognition using wearable electronics: Implement- ing k-nearest neighbors with dynamic time warping and convolutional neural network algorithms. *Sensors* **2020**, *20*, 3879. [CrossRef]
13. Sharma, S.; Gupta, R. On the use of temporal and spectral central moments of forearm surface EMG for finger gesture classification. In Proceedings of the 2018 2nd International Conference on Micro-Electronics and Telecommunication Engineering (ICMETE), Ghaziabad, India, 20–21 September 2018. [CrossRef]
14. Wadhawan, A.; Kumar, P. Deep learning-based sign language recognition system for static signs. *Neural Comput. Appl.* **2020**, *32*, 7957–7968. [CrossRef]
15. Mittal, A.; Kumar, P.; Roy, P.P.; Balasubramanian, R.; Chaudhuri, B.B. A modified LSTM model for continuous sign language recogni- tion using leap motion. *IEEE Sens. J.* **2019**, *19*, 7056–7063. [CrossRef]
16. Elakkiya, R.; Selvamani, K. Subunit sign modeling framework for continuous sign language recognition. *Comput. Electr. Eng.* **2019**, *74*, 379–390. [CrossRef]
17. Gao WE, N.; Wu, J.; Wang, C.; Kumar, S.D.; Kumar, V.H. Sign Language Recognition Based on Hmm/Ann/Dp. *Int. J. Pattern Recognit. Artif. Intell.* **2018**, *9*, 411–417. [CrossRef]
18. Kumar, D.A.; Sastry, A.S.C.S.; Kishore, P.V.V.; Kumar, E.K. Indian sign language recognition using graph matching on 3D motion captured signs. *Multimed. Tools Appl.* **2018**, *77*, 32063–32091. [CrossRef]
19. Huang, J.; Zhou, W.; Li, H.; Li, W. Attention-Based 3D-CNNs for Large-Vocabulary. *IEEE Trans. Circuits Syst. Video Technol.* **2019**, *29*, 2822–2832. [CrossRef]
20. Avola, D.; Bernardi, M.; Cinque, L.; Foresti, G.L.; Massaroni, C. Exploiting Recurrent Neural Networks and Leap Motion Controller for the Recognition of Sign Language and Semaphoric Hand Gestures. *IEEE Trans. Multimed.* **2019**, *21*, 234–285. [CrossRef]
21. Xiao, Q.; Chang, X.; Zhang, X.; Liu, X. Multi-Information Spatial-Temporal LSTM Fusion Continuous Sign Language Neural Machine Translation. *IEEE Access* **2020**, *8*, 216718–216728. [CrossRef]

22. Pu, J.; Zhou, W.; Li, H. Iterative alignment network for continuous sign language recognition. *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.* **2019**, *2019*, 4160–4169. [CrossRef]
23. Tunga, A.; Nuthalapati, S.V.; Wachs, J. Pose-based Sign Language Recognition using GCN and BERT. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, Snowmass, CO, USA, 1–5 March 2020; pp. 31–40.
24. Sharma, S.; Gupta, R.; Kumar, A. Continuous sign language recognition using isolated signs data and deep transfer learning. *J. Ambient Intell. Humaniz. Comput.* **2021**, *1*, 1–12. [CrossRef]
25. Tripathi, K.; Baranwal, N.; Nandi, G.C. Continuous dynamic Indian Sign Language gesture recognition with invariant backgrounds. In Proceedings of the 2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Kochi, India, 10–13 August 2015; pp. 2211–2216. [CrossRef]
26. Tuba, M.; Akashe, S.; Joshi, A. Lecture Notes in Networks and Systems. In *Proceedings of ICT4SD*; Springer: Singapore, 2018; Volume 1. [CrossRef]
27. Rao, G.A.; Kishore, P.V.V. Selfie video based continuous Indian sign language recognition system. *Ain Shams Eng. J.* **2018**, *9*, 1929–1939. [CrossRef]
28. Adithya, V.; Rajesh, R. Hand gestures for emergency situations: A video dataset based on words from Indian sign language. *Data Brief* **2020**, *31*, 106016. [CrossRef]
29. Singh, D.K. 3D-CNN based Dynamic Gesture Recognition for Indian Sign Language Modeling. *Procedia Comput. Sci.* **2021**, *189*, 76–83. [CrossRef]
30. Singh, T.P.; Shreevastava, M. The Study of Detecting Replicate Documents Using MD5 Hash Function. *Int. J. Adv. Comput. Res.* **2011**, *1*, 190–200.
31. Dimitri, G.M.; Spasov, S.; Duggento, A.; Passamonti, L.; Lió, P.; Toschi, N. Multimodal and multicontrast image fusion via deep generative models. *Inf. Fusion* **2022**, *88*, 146–160. [CrossRef]
32. Singla, N. Motion Detection Based on Frame Difference Method. *Int. J. Inf. Comput. Technol.* **2014**, *4*, 1559–1565.
33. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J. Rethinking the Inception Architecture for Computer Vision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 2818–2826.
34. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vis.* **2015**, *115*, 211–252. [CrossRef]
35. Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; IEEE: Piscataway, NJ, USA, 2009; pp. 255–288.
36. Uddin, M.Z.; Kim, J. A Robust Approach for Human Activity Recognition Using 3-D Body Joint Motion Features with Deep Belief Network. *KSII Trans. Internet Inf. Syst.* **2017**, *11*, 1118–1133. [CrossRef]
37. Shen, J.; Liu, N.; Sun, H.; Tao, X.; Li, Q. Vehicle Detection in Aerial Images Based on Hyper Feature Map in Deep Convolutional Network. *KSII Trans. Internet Inf. Syst.* **2019**, *13*, 1989–2011. [CrossRef]
38. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. In Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015. [CrossRef]
39. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
40. Lin, M.; Chen, Q.; Yan, S. Network in network. *arXiv* **2013**, arXiv:1312.4400. [CrossRef]
41. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going Deeper with Convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 1–5 June 2015; pp. 1–9.
42. Abiyev, R.H.; Arslan, M.; Idoko, J.B. Sign language translation using deep convolutional neural networks. *KSII Trans. Internet Inf. Syst.* **2020**, *14*, 631–653. [CrossRef]
43. Hochreiter, S.; Urgen Schmidhuber, J. Long Shortterm Memory. *Neural Comput.* **1997**, *9*, 17351780. [CrossRef]
44. Vijayaprabakaran, K.; Sathiyamurthy, K. Towards activation function search for long short-term model network: A differential evolution based approach. *J. King Saud Univ. Comput. Inf. Sci.* **2020**, *34*, 2637–2650. [CrossRef]
45. Yu, L.; Qu, J.; Gao, F.; Tian, Y. A Novel Hierarchical Algorithm for Bearing Fault Diagnosis Based on Stacked LSTM. *Shock Vib.* **2019**, *2019*, 2756284. [CrossRef]
46. Clevert, D.A.; Unterthiner, T.; Hochreiter, S. Fast and accurate deep network learning by exponential linear units (elus). *arXiv* **2015**, arXiv:1511.07289. [CrossRef]
47. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep learning*; Adaptive Computation and Machine Learning Series; MIT Press: Cambridge, MA, USA, 2017; pp. 321–359.
48. Sridhar, A.; Ganesan, R.G.; Kumar, P.; Khapra, M.I. INCLUDE: A Large Scale Dataset for Indian Sign Language Recognition. In Proceedings of the 28th ACM International Conference on Multimedia, Seattle, WA, USA, 12–16 October 2020; pp. 1366–1375. [CrossRef]
49. Dhanjal, A.S.; Singh, W. An automatic machine translation system for multi-lingual speech to Indian sign language. *Multimed. Tools Appl.* **2022**, *81*, 4283–4321. [CrossRef]

50. Núñez-Marcos, A.; Perez-de-Viñaspre, O.; Labaka, G. A survey on Sign Language machine translation. *Expert Syst. Appl.* **2023**, *213*, 118993. [CrossRef]
51. Papastratis, I.; Dimitropoulos, K.; Daras, P. Continuous sign language recognition through a context-aware generative adversarial network. *Sensors* **2021**, *21*, 2437. [CrossRef]