INDIAN SIGN LANGUAGE RECOGNITION USING CNN WITH SPATIAL PYRAMID AND GLOBAL AVERAGE POOLING

Poornima B V

Research Scholar, JSS Science & Technology University
Mysuru, Karnataka, India
Poornimabv.85@gmail.com

Srinath S

Professor, JSS Science & Technology University
Mysuru, Karnataka, India
srinath@sjce.ac.in

Abstract

Sign language recognition plays a pivotal role in bridging communication gaps for the speech and hearing-impaired community. In this research, we present an innovative approach to enhance the accuracy and effectiveness of Indian sign language recognition through the integration of convolutional neural networks with spatial pyramid and global average pooling layers. The objective of this study is to address the intricate challenges by exploiting multi-scale feature representations and global context information. This architecture incorporates spatial pyramid layer to capture fine-grained spatial information across multiple scales. Additionally, global average pooling layer is introduced to consolidate context-aware features, further improving the model's discriminative power. Experiments are conducted using a standard dataset, encompassing a diverse range of gestures. The results demonstrate a significant improvement in recognition accuracy compared to conventional models. Incorporating spatial pyramid and global average pooling layers enables our model to effectively recognize the gestures, even in challenging scenarios with variations in lighting, complex background, and signer-dependent factors by achieving high accuracy. This research not only contributes to the advancement of ISL recognition technology but also holds promise for practical applications in real-world communication and assistive technologies.

***Keywords*: Sign language recognition (SLR), Convolutional Neural Networks (CNN), Spatial Pyramid Pooling (SPP), Global Average Pooling (GAP), Indian Sign Language (ISL)**

1. Introduction

ISL [1] is a natural language used by the speech and hearing impaired community in India. It is a visual language that uses a combination of hand gestures, facial expressions, and body language to convey meaning. ISL has its own unique grammar and syntax, and it is used to communicate ideas and concepts just like any spoken language. A SLR system is a technology that uses computer vision and machine learning techniques to recognize and interpret sign language gestures. These systems serve as foundational technology for the creation of assistive tools tailored to meet the unique needs of the deaf community. This research aims to develop an efficient ISL recognition system using CNN with SPP and GAP on a complex background dataset. Neural networks [2] have been found to be more effective than traditional machine learning algorithms for complex background datasets. This is because neural networks are better able to learn complex patterns and relationships in the data, which is important for recognizing the subtle differences between different sign language gestures. Neural networks are also able to adapt to new data more easily than traditional machine learning algorithms, which makes them more flexible and better suited for real-world applications. In the context of SLR, CNNs [3] have been harnessed to automatically decipher intricate gestures made by signers, enabling real-time interpretation and translation. However, recognizing sign language gestures presents unique challenges. These challenges include the dynamic nature of signs, varying signers, complex hand configurations, and the necessity to capture both fine-grained spatial details and holistic context. In response to these challenges, this research delves into the enhancement of SLR using a tailored approach that incorporates SPP and GAP layers within the

CNN framework. The integration of SPP facilitates the extraction of multi-scale features, enabling the model to capture intricate spatial details within gestures of varying sizes. Simultaneously, GAP layer provide a mechanism for aggregating global context information. The combination of CNNs with SPP and GAP layers has great potential to improve recognition rate significantly.

Previous research investigation [4], a 1-dimensional CNN was utilized to identify signs in American Sign Language (ASL). The data for this particular study was gathered through the use of wearable gloves equipped with both an accelerometer and a gyroscope. To assess the performance of their system, the researchers conducted evaluations in two distinct categories: general sentences and interrogative sentences. In their work [5], a SLR system that is independent of the signer was introduced. They harnessed the power of deep learning techniques and adopted a CNN architecture to identify static alphabets in ISL. In the research conducted by Wadhawan and Kumar [6], they put forth a deep learning model designed for the recognition of ISL. To build their dataset, they used a web camera to capture a wide range of ISL elements, including digits, alphabets, and words. Their approach involved the utilization of a CNN-based architecture for classifying these gestures. The effectiveness of their model was assessed through performance evaluations with multiple optimization techniques. In a different approach detailed in the reference [7], a computer vision-based technique was introduced to recognize all 26 alphabets in ISL. This method entailed the utilization of machine learning algorithms combined with features extracted through the Histogram of Oriented Gradients (HOG). Notably, this approach displayed enhanced efficiency in both training and testing phases when compared to traditional machine learning techniques. It achieved an average accuracy rate of 80.76%, demonstrating its effectiveness in ISL alphabet recognition. The study [8] introduced a method for recognizing ISL using a five-layer CNN model. They collected their dataset by employing the microsoft kinect sensor and recorded gestures from 12 distinct signers. It's important to note that their method's performance evaluation focused solely on recognizing numbers and alphabets within the ISL. Authors of [9] introduced a vision-based technique for hand gesture recognition. Their method involves two crucial steps: feature extraction and gesture recognition. They utilize grey-scale transformation and edge detection for feature extraction. To recognize gestures, they apply a template-matching algorithm, which subsequently displays the corresponding text on the screen. The evaluation of this approach primarily focused on finger-spelled numeric and alphabet gestures. A deep learning model called G-CNN, enriched with a compact representation scheme, is put forth in this study [10]. Furthermore, two additional architectures, namely VGG11 and VGG-16, were also explored and adapted for the purpose of classifying hand gestures in sign language. What sets the proposed vision-based model apart is its ability to remove user dependency and eliminate the requirement for external hardware equipment. As a result, this model becomes significantly more practical for real-world applications. In the study conducted by [11], they devised a SLR method utilizing CNNs. They tested their approach on a dataset comprising 200 unique sign languages, captured from five distinct angles, and placed in various background environments. Impressively, their model achieved a recognition accuracy of 92.88%. Authors of [12] utilized a transformer network to effectively detect sign language gestures present in videos. Their approach went beyond just manual gestures and also considered non-manual components, such as the orientation of the mouth and eyebrows. To achieve this, they introduced three separate transformer networks: a pose transformer network, a video transformer network, and a multimodal transformer network. Each of these networks was specifically designed to enhance the recognition of signs using diverse neural network architectures.

1.1 Observations made from the literature

- Many of the studies rely on limited datasets collected under specific conditions, such as controlled lighting and backgrounds. This may not accurately represent real-world scenarios, leading to potential challenges when deploying these systems outside of the laboratory.
- Some approaches, like those using microsoft kinect sensors or wearable gloves, rely on specific hardware equipment. This can limit the practicality of the systems as they require additional resources and may not be readily accessible to users.
- Deep learning models, when not properly regularized or when trained on limited data, can be prone to over fitting. This can lead to good performance on training data but poor generalization to unseen data.

1.2 Research gap

Traditional classification methods typically involve image resizing and RGB to gray-scale conversion as pre-processing steps. In contrast, in the majority of CNN-based approaches, resizing of the input image is a standard requirement. CNNs are adept at automatically learning features from raw pixel data [13], enabling tasks like image classification and object detection without the need for handcrafted features. However, one limitation of standard CNNs is their fixed input size requirement, which can be restrictive for images of varying dimensions and aspect ratios. This is where SPP [14] steps in as a crucial enhancement. SPP allows CNNs to accept input

images of arbitrary sizes and aspect ratios, overcoming the constraint of fixed input dimensions. Presenting a distinctive advantage in terms of computational efficiency and preserving spatial information.

1.3 Contribution of the proposed work

The proposed work significantly contributes to ISL recognition by achieving enhanced accuracy through the integration of CNNs with SPP and GAP layers. It addresses challenges in ISL recognition, making the model robust to variations in lighting, background, and signer-dependent factors. What truly sets SPP apart is its remarkable ability for multi-scale feature integration [15]. By pooling features at multiple spatial levels, SPP enables CNNs to capture intricate details, textures, and contextual information, all within a single forward pass. This scalability, adaptability, and robustness to input variations make SPP a powerful companion to CNNs, enhancing their versatility and performance across a wide range of computer vision tasks. It empowers CNNs to perceive objects and scenes comprehensively, from fine-grained local features to broader global context, resulting in more accurate and context-aware visual understanding. The incorporation of multi-scale feature representation and global context awareness not only improves recognition accuracy but also holds promise for practical applications, potentially transforming communication and assistive technologies for the speech and hearing-impaired community.

2. Methodology

2.1 Convolutional Neural Network

Our base model comprises a series of convolutional layers. The architecture is selected based on its proven efficacy in image classification tasks, tailored to the unique requirements of sign language gestures. The initial convolutional layer serves as a feature extractor, employing a 1x1 filter to capture basic features with a stride of 1, ensuring a comprehensive examination of input images. Subsequent convolutional layers leverage 3x3 filters with a stride of 1, promoting the learning of hierarchical spatial representations. The choice of 3x3 filters aligns with standard practices in image processing. Activation function, rectified linear unit (ReLU), is applied after each convolutional layer. ReLU introduces non-linearity, enhancing the model's capacity to capture complex patterns. The model is trained using the Nadam optimizer, known for its efficiency in optimizing complex models. The learning rate, set to 0.001, is fine-tuned through extensive experimentation to achieve optimal convergence and generalization. To prevent overfitting and ensure robust learning, additional techniques such as dropout and batch normalization are employed.

2.2 CNN with SPP

Following the convolutional layers and before fully connected layer, SPP layer is added to capture spatial information at multiple resolutions. The proposed architecture in represented in Fig. 1. The details of the proposed model is shown in Fig. 2.

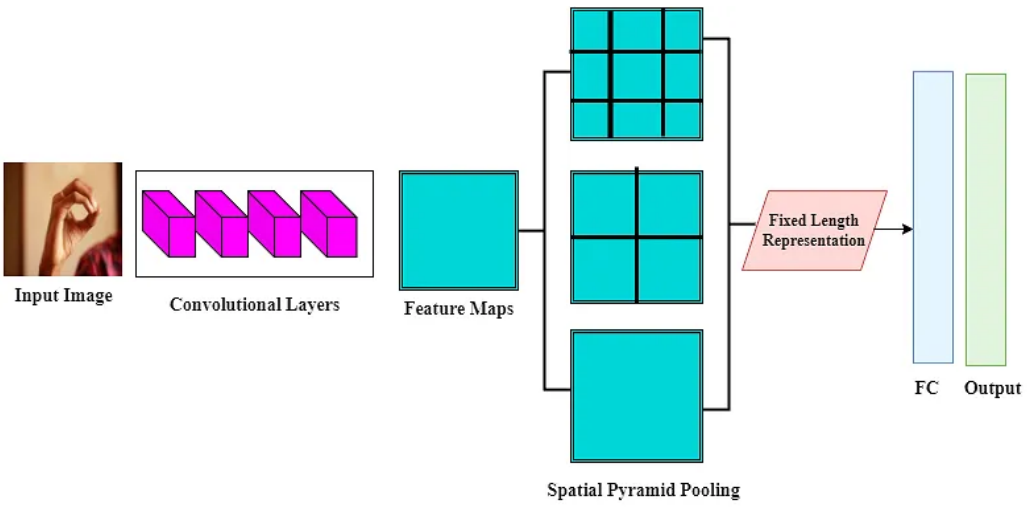

Fig. 1 Proposed CNN+SPP architecture

Layer (type)	Output Shape	Param #
conv2d_14 (Conv2D)	(None, 286, 286, 3)	84
conv2d_15 (Conv2D)	(None, 284, 284, 16)	448
conv2d_16 (Conv2D)	(None, 282, 282, 16)	2320
max_pooling2d_5 (MaxPooling 2D)	(None, 141, 141, 16)	0
conv2d_17 (Conv2D)	(None, 139, 139, 32)	4640
conv2d_18 (Conv2D)	(None, 137, 137, 32)	9248
max_pooling2d_6 (MaxPooling 2D)	(None, 68, 68, 32)	0
conv2d_19 (Conv2D)	(None, 66, 66, 64)	18496
conv2d_20 (Conv2D)	(None, 64, 64, 64)	36928
spatial_pyramid_pooling_1 (SpatialPyramidPooling)	(None, 344064)	0
flatten_2 (Flatten)	(None, 344064)	0
dense_4 (Dense)	(None, 512)	176161280
dense_5 (Dense)	(None, 10)	5130

Total params: 176,238,574
Trainable params: 176,238,574
Non-trainable params: 0

Fig. 2 Description of the proposed model

It performs the multi-scale spatial value integration on the features maps generated by the CNN layers. This layer divides the feature maps into a fixed number of regions of different sizes, capturing both local and global information [16]. The number of regions is determined by the levels of the pyramid. The SPP layer performs max-pooling with different kernel sizes, facilitating multi-scale feature integration. This enables the network to capture features at various levels of granularity within the input. In our work, the SPP layer is configured to perform max-pooling with three different kernel sizes: 1x1, 2x2, and 4x4. These kernel sizes determine the size of the pooling windows used for max-pooling. Each max-pooling operation within the SPP layer will use the specified kernel size to pool features from the input feature maps. To illustrate the work, Fig. 3 shows the three different scales or levels [2, 3, 4] considered. A fixed vector length of 29 is generated after concatenating the bins from the three levels. The scale 2 has 4 bins (2*2), likewise scale 3 and 4 have 9 and 16 bins respectively. All the bins from various scales are concatenated and forms a fixed vector of the size 29 (4+9+16).

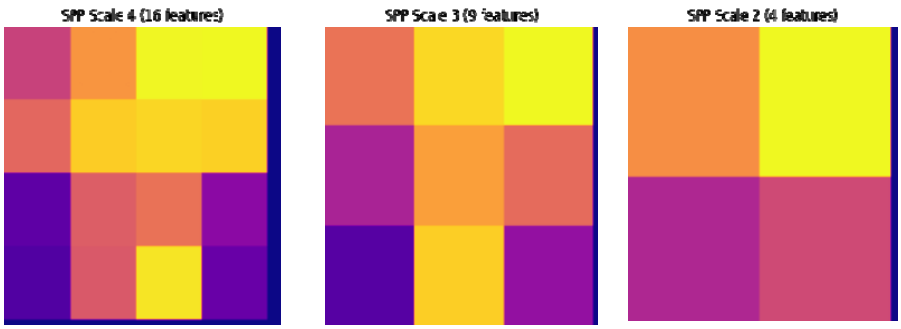

Fig. 3 SPP levels

2.3 Global Average Pooling (GAP)

Proposed work makes use of GAP [17] placed after SPP which make the model more robust and avoids over fitting. After SPP aggregates information at multiple scales, GAP further reduces the spatial dimensions to create a spatially invariant representation. This can be useful because it ensures that the network focuses on the

most salient features from the different spatial scales captured by SPP. Fig. 4 represents the architecture of CNN with SPP & GAP layers. SPP first divides feature maps into grids at various scales, applying independent max-pooling within each grid. The pooled results are concatenated into a fixed-length vector, offering a unified representation across diverse spatial scales. Subsequently, this concatenated vector undergoes GAP, where the average value of each element is computed, resulting in a condensed 1x1 vector for each feature map. This GAP-processed vector serves as input in the subsequent classification stage. The utilization of GAP allows for effective dimensionality reduction and captures essential features for accurate classification.

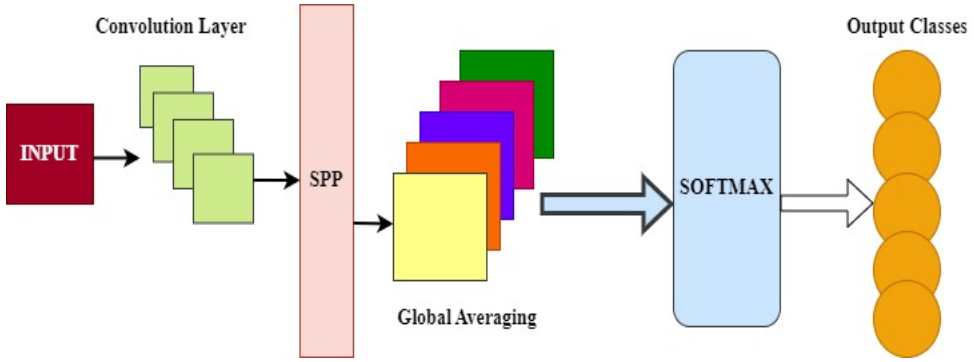

Fig. 4 Proposed CNN with SPP and GAP layer

Steps to perform global averaging:

- Input feature map: A feature map with dimensions H*W*C, where H is the height, W is the width, and C is the number of channels (or features) in the map.
- Pooling operation: For each channel (feature) in the feature map, GAP calculates the average of all the values in that channel. Specifically, it sums up all the values and divides by the total number of values in that channel.
- Output: After applying GAP, a single value is obtained for each channel in the feature map. So, if the process is started with C channels, then we end up with C values, effectively reducing the spatial dimensions of the feature map to 1x1.
  The operation can be represented mathematically as shown in equation (1).
  For each channel C:

$$Output(C) = \frac{(sum\ of\ \ all\ values\ in\ channel\ C)}{(total\ number\ of\ \ values\ in\ channel\ C)} \quad (1)$$

Consider a 2x2 feature map with two channels:
Channel 1:|1.0 2.0|
Channel 2:|3.0 4.0|
For Channel 1:
Output (Channel 1) = (1.0+2.0)/4 = 0.75
For Channel 2:
Output (Channel 2) = (3.0+4.0)/4=1.75

"Channel 1" and "Channel 2" are two separate channels within the feature map. These channels might represent different features or aspects of the input data. GAP would then be applied independently to each of these channels to compute the average value for each channel.

3. Evaluation process

3.1 Dataset details

In this study, the experiments are conducted on the publicly available standard dataset [18]. We have used two datasets i.e. Dataset-I and Dataset-II for the experiment.

3.1.1 Dataset-I

Original dataset as obtained from the public repository without any transformation**.** The images are captured in different background, lighting conditions and cluttered backgrounds. So this dataset can be considered a complex background, non-uniform dataset. Since deep learning models require a large volume of data, we have used 54000 images for our experiment. It consists of 36 classes, representing 26 ISL alphabets and 10 digits (0-9). Samples are shown in Fig. 5.

Fig. 5 Samples of the Dataset-I

3.1.2 Dataset-II (Augmented images)

In our experiment, we assess the model's robustness by utilizing augmented images, specifically applied to Dataset-I. Data augmentation, as described in reference [19], is a technique frequently employed in machine learning and deep learning to expand the dataset's size artificially. The primary objective of data augmentation is to enhance the performance and generalization of machine learning models by introducing diversity to the available data. This diversification offers several advantages, including increased robustness to various input conditions and noise, reduced over fitting risks, and improved generalization, as models trained on augmented data learn to recognize objects or patterns under a range of transformations.
The following transformation operations are done on the images.

- Rotation augmentations involve applying rotations to data samples, either to the right or left along an axis. The effectiveness of this augmentation technique depends on the chosen rotation angle. If the angle is too large, it can distort the labels associated with the data. In our study, we opted for implementing random rotations within the range of -45 degrees to +45 degrees, without cropping the images. This means that for an image I (i, j) in the training dataset, a rotated image IΘ (i, j) will be generated, where Θ is randomly chosen from the interval of -45 to 45 degrees. Fig. 6 displays examples of these rotated images.

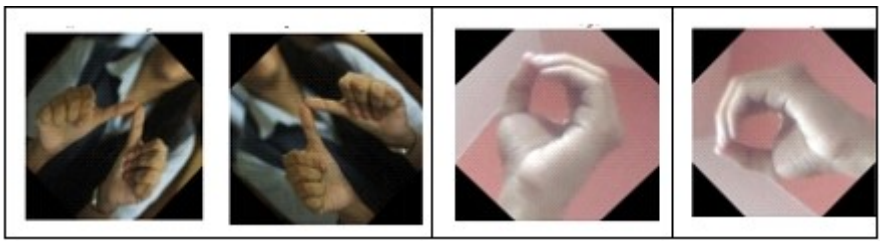

Fig. 6 Samples of the rotated images

- Shearing augmentations involve distorting the data samples by skewing them along one or more axes while keeping the overall shape intact. The appropriateness of this augmentation technique is contingent upon the magnitude of the shear applied, as excessive shear can lead to a loss of label fidelity. In our study, following experimentation and manual inspection of results, we adopted a random shearing approach within the range of (−0.2 to 0.2) along both the x and y axes. In this context, an image I(i, j) from the training dataset undergoes a shearing operation to produce a sheared image IS(i, j). The shearing factors along both the x and y axes are chosen randomly from the range of -0.2 to 0.2. Fig. 7 illustrates some sample images after applying the shearing operation.

Fig. 7 Samples of the sheared images

3.2 Training and Testing

We refer the first model i.e CNN model without integrating SPP & GAP as Model-I, while the second model which incorporates both SPP and GAP layers as Model-II. To maintain consistency, we used identical numbers of layers and filter configurations for both the models. We fine-tuned critical hyper parameters, including the number of training epochs and batch size, through empirical experimentation. One noteworthy advantage of using the Nadam optimizer is its adaptability to learning rates. This optimizer dynamically adjusts learning rates based on a moving window of gradient updates, which allows for continuous improvement in the model's performance over time. During the experimentation phase, we also explored alternative optimizers, but they yielded lower prediction rates. The results and implications of these experiments will be thoroughly discussed in the upcoming section 4 of this paper. The trained model is tested on 20% of the total images comprising of 10800 images to evaluate the performance and 80% training samples i.e. 43200.

4. Results and discussion

The experiment is conducted using Keras version 2.2.4 and TensorFlow version 2.12. Computations are performed on google colab platform. The subsequent sections gives the details about the reacogntion rate obtained.

4.1 Evaluation Metrics

Accuracy is a critical performance measure, representing the ratio of correctly predicted observations to the total number of observations. In terms of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN), the formula for accuracy can be expressed as follows:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

In terms of performance metrics, precision is the ratio of correctly predicted positive observations to the total positive observations. Recall, on the other hand, is the ratio of correctly predicted positive labels to the total number of positive labels. The F1 score represents the weighted average of both precision and recall.

$$\Pr ecision = \frac{TP}{TP + FP}$$

$$\mathrm{Re}call = \frac{TP}{TP + FN}$$

$$F1Score = 2 * \frac{\Pr ecision * \Re call}{\Pr ecision + \Re call}$$

4.2 Recognition rate obtained

4.2.1 Results obtained with MODEL-I on Dataset I & II

This section is dedicated to present the results obtained from the proposed Model-I. For Dataset-I, our model achieved an impressive accuracy of 97.03% with a corresponding loss of 0.22561. It's noteworthy that the recognition accuracy consistently improved as the number of training epochs increased and reached a stable point around the 10th iteration. Similarly, both the training and validation losses exhibited a stabilizing trend by the 10th iteration. In the case of Dataset-II, we observed that the accuracy stabilized at 95.01%, with a corresponding loss of 0.3046, which occurred around the 25th iteration. It's important to note that the reduction in recognition accuracy in Dataset-II can be attributed to the increased intra-class variance present in this dataset. Figures 8 and 9 display the ROC curves representing the training and validation accuracy and loss for Dataset-I respectively, while Figures 10 and 11 show the corresponding ROC curves for Dataset-II.

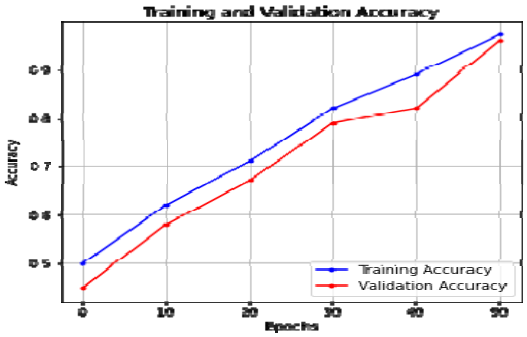

Fig. 8 Training & Validation accuracy on Dataset-I

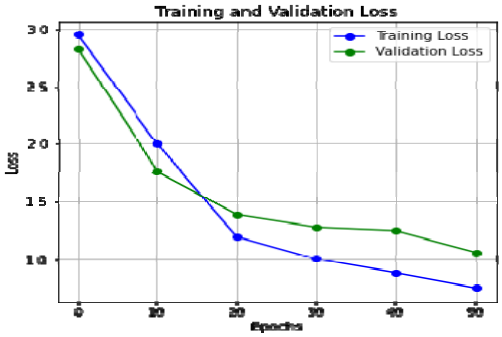

Fig. 9 Training & Validation loss on Dataset-I

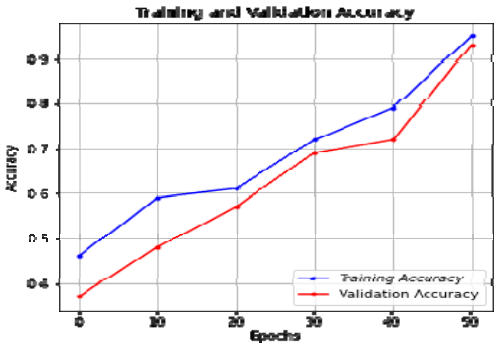

Fig. 10 Training & Validation accuracy on Dataset-II

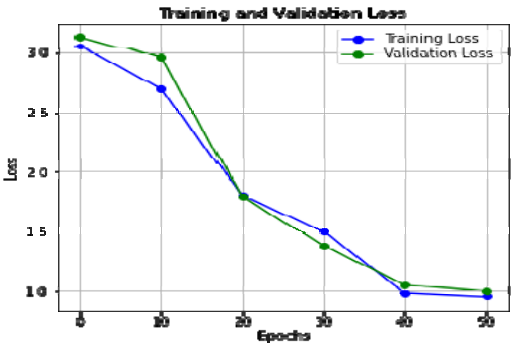

Fig. 11 Training & Validation loss Dataset-II

4.2.2 Results obtained with MODEL-II on Dataset I & II

This section presents the results obtained from our Model-II. In the case of Dataset-I, Model-II achieved an outstanding accuracy of 99.02%, along with a loss of 0.11829. It's worth noting that the recognition accuracy consistently improved as the number of training epochs increased. Turning our attention to Dataset-II, Model-II achieved a remarkable accuracy of 96.67% and a corresponding loss of 0.14571. Despite the challenges posed by increased intra-class variance in Dataset-II. Figures 12 and 13 display the ROC curves representing the training and validation accuracy and loss for Dataset-I, while Figures 14 and 15 depict the corresponding ROC curves for Dataset-II. These results highlight the impressive performance of Model-II in our study

Fig. 12 Training & Validation accuracy on Dataset-I

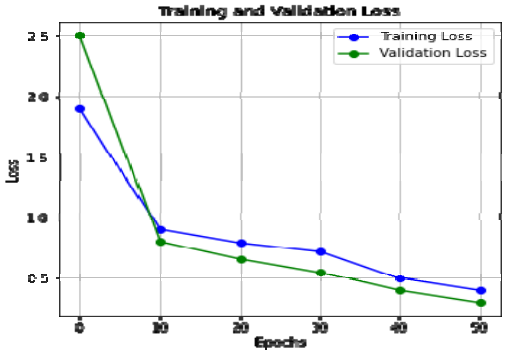

Fig. 13 Training & Validation loss on Dataset-I

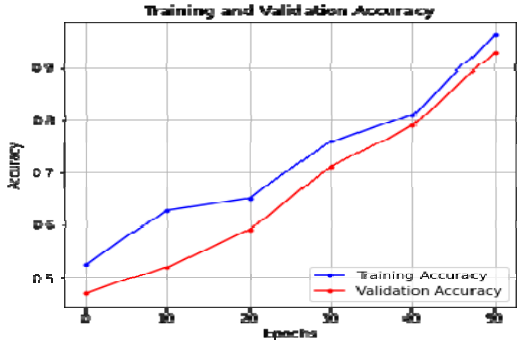

Fig. 14 Training & Validation accuracy on Dataset-II

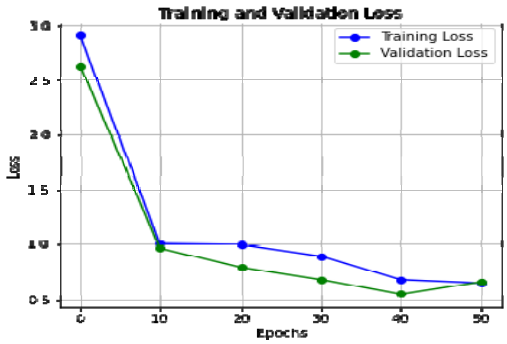

Fig. 15 Training & Validation loss on Dataset-II

The performance evaluation metrics are represented in Table 1 & 2. In addition, Table 3 and Table 4 present class-wise accuracy scores obtained for Dataset-I and Dataset-II respectively, further enhancing our understanding of the model's performance at the individual class level.

	S NO.	Dataset	Precision	Recall	F Score
MODEL-I	1	Dataset-I	0.92	0.95	0.90
	2	Dataset-II	0.86	0.89	0.94

Table 1 Performance metrics table for Model-I

	S NO.	Dataset	Precision	Recall	F Score
MODEL-II	1	Dataset-I	0.94	0.98	0.96
	2	Dataset-II	0.89	0.93	0.92

Table 2 Performance metrics table for Model-II

ISL	MODEL-I	MODEL-II	ISL	MODEL-I	MODEL-II	ISL	MODEL-I	MODEL-II
0	99	100	C	99	100	O	96	97
1	100	100	D	97	99	P	97	97
2	98	99	E	96	97	Q	93	96
3	98	100	F	98	98	R	95	96
4	99	99	G	95	96	S	96	98
5	97	98	H	96	97	T	93	95
6	97	98	I	99	97	U	97	98
7	98	99	J	94	95	V	98	99
8	96	97	K	95	96	W	95	99
9	96	99	L	96	97	X	94	98
A	99	99	M	93	94	Y	96	100
B	98	99	N	94	97	Z	92	99

Table 3 Class wise accuracy obtained for Dataset-I

ISL	MODEL-I	MODEL-II	ISL	MODEL-I	MODEL-II	ISL	MODEL-I	MODEL-II
0	96	98	C	94	100	O	95	94
1	100	100	D	95	100	P	96	96
2	96	97	E	95	95	Q	90	92
3	98	99	F	96	96	R	89	91
4	97	99	G	95	95	S	86	92
5	96	97	H	92	96	T	91	91
6	97	98	I	94	94	U	93	99
7	96	97	J	91	95	V	94	100
8	94	96	K	95	95	W	94	96
9	94	96	L	93	94	X	92	94
A	95	96	M	95	95	Y	93	100
B	92	94	N	96	95	Z	90	100

Table 4 Class wise accuracy obtained for Dataset-II

4.2.3 Experiment results with respect to optimizers

Optimizers play a crucial role in fine-tuning a model's parameters or weights, with the ultimate aim of reducing the loss function and improving the accuracy of predictions. We conducted experiments using various optimization algorithms, including adaptive moment estimation (Adam) [20], stochastic gradient descent (SGD) [20], and Nadam [21], to evaluate their impact on the model's performance. Table 5 shows the accuracy and loss values obtained by various optimizers. Nadam stands for "Nesterov-accelerated adaptive moment estimation". It is an optimization algorithm that combines the Nesterov accelerated gradient descent (NAG) and Adaptive moment estimation (Adam) techniques to improve convergence during training neural networks. It's important to highlight that Nadam demonstrated strong performance across both the models. The presented results showcase the average accuracy achieved. The results clearly indicate that Nadam has demonstrated superior performance compared to the other two optimizers.

S No.	Training accuracy	Training loss	Validation accuracy	Validation loss	Optimizer
1	89.17	0.1080	88.80	0.1684	Adam
2	93.72	0.674	92.56	0.759	SGD
3	99.08	0.0676	99.25	0.0776	Nadam

Table 5 Experimental results obtained with various optimizers

4.3 Prediction on unseen data

To evaluate the trained model on unseen data, we saved the model as 'ISLCOMPLEX.h5' and subsequently loaded it for inference on previously unseen data. The Fig. 16 demonstrates the successful performance of the proposed model when applied to previously unseen data. Specifically, our model correctly predicted class labels 3 and 7 for the given images. To ensure the robustness and reliability of our model, we conducted random checks on additional images, and the accuracy of the predictions remained consistent. These results underscore the effectiveness and generalization capability of our proposed approach in image classification tasks.

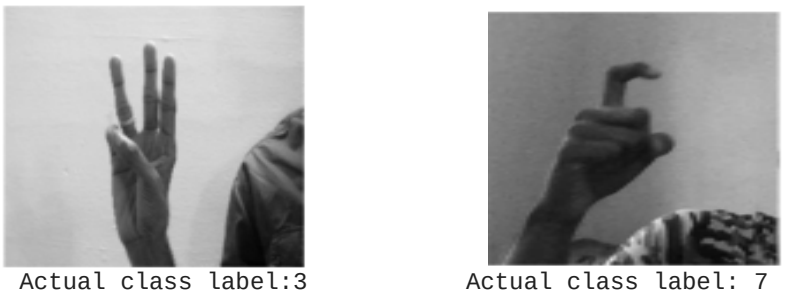

Fig. 16 Predicted results on unseen data

4.4 Comparison with existing methods

Table 6 presents the comparative analysis of existing methods with the proposed models in terms of the number of classes used for the experimentation and the accuracy obtained.

Authors	Year	Classes	Accuracy
Athira et al. [22]	2019	24	90%
Kumar et al.[23]	2021	26	80.76%
Sakshi sharma et al. [24]	2021	26	92.443%
Rachana Patil et al. [25]	2021	10	95%
Nurzada Amangelde et al.[26]	2022	41	97%
Deep Kothadiya et al.[27]	2022	11	97%
Proposed		***36***	***Dataset-I-99%*** ***Dataset-II-96%***

Table 6 Proposed vs. Existing approaches

Conclusion

The proposed research work presents a significant advancement in the field of ISL recognition. It tackles essential challenges that the speech and hearing-impaired community encounter in their communication. By combining CNNs with SPP and GAP layers, we have developed a robust and innovative approach that excels in recognizing 36 ISL gestures which includes alphabets and digits (0-9). Our model demonstrated exceptional performance with a recognition accuracy of 99% when evaluated on Dataset-I, which comprises a diverse set of ISL gestures in the presence of complex background scenarios. Even with the introduction of increased data variability through data augmentation in Dataset-II, our approach maintained a robust accuracy rate of 96%, surpassing the challenges posed by the extended version of the standard dataset. In contrast, utilizing CNN alone, without the incorporation of SPP and GAP layers, yielded less accuracy. CNN-SPP, which integrates spatial pyramid layers to capture fine-grained spatial information across multiple scales, and GAP, which consolidates feature representations, are crucial components of our model. These innovations enhance the model's discriminative power and its ability to effectively recognize ISL gestures.

Conflict of Interest

The authors state that there is no conflict of interest to declare and the research is not funded by any source.

References

[1] A. K. Sahoo, “Indian Sign Language Recognition Using Machine Learning Techniques,” Macromol. Symp., vol. 397, no. 1, pp. 1–7, 2021.
[2] K. Bantupalli and Y. Xie, “American Sign Language Recognition using Deep Learning and Computer Vision,” Proc. 2018 IEEE Int. Conf. Big Data, Big Data 2018, pp. 4896–4899, Jan. 2019.
[3] G. Pala, J. B. Jethwani, S. S. Kumbhar, and S. D. Patil, “Machine Learning-based Hand Sign Recognition,” Proc. - Int. Conf. Artif. Intell. Smart Syst. ICAIS 2021, pp. 356–363, 2021.
[4] Suri, K., & Gupta, R. (2019). Convolutional neural network array for sign language recognition using wearable IMUs. In: 2019 6th International Conference on Signal Processing and Integrated Networks (SPIN), (pp. 483–488).
[5] Sruthi, C. J., & Lijiya, A. (2019). Signet: A deep learning-based Indian sign language recognition system. In: 2019 International Conference on Communication and Signal Processing (ICCSP), (pp. 596–600).
[6] Wadhawan, A., & Kumar, P. (2020). Deep learning-based sign language recognition system for static signs. Neural Computing and Applications, 1–12.
[7] Kumar, A., & Kumar, R. (2021). A novel approach for ISL alphabet recognition using Extreme Learning Machine. International Journal of Information Technology, 13(1), 349–357.
[8] Gangrade, J., & Bharti, J. (2020). Vision-based hand gesture recognition for Indian sign language using convolution neural network. IETE Journal of Research, 1–10
[9] Shrenika, S., & Bala, M. M. IEEE SIGN LANGUAGE RECOGNITION USING TEMPLATE MATCHING TECHNIQUE.
[10] Sharma, S., & Singh, S. (2021). Vision-based hand gesture recognition using deep learning for the interpretation of sign language. Expert Systems with Applications, 182.
[11] G. A. Rao, K. Syamala, P. V. V. Kishore and A. S. C. S. Sastry, "Deep convolutional neural networks for sign language recognition," 2018 Conference on Signal Processing and Communication Engineering Systems (SPACES), Vijayawada, 2018, pp. 194-197.
[12] De Coster, M.; Herreweghe, M.V.; Dambre, J. Sign Language Recognition with Transformer Networks. In Proceedings of the Conference on Language Resources and Evaluation (LREC 2020), Marseille, France, 13–15 May 2020; pp. 6018–6024.
[13] Dudhal, A., Mathkar, H., Jain, A., Kadam, O., & Shirole, M. (2018). Approach for Indian Sign Language Recognition System Based on CNN (Vol. 2018). Springer International Publishing.
[14] Tai, S. K., Dewi, C., Chen, R. C., Liu, Y. T., Jiang, X., & Yu, H. (2020). Deep learning for traffic sign recognition based on spatial pyramid pooling with scale analysis. Applied Sciences (Switzerland), 10(19), 1–16.

[15] Lim, L. A., & Yalim Keles, H. (2018). Foreground segmentation using convolutional neural networks for multiscale feature encoding. Pattern Recognition Letters, 112, 256–262.
[16] Ravi, S., Maloji, S., Polurie, V. V. K., & Eepuri, K. K. (2018). Sign language recognition with multi feature fusion and ANN classifier. Turkish Journal of Electrical Engineering and Computer Sciences, 26(6), 2871–2885. https://doi.org/10.3906/elk-1711-139.
[17] Subburaj, S., & Murugavalli, S. (2022). Survey on sign language recognition in context of vision-based and deep learning. Measurement: Sensors, 23.
[18] https://github.com/sajanraj/Indian-Sign-Language-Recognition.
[19] Sharma, P., & Anand, R. S. (2021). A comprehensive evaluation of deep models and optimizers for Indian sign language recognition. Graphics and Visual Computing, 5, 200032.
[20] https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6
[21] https://machinelearningmastery.com/gradient-descent-optimization-with-nadam-from-scratch/
[22] Athira, P. K., Sruthi, C. J., & Lijiya, A. (2019). A signer independent sign language recognition with co-articulation elimination from live videos: An indian scenario. Journal of King Saud Uni- versity-Computer and Information Sciences.
[23] Kumar, A., & Kumar, R. (2021). A novel approach for ISL alphabet recognition using Extreme Learning Machine. International Journal of Information Technology, 13(1), 349–357.
[24] Sharma, S., & Singh, S. (2022). Recognition of Indian Sign Language (ISL) Using Deep Learning Model. Wireless Personal Communications, 123(1), 671–692.
[25] Patil, R., Patil, V., Bahuguna, A., & Gaurav Datkhile, M. (n.d.). Indian Sign Language Recognition using Convolutional Neural Network.
[26] Ma, Y., Xu, T., & Kim, K. (2022). Two-Stream Mixed Convolutional Neural Network for American Sign Language Recognition. Sensors, 22(16).
[27] Kothadiya, D., Bhatt, C., Sapariya, K., Patel, K., Gil-González, A. B., & Corchado, J. M. (2022). Deepsign: Sign Language Detection and Recognition Using Deep Learning. Electronics (Switzerland), 11(11), 1–12.

Authors Profile

Poornima B V
Research Scholar
Completed BE from Adichunchangiri Institute of Technology, Chikmaglore and obtained MTech in CSE from Srinivas Institute of technology, Mangaluru. She has a teaching experience of 11years. Currently pursuing full-time Ph.D at JSSSTU, Mysuru. Area of research is Machine learning and research interests include Digital image processing, Machine learning, pattern recognition and deep learning.

Dr. Srinath S
Professor, BE, MTech, Ph.D
Obtained BE in Computer science & Engineering from SIT, Tumkur, secured 1st rank in MTech under VTU in 2002 and received Ph.D from Mysore university in 2015. He is an Associate professor in the Dept. of CSE, SJCE, JSSSTU, Mysuru. He has around 28 years of teaching experience, he has guided several Ph.D scholars and has more than 40 publications. His area of research is Digital Image processing and subject expertise in Machine learning, deep learning, pattern recognition.