Ingénierie des Systèmes d'Information
Vol. 30, No. 12, December, 2025, pp. 3287-3300
Journal homepage: http://iieta.org/journals/isi

Adaptive Graph Convolution Deep Learning Model for Multilingual Hand Sign Recognition with Texture and Frequency-Based Features

Ishraq Abdul Alameer[1*], Nidaa A. Abbas[2], Mehdi Ebady Manaa[3,4]

[1] Department of Computer Science, Faculty of Women Science, University of Babylon, Babylon 51002, Iraq
[2] Department of Software, Faculty of Information Technology, University of Babylon, Babylon 51002, Iraq
[3] Department of Information Networks, College of Information Technology, University of Babylon, Hilla 51002, Iraq
[4] Intelligent Medical Systems Department, College of Sciences, Al-Mustaqbal University, Hilla 51001, Iraq

Corresponding Author Email: ishraq.abdalmer@uobabylon.edu.iq

https://doi.org/10.18280/isi.301221

Received: 22 August 2025
Revised: 5 November 2025
Accepted: 25 November 2025
Available online: 31 December 2025

Keywords:
hand sign recognition, K-Nearest Neighbors, Graph Convolutional Network, multimodal feature fusion, texture features, American Sign Language, frequency-domain analysis, Arabic Sign Language

ABSTRACT

Sign language recognition facilitates direct communication between a deaf or hard-of-hearing person and individuals who are not familiar with sign language. This paper proposes a multilingual recognition framework based on an integration of Graph Convolutional Networks (GCNs), standard Convolutional Neural Networks (CNNs), and recurrent models (Gated Recurrent Units (GRU) & Long Short-Term Memory (LSTM)) to capture structural, spatial as well as temporal information. During preprocessing the images are converted into grayscale; enhanced using histogram equalization and Gaussian filtering; resized-and normalized so as to improve visual consistency across them. The system then extracts complementary descriptors by Principal Component Analysis (PCA), Fast Fourier Transform (FFT) and Tamura texture features, providing statistical cue, frequency domain cue and perceptual texture cue respectively. These feature vectors build up a K-Nearest Neighbors (k-NN) graph where each node connects its most similar neighbors in terms of Euclidean distance forming adjacency matrix encoding local similarity patterns. The resultant graph structure allows efficient propagation and aggregation of information inside Graph Convolutional Network (GCN) layers strengthening discriminative representation towards classification. The proposed approach is evaluated on Arabic Sign Language (ArSL) and American Sign Language (ASL) benchmark datasets, where it achieves 97% and 100% accuracies, respectively. Results show that a combination of graph-based learning with CNN and recurrent modeling makes recognition more robust while the modular design gives a scalable base to extend the framework for other sign languages.

1. INTRODUCTION

Sign language is a fully developed natural human language using hand shapes, movements and orientations together with facial expressions and body postures. It has its own grammar and vocabulary, independent of spoken languages, but is used for the same general communicative purposes [1].

Each sign language arises from its cultural-linguistic community embodying particular historical social influences. Sign languages have two big parts: manual and non-manual signs. The manual sign has different components such as the position, orientation, shape, and movement of the hand or hands while the non-manual sign is about body movement most especially facial expression. Non-manual signs are important in specifying and emphasizing meanings carried by manual gestures but since they provide core information content of a message, most studies focus on manuals [2].

Sign Language Recognition (SLR) is a branch of research aimed at the accurate interpretation of visual signing and its translation into spoken or written language. This technology helps reduce communication barriers for people who are deaf or hard of hearing, between them and those who do not know sign language [3].

Most current SLR methods fall into one of two broad camps: image-based or sensor-based. Image-based solutions have lately become more popular for their ease of access, friendly use, and low hardware requisites. Applying computer vision algorithms to recognize and track hand gestures, such systems do not need any wearable apparatus; they can simply be implemented on any platform with a built-in camera. The proliferation of high-quality cameras in smartphones, tablets, and laptops has made it easier to bring image-based SLR into every day on-demand communication applications [4].

The basic motivation of sign language is in its function as the natural medium of human interaction allowing a deaf person to communicate self-expression involving communication with other people and community participation. Sign language, therefore, as the tool for social inclusion, enables actual communication at educational, work, or social situations. Its recognition as a complete linguistic system develops steadily along with international movements widening its visibility and usage [1].

The main objective of this paper is to propose a generalized lightweight deep learning model that can be tuned to recognize

any static sign language dataset. The rest of the paper is organized as follows: Section 2 provides related works. Section 3 describes the main theoretical concepts of this study. The proposed model for this study is presented in Section 4. Implementation details and experimental results are discussed in Section 5. Finally, all the conclusions from this study are explained in Section 6.

2. RELATED WORK

Many works have been carried out in the area of sign language recognition due to this perpetual quest for an effective and efficient method of recognizing hand gestures. However, opportunities still abound in delivering results more accurate and robust than the existing approaches can muster. Recent research proposals attempt new techniques to clearly detect and analyze complex hand movements so that a two-way communication system involving somebody who is hard-of-hearing or deaf on one end but does not know sign language on the other can understand them. The related works are classified into three main categories for systematic review:

2.1 Studies related to the preprocessing stage

These works focus on the initial stages of developing grayscale and normalized images to enhance quality and maintain consistency. Common techniques include converting the image into grayscale, applying histogram equalization, Gaussian Blur, resizing the image as per requirement, and Normalization.

The main objective of preprocessing is to normalize abnormal input conditions and eliminate irrelevant variations that may badly affect learning stages later on. Several related papers to this stage have been analyzed for this purpose.

Ahmed et al. [2] developed an American Sign Language static image recognizer of 29 classes in the year 2025. The pre-processing steps on each frame are grayscale conversion, resizing to 224 × 224 pixels, normalization of the range 0–255 to [0,1], and noise removal by Gaussian/median filters. Three Tamura texture (descriptors- coarseness, contrast, directionality) are fused with raw image data at input level and fed into a pruned ResNet-50 backbone. Sequential Feature Selection further refines these features while tuning Generalized Additive Model (GAM) classifier using tenfold cross validation scheme. Average accuracy over all letters is about 96.7%.

Authors emphasize that this shallow Tamura–ResNet50–GAM hybrid comes close to YOLOv3-like deep detectors yet low resource hungry, however real time deployment plus generalization outside ASL dataset remain open issues.

In 2024, Abd Al-Latief et al. [3] first converted the RGB frames into grayscale, enhances the local contrast through histogram equalization, reduces noise, and then applies a contour-based segmentation method to crop out the hand region; finally resizes the cropped region to 50×50 for classification.

It thereafter applies WAR-Strategy meta-heuristic that prunes features and fine-tunes six classical ML classifiers-in achieving accuracies of 93.11–100% on American, Arabic, and Malaysian static-image datasets with training times reduced to 0.038–10 s across these dataset languages-with sub second inference on some models as clear strengths but is still limited to isolated frames with conventional ML pipeline scalability ceiling inherited.

2.2 Studies related to the feature extraction stage

Research in this category aims to identify and extract the most discriminative features from hand gestures, either through handcrafted descriptors or learned representations. There are many previous studies within this stage, the feature extraction stage.

In 2023, Park et al. [4] employed a combination of 2D-FFTand convolutional neural networks (CNNs) to address complex hand gesture input and noise caused by the external environment. 2D FFT have been used to convert time data (image) into frequency domain then applied normalization and resizing. The method's reliance on specialized radar hardware is an additional cost and a five-word vocabulary limits broader sign-language applicability.

In 2023, Ahmed [5] introduces a hybrid Tamura–SIFT pipeline for texture features. The Tamura descriptors provide perceptual global cues (coarseness, contrast, directionality), and SIFT injects scale- and rotation-invariant key-points that represent fine local structure. Feature-level concatenation forms a composite vector that carries global statistics united with local gradients. It beats every single descriptor under changing illumination, scale, and view by a good margin in standard benchmarks. The richer signature improves retrieval, segmentation, and object-recognition accuracy across multiple datasets. The major limitation of the method is its high computational cost which restrains real-time deployment.

2.3 Studies related to the classification stage

This group covers studies that design and evaluate models for mapping gesture inputs to particular sign language classes. Models run the gamut from baseline SVMs and k-NNs up to state-of-the-art deep learning models, including CNNs and even Graph Neural Networks (GCNs). A couple of papers also venture hybrid models toward accuracy and generalization enhancement.

In 2024, Miah et al. [6] proposed a two-stream GCAR model in which an effective combination of spatial and temporal information is achieved through GCN, sep-TCN, and channel attention. The architecture produces very high accuracies on big largescale datasets such as WLASL and PSL because it represents full-body dynamics and was designed to effectively manage joint discontinuity drawbacks. However, results remain modest for ASLLVD and this cannot be applied in real-time due to the complexity of the model.

In that year, a three-layer GCN with successive residual connections applied to 21-landmark hand graphs attains 99.1% accuracy on the ASL-Alphabet dataset. The model is lightweight, enjoys stable gradients, and resists over-fitting, but it speaks only to static letters, ignores temporal dynamics, and generalizes poorly outside of America where the landmark detector typically does not work [7].

In 2023, Vasudevan et al. [8] introduced WaveMesh superpixels and WavePool pooling as inputs to SplineCNNs that outperform SLIC on MNIST, Fashion-MNIST, and CIFAR-10. The method has strengths in adaptive multiscale graph construction but suffers from computationally expensive wavelet preprocessing per image, requires tuning for each image, and has not been tested on skeleton-based sign-language data.

In 2022, Han et al. [9] proposed Vision GNN (ViG) as an

image backbone representing each patch as a graph node and by alternating Grapher and FeedForward blocks to overcome the over-smoothing problem. ViG-S achieves 82.1 % Top-1 on ImageNet. Its strength lies in scalable patch-level graph reasoning that flexibly models spatial context. However, the use of fixed k-NN graph construction and the computational overhead of graph building, along with larger model variants, may limit its suitability for real-time sign-language recognition, which requires fine-grained hand-joint modeling and temporal information.

3. THE MAIN THEORETICAL CONCEPT

Much of what follows is a definition of the key terms that will enable the avoidance of ambiguity in describing the research methodology and thus establish the level of robustness that can be attained, besides helping others repeat the work. By specifying how each abstract concept is measured, fair comparison with other research becomes possible. This upfront clarity helps to lay a solid groundwork for later stages—for example, data preparation, model design, and performance evaluation—so that every step can be transparently examined, critiqued, and improved upon in further investigations.

3.1 Datasets

Two global datasets, American Sign Language and the Arabic Sign Language (ArSL) downloaded from Kaggle and used in the experimental. Each dataset is divided into training and testing.

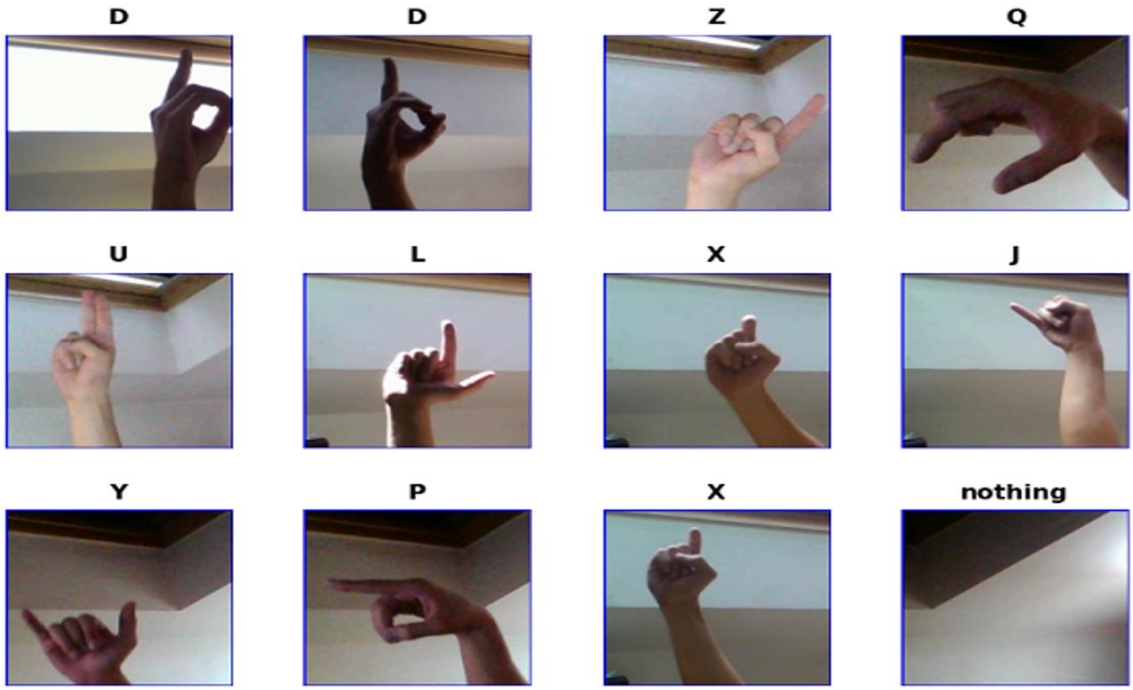

Figure 1. Random sample of American Sign Language (ASL) dataset

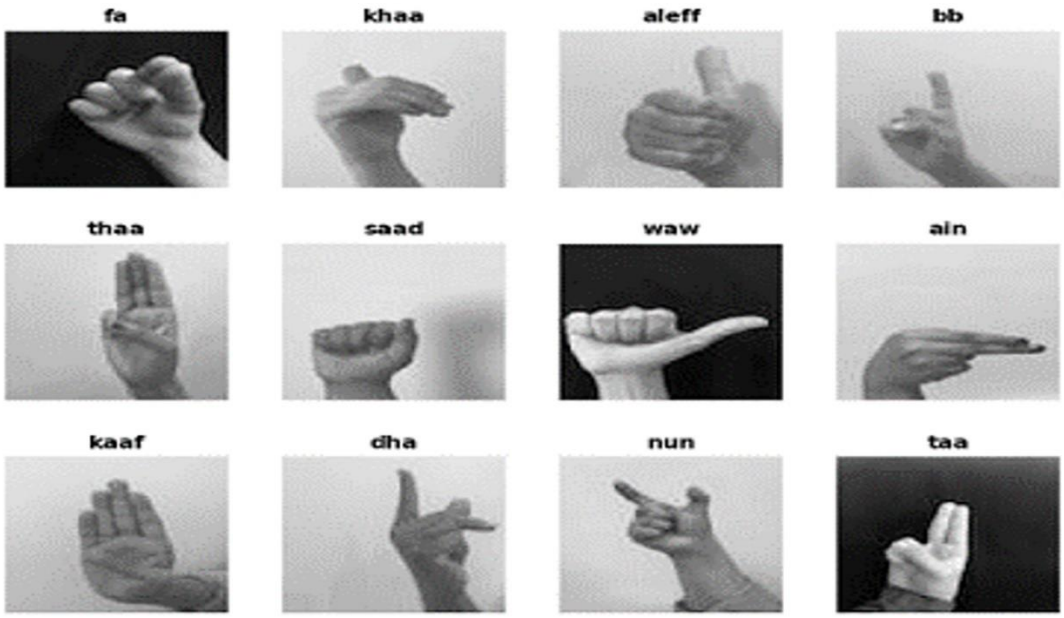

Figure 2. Random sample of the Arabic Sign Language (ArSL) dataset

3.1.1 American Sign Language dataset

The ASL dataset which has been used in the study has been taken from the Kaggle repository and is available at https://www.kaggle.com/datasets/asl-alphabet/asl_alphabet_train. The dataset has been divided into training and testing subsets. Particularly, the data for training contains 29 different classes or labels corresponding to 26 alphabetical letters from A through Z plus three extra categories labeled as "Space," "Delete," and "Nothing." Figure 1 provides an example of this data that clearly shows what kind of images there are and how diverse they can be.

3.1.2 Arabic Sign Language

The ArSL has been sourced from Kaggle at https://www.kaggle.com/datasets/cherryshad0/arasl-database-54k-final and then further divided into the training and testing sets. It contains 54,049 images that belong to 28 different classes, where every class signifies a letter of the Arabic alphabet. Because generalization requires variation in hand shape as well as orientation, images have been taken from 40 signers. An example excerpted in Figure 2 demonstrates both manifoldness and features of the dataset.

3.2 Preprocessing concepts

3.2.1 Grayscale conversion

Grayscale takes away color information and keeps the strength of light at every pixel. It makes it efficient and less complex so as to allow analysis based on texture and intensity.

$$I_{gray}(x,y) = 0.299 * R(x,y) + 0.587 * G(x,y) + 0.114 * B(x,y) \quad (1)$$

This formula reflects the way brightness is perceived by the human eye. In grayscale, data is reduced to a fair amount and content structure of the image comes out which is usually more important for pattern recognition than color [3].

3.2.2 Histogram equalization

Histogram equalization is a method of readjusting pixel intensities such that contrast can be improved. It helps in the visibility of features by ensuring that pixel intensities are well spread within the available intensity range. It helps to a great extent in improving the clarity of the image, thus facilitating better feature extraction and classification accuracy.

$$s_k = (L-1) = \sum_{j=0}^{k} p_r(r_j) \quad (2)$$

where,

- s_k: The newly equalized intensity value.
- L: The number of total intensity levels.
- $p_r(r_j)$: Probability of intensity level r_j occurring.

This becomes particularly handy when the image is too pale or lacks enough brightness. It redistributes the values of intensity, hence more details on texture and edges which become visible to the model [10].

3.2.3 Gaussian blur

Gaussian blur helps smooth out image noise using a weighted average around each pixel.

$$G(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \quad (3)$$

where,

- $G(x,y)$: The value of the Gaussian kernel at point (x,y).
- σ: Standard deviation of the Gaussian distribution.

By softening edges and reducing small pixel variations, it allows the model to focus on significant patterns instead of reacting to random noise. It provides smoother, cleaner images, aiding in accurate detection of meaningful features [11].

3.2.4 Image resizing

Images were equally adjusted to one size by interpolation. It keeps all the images at one standard, fulfilling the size requirement of analytical methods and neural networks as input. It assures consistency, making direct comparison between images possible and feature extraction steady [3].

3.3 Features extraction methods

3.3.1 Principal Component Analysis

PCA is a statistical technique increasingly popular for applications such as dimensionality reduction, visualization, noise filtering, and decorrelation. Therefore, in the context of this study, PCA has been used as a feature extraction method that will output DE correlated, energy ordered coefficients (scores) carrying the most discriminative variance. These coefficients will now be considered extracted features to be used in classification rather than reducing dimensions [12].

$$Y = W^T * X \quad (4)$$

where,

- X: represents the original high-dimensional image data.
- W : is the matrix of principal components (eigenvectors).
- Y : represents the transformed, lower-dimensional features.

PCA finds the principal directions in the data where variance is high. It does not emphasize dimensionality reduction but rather feature extraction, such that the most significant variance is preserved. By choosing these top components, a small feature set is obtained which has high representation and hence improves both efficiency and accuracy in the classification process.

3.3.2 Fast Fourier Transform

FFT takes an image from the spatial domain to the frequency domain.

$$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)} \quad (5)$$

where,

- $f(x,y)$: Intensity at pixel (x,y).
- $F(u,v)$: Frequency domain representation at (u,v).
- M,N : Width and height of the image.

This change exposes recurring patterns and structural regularities that were not visible in the original pixel arrangement. A lot of textures have special frequency marks, and FFT does a great job picking up those differences, giving the model extra ways to spot texture-based differences. It gets frequency-based features, offering views into structural and textural qualities not clear in spatial areas. It boosts feature discrimination abilities, helping in getting classification right

[4].

3.3.3 Tamura features

Tamura features are conceived as classical, manual texture descriptors, from perceptual principles of human vision. They were designed to approximate the psychological dimensions by which humans perceive differences in texture and thus encompass both low-level statistical variation and high-level structural properties of images. Tamura features go beyond purely statistical measures in the sense that they emphasize textural perception dimensions—granularity, regularity, and orientation—for image analysis applications such as content-based image retrieval, classification, and recognition. Standard six Tamura features are Coarseness, Contrast, Directionality, and Line-likeness Regularity [13].

a. Coarseness (Granularity Measure)

Coarseness reflects the size of texture primitives (granules). An image can be said to contain fine-grained or large-grained patterns. Mathematically, coarseness is obtained as a multi-scale average of intensities for the image. For each pixel(x,y) give the final coarseness measure as an average over the entire image:

$$F_{croaseness} = \frac{1}{N}\sum_{x,y} 2^{k(x,y)} \tag{6}$$

where, N is the number of pixels. A higher value indicates coarser textures.

b. Variation in Pixel Intensities

Contrast measures the distribution of gray-level differences, reflecting the degree of visual intensity variation. It combines both standard deviation and kurtosis of the intensity histogram. Tamura defined contrast as:

$$F_{contrast} = \frac{\sigma}{\alpha_4{}^{1/4}} \tag{7}$$

where, σ is the standard deviation of gray levels, and α_4 be the fourth central moment (kurtosis). This ensures that both the spread and the peakedness of intensity variations are considered. Textures with high brightness variation will produce higher contrast values**.**

c. Directionality (Orientation Distribution)

Directionality evaluates the degree to which texture exhibits strong orientation patterns. It is based on the gradient field of the image. Directionality is defined as the sharpness of peaks in this histogram:

$$F_{\text{directionality}} = \sum_i (\theta_i - \theta_{peak})^2(\theta_i) \tag{8}$$

Lower values indicate strong directional alignment, while higher values indicate random or isotropic textures [5].

d. Line-Likeness

Line-likeness quantifies the degree to which neighboring pixels share similar orientations, effectively capturing linear structures in the texture. It is based on co-occurrence of gradient orientations.

$$F_{\text{line-likeness}} = \sum_{i,j} p(i,j)\cos(2(\theta_i - \theta_j)) \tag{9}$$

where, $p(i,j)$ is the probability of two adjacent pixels having orientations θ_i and θ_j . High values indicate textures dominated by aligned linear structures.

e. Regularity

Regularity measures how uniform or repetitive the texture patterns are. It is defined as the inverse of the variance across the other features:

$$F_{\text{regularity}} = \frac{1}{1+\sigma_{features}} \tag{10}$$

where, $\sigma_{features}$ is the variance among the set $\{F_{croaseness}, F_{contrast}, F_{\text{directionality}}, F_{\text{line-likeness}}\}$. Regular textures such as grids or stripes have higher regularity values.

f. Roughness

Roughness is a composite measure that integrates both coarseness and contrast, reflecting the overall complexity of the texture:

$$F_{\text{roughness}} = F_{croaseness} + F_{contrast} \tag{11}$$

This measure is essential for distinguishing textures that are visually complex from those that are smooth [2].

3.4 Graph construction theory (K-Nearest Neighbors algorithm)

K-Nearest Neighbors is a simple powerful algorithm used for classification, regression, and clustering. In graph-based models (like fingerprint or image matching papers), k-NN can also be used for clustering or neighborhood graph construction. Each node (feature or point) connects to its K nearest nodes based on Euclidean or similarity distance. This builds a k-NN graph, where edges represent neighborhood relationships. This helps in organizing data and reducing computational complexity before classification or matching [14].

3.5 Classification deep learning networks

3.5.1 Graph Convolution Neural Network

GCN is the most cited paper in the GNN literature and the most commonly used architecture in real-life applications. GCNs extend the traditional convolution operation to graph structures by defining convolutions on nodes and their neighborhoods. GCNs include both spatial-based and spectral-based approaches. Spatial-based GCNs directly perform convolutions on each node's local neighborhood, while spectral-based GCNs use graph Laplacians to define convolution in the frequency domain [15].

It relies on spectral convolutions that capture global graph properties, GCNs are widely used in semi-supervised learning tasks like node classification, community detection, and other graph-based tasks where localized information is crucial. Mathematically, each layer is characterized by the propagation rule defined by Kipf and Welling [16]:

$$H^{(l+1)} = \sigma\left(\widetilde{\mathrm{D}}^{-\frac{1}{2}}\widetilde{\mathrm{A}}\,\widetilde{\mathrm{D}}^{-\frac{1}{2}}H^{(l)}\,W^{(l)}\right) \tag{12}$$

where,

- $\widetilde{\mathrm{A}} = A + I$: is the adjacency matrix with added self-loop.
- $\widetilde{\mathrm{D}}$: is the degree matrix of $\widetilde{\mathrm{A}}$.
- $H^{(l)}$: is the feature matrix at layer l.
- $W^{(l)}$: is the weight matrix.
- σ: is an activation function.

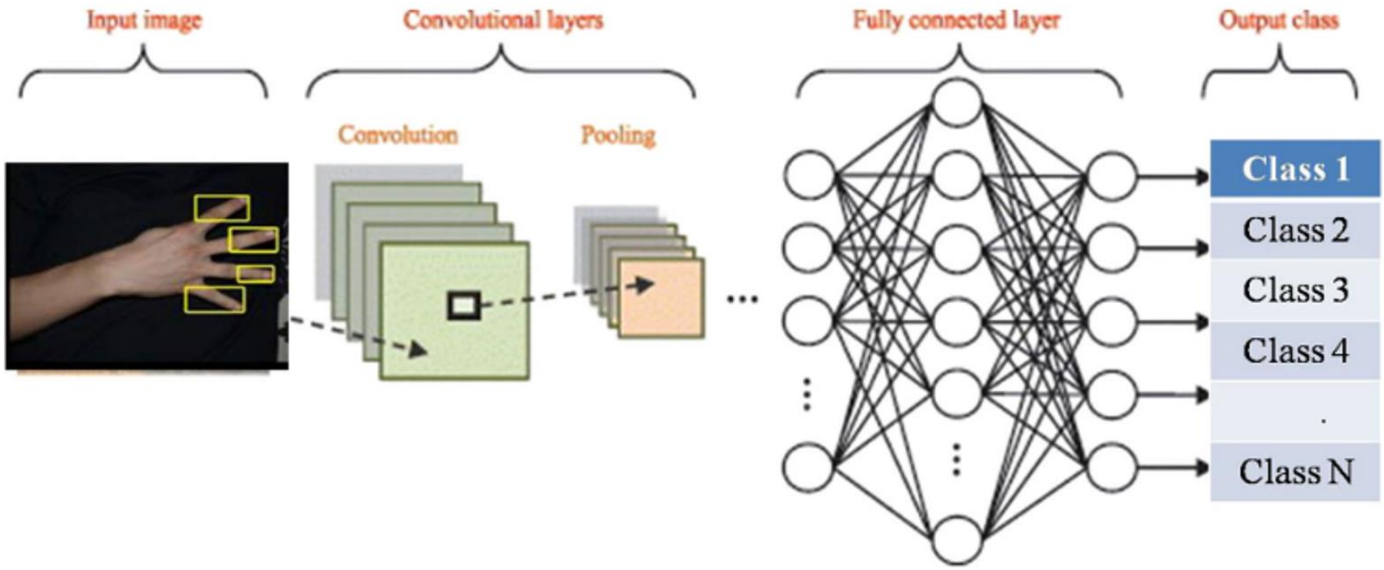

Figure 3. Convolution Neural Network (CNN)

3.5.2 Convolutional Neural Network

CNNs are a form of deep learning specifically created for the analysis of visual data and thus can images and videos. They have achieved greater accuracy than traditional machine learning techniques. CNN architectures normally comprise a few sequential layers, including convolutional layers that extract features with the help of filters, pooling layers that lower the dimensionality and complexity of computation by reducing the size of data, flattening layers that transform extracted features into linear vectors, dense layers generating output through activation functions like ReLU, etc. To prevent overfitting some neurons are randomly dropped during training using dropout layers. Figure 3 shows an example of convolution neural network.

CNNs find extensive applications across image processing, robotics, medical imaging, data analysis, and business intelligence [17].

3.5.3 Gated Recurrent Unit

The Gated Recurrent Unit (GRU) is particularly known as that type of RNN among the family of recurrent neural networks which can efficiently model sequential data. It has two major control gates: a reset gate that, if not very correctly tuned will allow some irrelevant past information, and an update gate that makes new input balanced with historical context. The rules control the flow of information in such a way that long-range dependencies can be effectively captured and at the same time address the problem of vanishing gradients encountered by RNNs. The GRUs are computationally less complex than Long Short-Term Memory networks (LSTMs), hence providing better efficiency as well as faster training. The temporal patterns sign language recognition tasks can be identified by these layers and further utilized for improving identification concerning gesture sequences and transitions [18].

3.5.4 Long Short-Term Memory

LSTM networks extend and improve the traditional architecture of Recurrent Neural Networks in learning long-range dependencies within sequential data. Where basic RNNs mostly fail due to vanishing or exploding gradients, LSTMs avoid such limitations with specific gating architecture. There are an input gate controlling what new information is added to the cell state, a forget gate that continuously removes less important information, and an output gate controlling what part of the information should be outputted at each time step. All these features enable the LSTM model to keep relevant context information for very long sequences and hence make it extremely useful in tasks such as speech recognition, language modeling, or time-series prediction [19].

They control what information comes in and goes out of the network, allowing it to keep important context for long sequences while getting rid of less important information. Because they can model time so well, LSTMs are used in all applications that need sequence processing, including speech recognition and language modeling; and in particular sign language recognition where the correct understanding of rather complicated gesture sequences is crucial [20].

4. PROPOSED METHODOLOGY

Figure 4 presents the research framework of the proposed system. It summarizes the complete pipeline, starting from image preprocessing and feature extraction, followed by k-NN graph construction and data splitting, then adaptive GCN training and testing. Finally, the model is evaluated using standard performance metrics, including accuracy, precision, recall, and F1-score.

This study hereby develops an adaptive GCN deep learning model which uses the skeleton keypoints extracted for one particular word as input. The model makes a GCN network and inserts layers of CNN, GRU, and LSTM within the internal structure of the network utilizing their properties to come up with a generalized powerful model that could be used for sign language recognition. To validate this proposed model, two comprehensive datasets covering both American Sign Language (ASL) and ArSL were used to test how versatile and broadly applicable it is. This framework presents the system under five stages: preprocessing stage, features extraction stage, training stage, testing stage, and evaluation stage shown in Figure 4.

4.1 Preprocessing

Preprocessing of the hand sign image dataset involves steps

meant to improve visual quality, reduce computational requirements, and provide favorable conditions for extracting features. It starts with grayscale conversion whereby RGB images are transformed into single channel presentations. This reduces data dimensionality and the cost of computation while maintaining shape and texture clues that are very relevant towards achieving recognition accuracy. Next, histogram equalization is applied to enhance contrast through better utilization of the full range by setting pixel intensities to be more evenly spread out; detail is increased on an image surface and minute variations supporting pattern recognition become apparent. Images are normalized whereby pixel values get scaled within a standard range (most commonly between 0-1) so as to keep consistency across the entire dataset thus enabling steady model training.

The pictures are then run with a Gaussian blur, which takes away high sounds and smooths out unwanted looks, making it less likely that noise will cause mistakes when getting features. A step that comes right after normalizing keeps the same level of data spread after this smoothing. Changing the size of the image helps to make all examples the same size, allowing for consistent input handling and quick calculation during both training and testing. Another round of normalization makes sure data is steady before going into the feature-getting and deep learning steps, thus making the whole finding system better accurate strong and general.

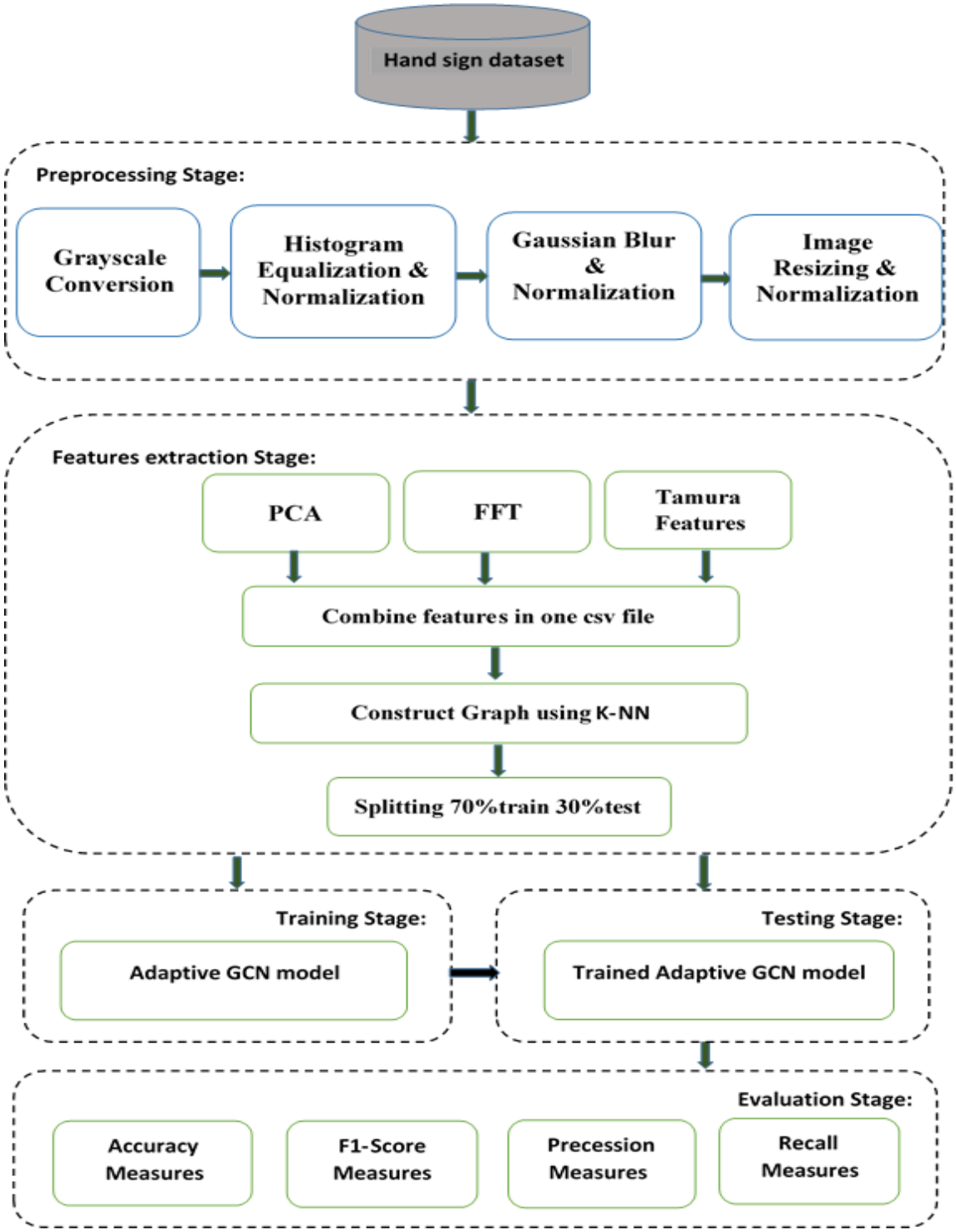

Figure 4. Research framework

4.2 Features extraction

The feature extraction becomes the core of a proposed framework for sign language recognition because it describes how the most salient and informative features are drawn out from preprocessed images of hand signs. Three complementary extraction techniques-PCA, Fast Fourier Transform (FFT), and Tamura texture descriptors-are used in this work to provide varied viewpoints on data. As the first step, PCA is applied as a feature-extraction transform to generate uncorrelated, energy-ordered scores that serve to enrich representation. Only the first ten principal components have been retained in this study as they contain most of the discriminative information and thus represent the strongest relevant features necessary for classification. Next, FFT will be applied such that image information will be converted from the spatial domain into its frequency domain. Periodic structures and subtle variations are well captured in their frequency domains which can never be seen from a spatial perspective. This step helps much when periodic textures need recognition and minute details regarding gestures are required. Tamura features encode those texture characteristics in terms of a human perceptual attribute with dimensions such as coarseness, contrast, and directionality. Such subjective intake makes the model more sensitive to fine changes in shape and texture. Normalization ensures equal participation for all features at this and subsequent stages, particularly when a feature set does not overly dominate results due to scale differentials. After normalization has been completed, these features from the PCA, FFT, and Tamura methods are appended together and stored within CSV files combining such miscellaneous features into a single representation will guarantee very wide and strong coverage over all possible visual and structural characteristics that exist within the dataset. Once the properties of each image have been saved as a vector, the KNN algorithm runs between every two vectors to find the edges among the nodes- hence, images features vectors in this chunk by measuring their degree of closeness so that they can be inputted into GCNs layers which require data structured as a graph of nodes and edges. The consolidated feature dataset is split into 70% training data and 30% testing data to comprehensively assess and validate how well the model generalizes, i.e., on unseen scenarios and datasets how reliable and effective recognition systems might turn out to be.

4.3 Graph construction procedure (K-Nearest Neighbors graph per mini-batch)

A separate graph is constructed for each mini-batch (chunk) of the dataset in order to provide a graph structure for subsequent graph convolution operations. The mini-batch size is 32 and each image features vector represnts anode in the graph. The resulting graph for each batch contains exactly 32 node. The graph is constructed using a k-nearest neighbor rule with K=2 under the Euclidean distance, meaning that each node is connected to its two closest neighbors in the feature space. Because the neighborhood relation is computed independently for each node, the resulting k-NN adjacency is generally asymmetric and the graph is directed. The outgoing degree of every node is fixed at 3 outgoing edges (two edges to the nearest neighbors plus one self-loop), whereas the incoming degree is not fixed and varies depending on how frequently a node is selected as a nearest neighbor by other nodes.

4.4 Classification stage

All important features are extracted in previse stage to use in the current stage. Where the classification stage is central to the effectiveness of the proposed methodology, transforming refined, high-dimensional feature vectors into accurate and interpretable class predictions. To this end, we designed a structured yet adaptable framework integrating graph-theoretic insights with advanced deep neural network architectures. This careful integration allows the model to capitalize on both spatial and temporal complexities present in multilingual hand gesture datasets, thus enhancing generalizability and accuracy across different languages, particularly Arabic and English.

4.4.1 Training stage: Adaptive Graph Convolutional representation

A more elaborate graph-based recognition pipeline where a k-nearest-neighbor (k-NN) graph injects relational structure into otherwise independent handcrafted descriptors. In this setup, feature vectors become nodes and edges represent proximity under some distance metric, typically the Euclidean distance as a special case of Minkowski. Then, information for one node embedding can be propagated by stacking Graph Convolutional Network (GCN) layers such that each embedding will have not only its own attributes but also those of the immediate local neighborhood.

Each GCN layer works by building a k-nearest neighbor (k-NN) graph. The single feature vectors shown as nodes in the graph, while edges show the closeness-based similarity among these vectors. Flexible behavior is done through changing graph updates, where joining links change during training to best show natural data structures across Arabic and English datasets. Figure 5 shows an adaptive GCNs model layers. The Adaptive GCNs layers are penetrated sequentially, meaning that the extracted features forms from each layer serve as an entry point to the next layer.

The Adaptive GCNs model architecture that progressively transforms hand-sign feature vectors into a robust representation suitable for accurate multi-class recognition across heterogeneous datasets. First, an adaptive neighborhood structure is introduced by constructing a k-nearest-neighbor graph, where each feature vector is treated as a node and edges encode proximity-based similarity. Two consecutive GCN layers then perform neighborhood aggregation using the normalized adjacency matrix. This stage injects relational context into otherwise independent descriptors and enables the model to learn similarity-consistent patterns by propagating information across nearby nodes, thereby improving robustness to intra-class variation and reducing sensitivity to dataset-specific capture conditions.

Next, a stack of convolution–activation–pooling blocks refines the graph-enriched embedding's by learning higher-level, locally compositional patterns while max-pooling compresses the feature maps and increases invariance to minor perturbations. A recurrent stage, implemented as a GRU followed by an LSTM, is subsequently applied to model sequential dependencies within the learned representations and to capture both short- and long-range correlations, which further stabilizes recognition under variability and improves generalization. Finally, the resulting representation is flattened and passed through fully connected layers to produce class

probabilities. Overall, by combining relational learning (GCN), hierarchical pattern abstraction (CNN), and dependency modeling (GRU/LSTM), the proposed architecture enhances feature robustness and is explicitly designed to generalize effectively across multiple sign-language datasets rather than overfitting to a single dataset distribution.

4.4.2 Testing stage

In this stage, the trained adaptive GCN model undergoes evaluation using the test subset (30% of the dataset, as per the provided document).

4.5 Evaluation stage

Precision, recall, and F1-score measures are also calculated in Figure 4 explain the results of these measures for the proposed model. It had the highest values over all these measures.

Accuracy measures the overall correctness of the model's predictions and is calculated as the proportion of correctly classified instances (both true positives and true negatives) relative to the total number of predictions. It is formally defined in Eq. (13), utilizing the standard classification components: True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) [20].

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{13}$$

The ratio of True Positives to All Positives is known as precision. It is formally defined in Eq. (14) [21].

$$precision = \frac{TP}{TP+FP} \tag{14}$$

The Recall (R) is the measure of the model correctly identifying True Positives. It is formally defined in Eq. (15) [22].

$$Recall = \frac{TP}{TP+FN} \tag{15}$$

The weighted average of Precision and Recall is known as F1-score. It is calculated through Eq. (16) [23].

$$F1 - score = \frac{2.Precision * Recall}{Precision + Recall} \tag{16}$$

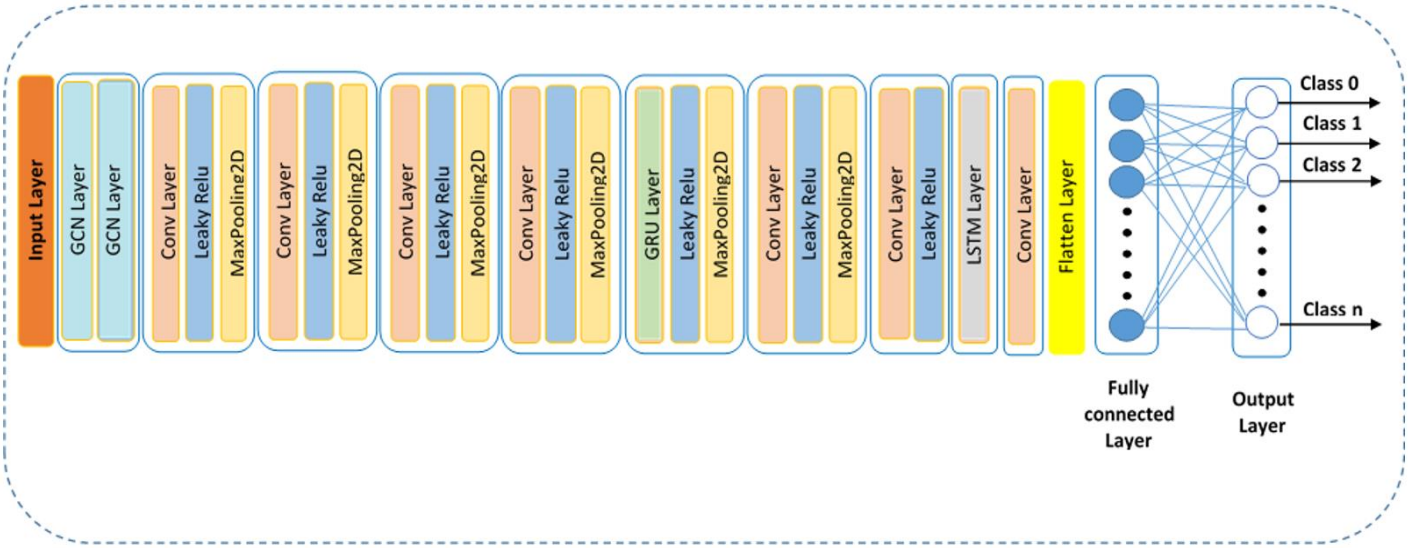

Figure 5. Adaptive Graph Convolutional Networks (GCNs) model

5. EXPERIMENTAL RESULTS

Windows 10 Professional has been utilized as an operating system. The proposed model code and other baseline models have been written in Python 3.6.5 and are implemented using sklearn and Keras library. All experiments have been conducted on processor Core i7 and 8GB RAM. Two datasets, ASL and ArSL have been used to train and test our proposed model.

Table 1 presents the summary of the hyper parameter settings employed during the training phase of the proposed model.

These settings were based on empirical evaluation and best general practices in deep learning for an optimum tradeoff between training efficiency and generalization performance. A learning rate of 10-3 was used so as to keep stable convergence during optimization. The model will be trained over 50 epochs using a batch size of 32, which gives an efficient and consistent update of model parameters. An adaptive GCNs architecture consists of 24 sequential layers wherein convolutional operations use a kernel size of 3 to extract fine-grained local patterns from the input images. Such a relatively small kernel size allows the network to conserve spatial detail while progressively learning hierarchical representations. Adam algorithm is used for optimization due to its adaptive learning rate mechanism by which speed of convergence and stability in training are balanced, hence improving the overall accuracy. LeakyReLU is used to solve the problem of gradients not flowing properly in deeper networks by allowing a small gradient, when the input is negative. This proposed model uses two GCN layers and seven CNN layers distributed among different layers of this model. The number of filters varies from 8 to 64 then goes down to 16. The number of protecting layers for GRU and LSTM is 16. This will ensure better gradient flow going through the network and will help feature learning be better in the later layers.

Table 1. The summary of hyper parameters setting

Name	Value
Learning rate	10^{-3}
No. of epoch	50
Batch size	32
No. of layers	25
Kernel size	3
Optimizer	Adam
Activation function	LeakyReLU
No. of Graph Convolutional Network (GCN) layers	2
No. of Convolutional Neural Network (CNN) filter	8, 16, 32, 64
Gated Recurrent Unit (GRU) units	16
Long Short-Term Memory (LSTM) units	16

Table 2 provide comparing the proposed Adaptive GCN model with the results of a previous work [24] that used the same ArSL dataset is provided.

Table 2. Comparison accuracy of the proposed model for Arabic Sign Language (ArSL) with other research

Researches	Dataset	Accuracy
Adaptive Graph Convolutional Networks (GCNs) model	ArSL	**97**
[24]	ArSL	96.05

The proposed method reached an accuracy rate equal to 97% better than the reference model by 96.05%. Numerically, the gain is not high, but in sign language recognition tasks where there exist subtle changes in hand configurations and complex gesture patterns, even small improvements are quite significant. This improvement explicitly highlights the ability of Adaptive GCN to extract spatial relationship and structural dependency information from hand keypoints which is essential in distinguishing visually similar signs. This also elaborates that the model generalizes very well across different signers and recording conditions so as to be robust enough for real-world applications. The persistence advantage over all existing methods offers empirical evidence for adaptive mechanisms' inclusion into GCN architectures toward improved ArSL recognition in Table 3.

Figure 6 shows both the training loss and validation loss of ArSL dataset.

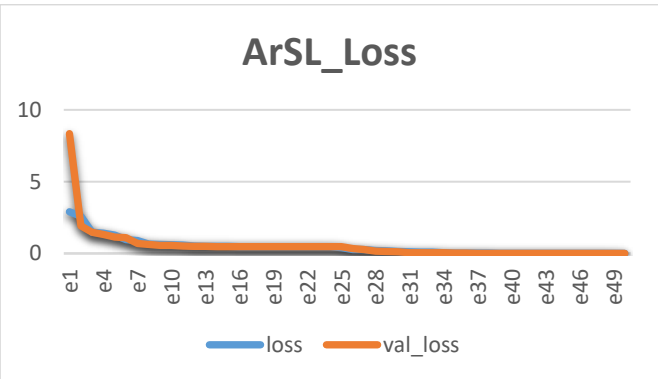

Figure 6. Loss of Arabic Sign Language (ArSL) dataset

Figure 7 shows that both the training loss and validation loss decrease over 50 epochs of model training on the ArSL dataset. Losses drop steeply for the first few epochs, starting with a validation loss of about 8.4 and a training loss of about 2.9, meaning that the model is able to pick up significant trends from the data very quickly. The next several epochs see continued decrease, though now at a reduced pace as the model makes adjustments to its parameters. Once e15 is passed, both curves go flat and head toward zero - an indication that convergence has been achieved by the model itself. The small but steady gap between those two lines suggests the model is likely to generalize well and thus work fine on new data - no overt sign of overfitting.

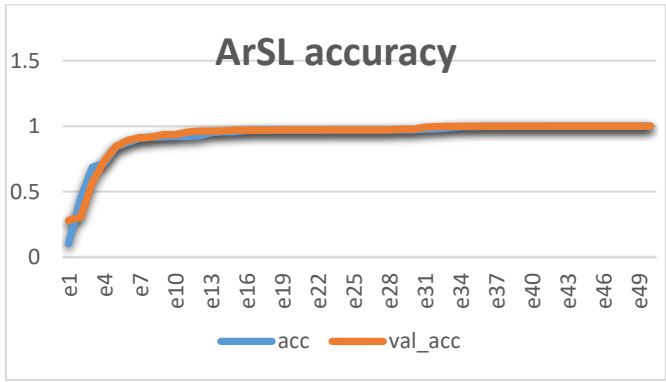

Figure 7. Accuracy of Arabic Sign Language (ArSL) dataset

Table 3. Evaluation metrics of the adaptive GCNs model for Arabic Sign Language (ArSL) dataset

Class	Precision	Recall	F1-Score	Support
0	1.00	1.00	1.00	634
1	1.00	1.00	1.00	502
2	1.00	1.00	1.00	537
3	1.00	1.00	1.00	490
4	1.00	1.00	1.00	517
5	1.00	1.00	1.00	501
6	1.00	1.00	1.00	587
7	1.00	1.00	1.00	511
8	1.00	1.00	1.00	586
9	1.00	1.00	1.00	478
10	1.00	1.00	1.00	458
11	1.00	1.00	1.00	466
12	1.00	1.00	1.00	532
13	1.00	1.00	1.00	482
14	1.00	1.00	1.00	550
15	1.00	1.00	1.00	529
16	1.00	1.00	1.00	546
17	1.00	1.00	1.00	498
18	1.00	1.00	1.00	568
19	1.00	1.00	1.00	491
20	1.00	1.00	1.00	452
21	1.00	1.00	1.00	545
22	1.00	1.00	1.00	551
23	1.00	1.00	1.00	539
24	1.00	1.00	1.00	475
25	1.00	1.00	1.00	412
26	0.48	1.00	0.65	388
27	0.00	0.00	0.00	412
Accuracy			**0.97**	**14228**
Macro avg	**0.95**	**0.96**	**0.95**	**14228**
Weighted avg	**0.96**	**0.97**	**0.96**	**14228**

Notice that both of training accuracy and validation accuracy are decreased gradually over 50 epochs for the ArSL dataset. In the first e1–e5 periods, a sharp rise in both training and validation accuracy is observed, indicating that learning for essential patterns within the data was quick. From about e5

to e15, growth is slow as the model makes minor adjustments toward better comprehension. After approximately e15, both curves settle near 1.0 indicating that indeed an extremely high level of accuracy has been attained by this particular model; there is close overlap between training and validation accuracies to indicate good generalization to unseen data with absolutely no sign of overfitting.

In the classification report proves that this model is strong and solid by getting an overall accuracy of 97% on the ArSL dataset with a total number of test samples equal to 14,228. For most categories from 0 to 25 it gets perfect precision, recall, and F1-score (1.00) meaning not only very precise in guessing the right class (high precision) but also quite exhaustive in finding all relevant samples (high recall). Such performance consistency gives an indication about how robust the model is towards minute nuances of hand gesture and at the same time keeps capability towards differentiating visually close signs.

But results for the last two classes are low. Class 26 gets 48 percent precision and 100 percent recall, which means that though all true instances of this class have been detected, there is a tendency to misclassify other samples from different classes as class 26 (higher false positives). No correct predictions for class 27 (precision = recall = F1-score = 0.00), has fully failed to recognize this particular sign. This can be due to class imbalance or inadequate training samples or more intra-class variability for that sign.

Precision at 0.95, recall at 0.96, and F1-score at 0.95 macro-averaged over all classes indicate the healthy treatment of all classes equally; weighted averages of precision at 0.96, recall at 0.97, and F1-score at 0.96 denote excellent overall accuracy when more heavily populated classes are taken into consideration.

The model shows state-of-the-art performance on most of the classes, except those classes which are underperforming. In specific reference to Classes 26 and 27, thereby suggesting targeted improvements towards these particular classes through data collection, class balancing, or improved feature extraction.

Table 4 provide comparing the proposed Adaptive GCN model with the results of previous works that used the same ASL dataset.

Table 4. Comparison accuracy of the proposed model for American Sign Language (ASL) with other research

Researches	Dataset	Accuracy
Adaptive GCNs	ASL	**1.0**
[2]	ASL	96.68
[25]	ASL	99.51
[26]	ASL	96.96

Table 4 compares the proposed adaptive generative neural network model with previous studies evaluated on the same ASL dataset [2, 25, 26]. The proposed approach achieves 100% test accuracy, surpassing previously reported results (96.68%, 99.51%, and 96.96%). While these baseline criteria are already robust in the field of ASL recognition, the exceptional accuracy observed in our experiments can be attributed to the training dynamics and evaluation protocol. Specifically, the learning curves exhibit a gradual and consistent decrease in loss across training sessions, coinciding with a rise in validation accuracy during training, without the abrupt drops or instability that might typically indicate abnormal training behavior or memory-induced distortions. This pattern is consistent with steady convergence rather than sudden spikes in performance. Furthermore, the test set was completely isolated from the training set, and the model was rigorously evaluated on samples it had never encountered during the optimization process. This minimizes the risk of interference between training and test data and supports the accuracy of the reported test performance under the adopted partitioning. However, we acknowledge that achieving 100% accuracy is uncommon and should be interpreted within the context of the dataset and protocol used.

Both of training loss and validation loss of ASL dataset shown in Figure 8.

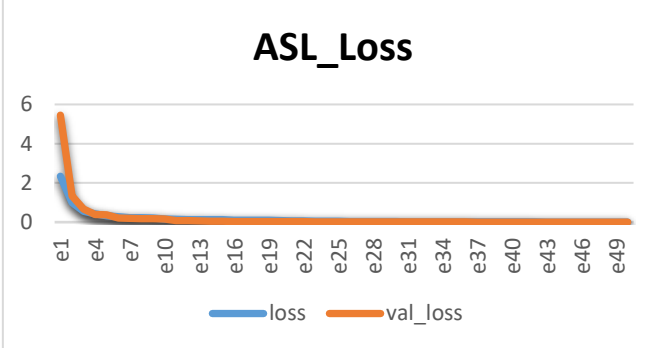

Figure 8. Loss of American Sign Language (ASL) dataset

Figure 8 which actually presents the loss curves and shows how the training and validation loss change over 50 epochs for the ASL dataset. Loss drops sharply at first epochs (e1-e5) for both, which means that the main patterns in the data are being picked up very fast by the model. The drop becomes slower between e5 and e15 as the model fine-tunes its learning further. After about e15, both curves flatten near zero meaning that the model has converged. The close match between training and validation loss all through training will always speak of strong generalization whereby a model will perform well on unseen data without overfitting.

Figure 9 illustrates the training and validation accuracy trends over 50 epochs for the ASL dataset.

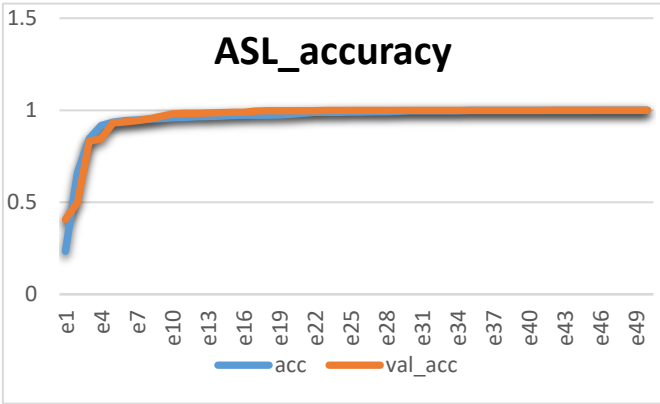

Figure 9. Accuracy of American Sign Language (ASL) dataset

In the first few epochs (e1–e5), there is a quick rise in accuracy as the model easily picks up major patterns from the data in Figure 9. By about e7, both training and validation accuracy are above 0.9 and after e10 they move toward 1.0. After about e15, the curves flatten showing that the model has settled down and is making correct predictions most of the time. The very close tracking between training and validation

accuracy all through indicates very good generalization and hence no overfitting that would show itself by performance differences on training versus unseen data.

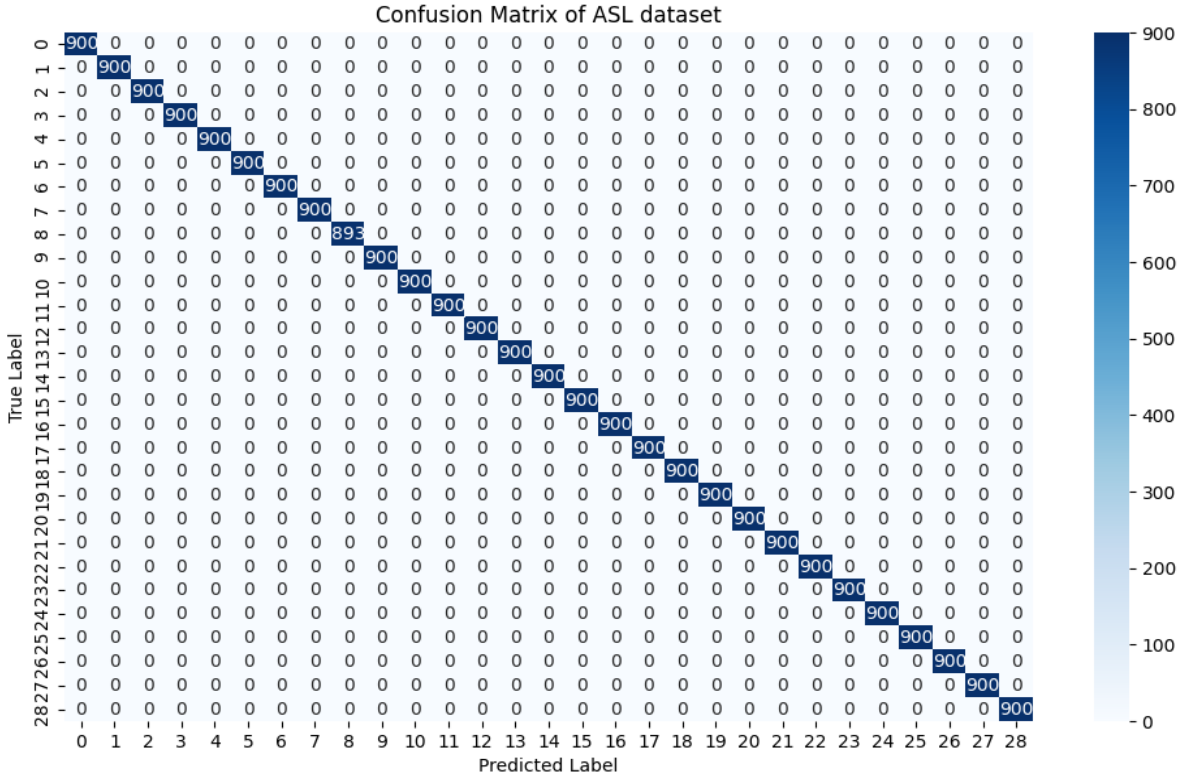

Figure 10. Confusion matrix for American Sign Language (ASL) dataset

Table 5. Evaluation metrics of the adaptive Graph Convolutional Networks (GCNs) model for American Sign Language (ASL) dataset

Class	Precision	Recall	F1-Score	Support
0	1.00	1.00	1.00	900
1	1.00	1.00	1.00	900
2	1.00	1.00	1.00	900
3	1.00	1.00	1.00	900
4	1.00	1.00	1.00	900
5	1.00	1.00	1.00	900
6	1.00	1.00	1.00	900
7	1.00	1.00	1.00	900
8	1.00	1.00	1.00	893
9	1.00	1.00	1.00	900
10	1.00	1.00	1.00	900
11	1.00	1.00	1.00	900
12	1.00	1.00	1.00	900
13	1.00	1.00	1.00	900
14	1.00	1.00	1.00	900
15	1.00	1.00	1.00	900
16	1.00	1.00	1.00	900
17	1.00	1.00	1.00	900
18	1.00	1.00	1.00	900
19	1.00	1.00	1.00	900
20	1.00	1.00	1.00	900
21	1.00	1.00	1.00	900
22	1.00	1.00	1.00	900
23	1.00	1.00	1.00	900
24	1.00	1.00	1.00	900
25	1.00	1.00	1.00	900
26	1.00	1.00	1.00	900
27	1.00	1.00	1.00	900
28	1.00	1.00	1.00	900
Accuracy			**1.00**	**26093**
Macro avg	**1.00**	**1.00**	**1.00**	**26093**
Weighted avg	**1.00**	**1.00**	**1.00**	**26093**

In Figure 10 the confusion matrix for all classes of the ASL dataset have been shown.

The confusion matrix of the ASL dataset shows that the proposed model achieved very high classification ability, with values almost entirely concentrated on the main diagonal. This means that most samples were correctly classified with minimal transitions to other categories. This pattern reflects predictive consistency and stability in distinguishing between signal features, indicating that the learned representations were sufficient to clearly separate the categories and minimize overlap. The number of correctly classified samples in most categories was also relatively similar, with some categories showing a decrease compared to others. This is generally understood to be due to a slight variation in the sample size available for that category within the test data, rather than classification ambiguity. Overall, these results support the conclusion that the model efficiently generalizes signal-associated patterns within the experimental setup, and that its performance was unaffected by the presence of similar categories in terms of visual structure or extracted features. This is directly reflected in the rarity of errors outside the matrix diagonal. Table 5 Shows the evaluation metrics of the adaptive GCNs model for ArSL dataset with its details.

6. CONCLUSION

This paper has presented a multilingual sign language recognition pipeline that incorporates advanced preprocessing, complementary feature extraction, and adaptive graph-based learning with the help of sequential deep neural modules. The framework comprises PCA, FFT, and Tamura texture descriptors in an adjustable architecture of GCN enriched

further by CNNs, GRUs, and LSTM layers to spatial textures at a very detailed level; patterns within the frequency domain; temporal dependencies. Empirical results reported herewith significantly outperform any current state-of-the-art efforts recording 97% accuracy on the ArSL dataset under both macro- and weighted-average scores as well as 100% accuracy on ASL where every single class achieves perfect precision, recall, and F1-score.

While the ASL results underscore extraordinary recognition capability, perfect scores must be guarded about in terms of possible due influences by specific factors pertaining to the dataset or overfitting. Performance disparity noticed in certain classes of ArSL are indicative and prescriptive towards improvements that may include balancing data, synthetic augmentation, or more optimized class-specific-features that can improve robustness even further. Meanwhile, these findings have convincingly validated that handcrafted descriptors conjoined with learned deep features within a graph-based schema adaptively maximize the accuracy for cross linguistic sign recognition tasks. Immediate future directions will involve extending this very framework to continuous sign language besides making it resilient under real-world variability in addition to speeding up computations on resource-constrained platforms.

REFERENCES

[1] Bhadra, R., Kar, S. (2021). Sign language detection from hand gesture images using deep multi-layered convolution neural network. In 2021 IEEE Second International Conference on Control, Measurement and Instrumentation (CMI), Kolkata, India, pp. 196-200. https://doi.org/10.1109/CMI50323.2021.9362897

[2] Ahmed, I.T., Gwad, W.H., Hammad, B.T., Alkayal, E. (2025). Enhancing hand gesture image recognition by integrating various feature groups. Technologies, 13(4): 164. https://doi.org/10.3390/technologies13040164

[3] Abd Al-Latief, S.T., Yussof, S., Ahmad, A., Khadim, S.M., Abdulhasan, R.A. (2024). Instant sign language recognition by WAR strategy algorithm based tuned machine learning. International Journal of Networked and Distributed Computing, 12(2): 344-361. https://doi.org/10.1007/s44227-024-00039-8

[4] Park, G., Chandrasegar, V.K., Koh, J. (2023). Accuracy enhancement of hand gesture recognition using CNN. IEEE Access, 11: 26496-26501. https://doi.org/10.1109/ACCESS.2023.3254537

[5] Ahmed, H.M. (2023). Texture feature extraction using tamura descriptors and scale-invariant feature transform. Journal of Education & Science, 32(4): 91-103. https://doi.org/10.33899/edusj.2023.143728.1394

[6] Miah, A.S.M., Hasan, M.A.M., Nishimura, S., Shin, J. (2024). Sign language recognition using graph and general deep neural network based on large scale dataset. IEEE Access, 12: 34553-34569. https://doi.org/10.1109/ACCESS.2024.3372425

[7] Sarkar, U., Chakraborti, A., Samanta, T., Pal, S., Das, A. (2024). Enhancing ASL recognition with GCNs and successive residual connections. arXiv preprint arXiv:2408.09567. https://doi.org/10.48550/arXiv.2408.09567

[8] Vasudevan, V., Bassenne, M., Islam, M.T., Xing, L. (2023). Image classification using graph neural network and multiscale wavelet superpixels. Pattern Recognition Letters, 166: 89-96. https://doi.org/10.1016/j.patrec.2023.01.003

[9] Han, K., Wang, Y., Guo, J., Tang, Y., Wu, E. (2022). Vision GNN: An image is worth graph of nodes. Advances in Neural Information Processing Systems, 35: 8291-8303.

[10] Syaputra, H., Nurmaini, S., Partan, R.U., Roseno, M.T. (2025). Enhancing medical image instance segmentation using histogram equalization and blind deblurring: A preliminary study. Ingénierie des Systèmes d'Information, 30(5): 1363-1372. https://doi.org/10.18280/isi.300521

[11] Hummel, R.A., Kimia, B., Zucker, S.W. (1987). Deblurring gaussian blur. Computer Vision, Graphics, and Image Processing, 38(1): 66-80. https://doi.org/10.1016/S0734-189X(87)80153-6

[12] Al Hammami, D.J., Hassan, R.F. (2025). A hybrid 1D CNN-LSTM model for face recognition using PCA features. Ingénierie des Systèmes d'Information, 30(8): 2067-2076. https://doi.org/10.18280/isi.300812

[13] Kamath, R.C., Vijay, G.S., Prasad, G., Rao, P.K., Shetty, U.K., Parameshwaran, G., Shenoy, A., Shetty, P. (2023). Feasibility analysis of Tamura features in the identification of machined surface images using machine learning and image processing techniques. Engineering Proceedings, 59(1): 92. https://doi.org/10.3390/engproc2023059092

[14] Alshammari, M., Stavrakakis, J., Ahmed, A.F., Takatsuka, M. (2023). Random projection forest initialization for graph convolutional networks. MethodsX, 11: 102315. https://doi.org/10.1016/j.mex.2023.102315

[15] Hayder, N.M.M.A., Seno, S.A. H., Noori, H., Zabihzadeh, D., Manaa, M.E. (2025). Improved DDoS attack detection-based feature selection by using graph convolutional network-transformer model. Operational Research in Engineering Sciences: Theory and Applications, 8(2): 22-46. https://doi.org/10.5281/zenodo.17160174

[16] Kipf, T.N., Welling, M. (2017). Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907. https://doi.org/10.48550/arXiv.1609.02907

[17] Al-Hammadi, M., Muhammad, G., Abdul, W., Alsulaiman, M., Bencherif, M.A., Mekhtiche, M.A. (2020). Hand gesture recognition for sign language using 3DCNN. IEEE Access, 8: 79491-79509. https://doi.org/10.1109/ACCESS.2020.2990434

[18] 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, 11(11): 1780. https://doi.org/10.3390/electronics11111780

[19] López, L.I.B., Ferri, F.M., Zea, J., Caraguay, Á.L.V., Benalcázar, M.E. (2024). CNN-LSTM and post-processing for EMG-based hand gesture recognition. Intelligent Systems with Applications, 22: 200352. https://doi.org/10.1016/j.iswa.2024.200352

[20] Hussain, A., Ul Amin, S., Fayaz, M. (2023). An efficient and robust hand gesture recognition system of sign language employing finetuned Inception-V3 and Efficientnet-B0 network. Computer Systems Science & Engineering, 46(3): 3509-3525.

https://doi.org/10.32604/csse.2023.037258
[21] Shhatha, A.M., Alsaif, O.I. (2025). Enhancing cybersecurity through malware detection based on machine learning technique. Kufa Journal of Engineering, 16(3): 82-100. https://doi.org/10.30572/2018/KJE/160306
[22] Sundar, B., Bagyammal, T. (2022). American sign language recognition for alphabets using MediaPipe and LSTM. Procedia Computer Science, 215: 642-651. https://doi.org/10.1016/j.procs.2022.12.066
[23] Allahem, H., El-Ghany, S.A., Abd El-Aziz, A.A., Aldughayfiq, B., Alshammeri, M., Alamri, M. (2025). A hybrid model of feature extraction and dimensionality reduction using ViT, PCA, and random forest for multi-classification of brain cancer. Diagnostics, 15(11): 1392. https://doi.org/10.3390/diagnostics15111392
[24] Elshaer, A.M., Ambioh, Y., Soliman, Z., Ahmed, O., Elnakib, M., Safwat, M., Elsayed, S.M., Khalid, M. (2024). Enhancing Arabic alphabet sign language recognition with VGG16 deep learning investigation. In 2024 14th International Conference on Electrical Engineering (ICEENG), Cairo, Egypt, pp. 184-186. https://doi.org/10.1109/ICEENG58856.2024.10566400
[25] Alsolai, H., Alsolai, L., Al-Wesabi, F.N., Othman, M., Rizwanullah, M., Abdelmageed, A.A. (2024). Automated sign language detection and classification using reptile search algorithm with hybrid deep learning. Heliyon, 10(1): e23252. https://doi.org/10.3390/electronics11111780
[26] Sharma, A., Mittal, A., Singh, S., Awatramani, V. (2020). Hand gesture recognition using image processing and feature extraction techniques. Procedia Computer Science, 173: 181-190. https://doi.org/10.1016/j.procs.2020.06.022