International Journal of Engineering Trends and Technology
ISSN: 2231-5381/ https://doi.org/10.14445/22315381/IJETT-V74I6P111

Volume 74 Issue 6, 157-172, June 2026

Original Article

TamilNets-V1-A Novel Resource Constraint Deep Learning Framework for Tamil Sign Language Detection Mechanism

V. Sivakamasundari[1], P. Anbarasu[2], M. Arivazhagan[3], K. Rajendran[4]

[1,2]*Department of Electronics, Dr. Kalaignar Government Arts College, (Affiliated to Bharathidasan University, Trichy), Kulithalai, Tamil Nadu, India.*
[3]*Department of Physics, Government Arts College, Trichy, (Affiliated to Bharathidasan University, Trichy), Tamilnadu, India.*
[4]*Department of Electronics, L.R.G Government Arts College for Women, Tiruppur, (Affiliated to Bharathiar University, Coimbatore), Tamilnadu, India.*

[1]*Corresponding Author : Pon_anbarasu@rediffmail.com*

Received: 14 October 2025 Revised: 23 March 2026 Accepted: 28 March 2026 Published: 27 June 2026

Abstract - *Tamil is considered one of the ancient and unique languages, enriched with distinctive texts and knowledge in various domains, including literature, healthcare, scientific devotion, agriculture, and so on. However, its rich resources remain on the darker side to the deaf and dumb community since they are facing the challenges of establishing better social interactions. Sign Language (SL) is a kind of communication that helps them to understand and increase their knowledge. Furthermore, signs, gestures, and facial expressions are used as catalysts for enhancing them to learn new skills, thereby improving their quality of life. With the advent of ML and DL, Tamil Sign Language Recognition Systems (TSLRs) have reached their new peak of design but suffer from bottlenecks such as computational complexity, high resource constraints, and consume more device power. To solve this challenge, this research article proposes a novel resource and computationally aware cognitive framework for TSLR Systems. The cognitive framework TamilNets-V1 ensembles the quantized Efficientnet-Lite0 architecture for achieving high performance and low computation. To prove the excellence of the suggested framework, the effectiveness of the model is compared with that of other existing models by deploying it on different hardware architectures. The suggested framework brings the bright insights of deploying the learning algorithms on the hardware so that portable devices will be in the hands of the deaf and dumb community to learn the uniqueness of the ancient Tamil language.*

Keywords - *EfficientNet-Lite0, Hardware deployment, Tamil, Tamil Sign Language Recognition System, Quantized model.*

1. Introduction

The oldest classical language, 'TAMIL,' offers a unique linguistic experience that challenges and inspires the next generation of learners [1-3] rich drapery of Tamil culture, spanning from healthcare to art, literature, and tradition. Thirukkural, Thirumanthiram, and many ancient Tamil texts have the ability to transform one's life into flying divine colours. Many Tamil texts are translated into many languages, which shows its cultural enrichments and healthy contents, to be utilized by today 's younger population [4-6]. Understanding and accessing the cultural aspects of the Tamil language can pose difficulties for individuals with speech and hearing disabilities. Sign Language (SL) serves as a primary mode of non-verbal communication for such individuals, offering a convenient way to convey thoughts and emotions to those familiar with it. However, interacting with individuals who lack knowledge of SL presents communication challenging and frequently necessitates the presence of a qualified interpreter [7]. To enable the deaf and mute community to appreciate better the richness and cultural legacy of the Tamil language, it is crucial to encourage the acquisition and practical use of Tamil Sign Language (TSL) [8]. Artificial Intelligence techniques [9], such as Machine and Deep Learning algorithms for SL detection, have an explosive potential to solve the aforementioned problem. To enable efficient real-time communication between SL users and non-signers, researchers are developing methods capable of accurately recognizing and converting SL gestures into textual or spoken forms. The goal of ongoing research is to increase the precision and usability of SL classification and recognition. In this context, the identification of TSL is considered a significant step toward preserving linguistic diversity and improving accessibility for the deaf and mute community [10].

Recently, ML techniques, such as CNN, ResNet, EfficientNet, and hybrid LSTM, have been utilized to solve sign language recognition problems [11]. Even though these techniques have shown high recognition accuracy, many of these techniques require high computational complexity and high memory usage. Therefore, these techniques have not yet been effectively deployed on edge devices such as Raspberry Pi, smartphones, etc. [15]. Most of these techniques have focused on recognizing American Sign Language and Indian Sign Language using various datasets, while there is limited literature available on recognizing Tamil Sign Language using machine learning techniques [12]. Therefore, there is a need to develop a computationally efficient, deployable deep learning framework that can effectively recognize sign language with low complexity and reduced inference time [12].

In comparison with existing sign language recognition techniques, which are mainly focused on achieving better performance by employing large deep learning models, the proposed TamilNets-V1 framework focuses on both recognition performance and computational efficiency. The novelty of the current contribution is the integration of a Pelican Optimization-based hyperparameter tuning mechanism [13] with a quantized EfficientNet-LiteL0 model, which can be effectively employed for Tamil Sign Language recognition with reduced model size, parameters, and recognition time [14]. Additionally, the proposed framework is also suitable for deployment on edge devices, as demonstrated by the Raspberry Pi deployment [15], which is another novelty of the current contribution. This is a major distinction from existing contributions [11].

1.1. Research Contribution

Motivated by the aforementioned challenge, this research article proposes a novel resource-constrained quantized Efficientnet-LiteL0 model for an effective classification of Tamil Sign letters, which will play an essential role in understanding the linguistically enriched language. The suggested framework consists of four components: 1) Tamil Sign Language Collection, 2) Data pre-processing and Augmentation, 3) Model Design, and 4) Hardware Deployment. To the best of our knowledge, this technique is considered to be the first of its kind to deploy the low-complex model design on different hardware architectures. The key research contributions are as follows:

1. The paper introduces the quantized EfficientNet-LiteL0 model for achieving the high efficacy detection of Tamil Signs.
2. The proposed cognitive framework consists of five components: 1) Tamil Sign Data Collection (TSDC), 2) Data pre-processing and Augmentation Process (DPAP), 3) Quantized Model Design (QMD), 4) Hardware Deployment, and 5) Performance Evaluation.
3. The hybrid CLAHE algorithm and SMOTE-GAN techniques are introduced to pre-process the images and to overcome the class imbalance problem.
4. The paper introduces the Hardware Deployable Deep Learning Model for achieving low computational complexity and high speed of classification.
5. Extensive Experimentation is carried out using Real time Tamil Sign Video Datasets, and various performance metrics includes accuracy, precision, recall, F1-score, memory, FLOPS, parameters, inference time, and model size, are calculated. Furthermore, the suggested model was deployed in different architectures in which the hardware metrics are analysed.
6. To prove the efficacy of the suggested methodology, performance metrics has been compared with the other SOTA learning models in which the TamilNets-V1 outperformed the existing models.

The novelty of the proposed TamilNets-V1 framework is based on the development of a resource-constrained DL architecture for the recognition of the TSL with high accuracy and low computational complexity. Contrary to existing resource-constrained DL architectures for sign language recognition and gesture recognition, the proposed framework has been developed based on the integration of the quantized EfficientNet-LiteL0 architecture with Pelican Optimization-based hyperparameter tuning for the improvement of classification accuracy with low computational complexity. Additionally, the proposed framework has been developed for hardware-level implementation with Raspberry Pi-based edge devices for the development of efficient SL recognition-based applications. The development of the dataset for the proposed framework based on the Tamil Sign Language and the integration of the proposed approach with the Raspberry Pi-based edge device for the analysis of the proposed framework's feasibility are the novelties of the proposed framework compared with existing studies in sign language recognition.

1.2. Structure of the paper

The remainder of the paper is organized as follows: 1) Section 2 presents the existing ML and DL models deployed for the Sign Language detection and classification. The dataset collection, data pre-processing, augmentation techniques, and suggested method were illustrated in Section 3. The experimental evaluation, results discussion, and comparative analysis are demonstrated in Section 4. Finally, the paper concludes with future directions for improvement in Section 5.

2. Related Works

Gobhinath et al. (2025) [16] proposed the TSLI-AGNN-GEO method for TSL identification using an Auto-Metric GNN optimized with Golden Eagle Optimization. Input gesture images were denoised using Savitzky–Golay and

features extracted via GLCM-WAA. The optimized AGNN accurately identified TSL gestures. Their model achieved 19.18–24.48% higher accuracy and reduced computation time compared to existing techniques inludes TSLI-SFO-SOM and TSLI-CED-SIFT. However, the system's real-time adaptability and generalization across various lighting conditions remain limited.

In "Comparative Study of Hybrid DL Models for Kannada Sign Language Recognition," Hugar et al. (2025) [17] introduce a novel hybrid CNN-HKP-LSTM-Transformer architecture. This research, created using a newly curated dataset of 1080 KSL gestures within the medical domain, effectively bridges the gaps of dataset diversity and spatiotemporal modeling for underrepresented languages. The novel architecture achieved a remarkable 81% testing accuracy, far surpassing conventional CNN-LSTM architectures. However, there exist certain drawbacks, such as the limited vocabulary set (10 gestures) and the sensitivity to lighting conditions, despite its high computational cost.

In "Efficient spatio-temporal modeling for SL recognition using CNN and RNN architectures," Myagila et al. (2025) [18] suggested a hybrid CNN-GRU model with ELU activation to enhance learning efficiency. Utilizing a custom Tanzania Sign Language (TSL) dataset of 16,000 mobile-captured videos, the research overcomes communication barriers and temporal modeling difficulties. The introduced architecture achieved a 94% testing accuracy. However, key limitations include significantly reduced performance in signer-independent settings, where hand dominance variations and unconstrained environments negatively impact the model's overall generalizability.

Arunkumar et al. (2024) [19] suggested a sign language recognition framework based on machine learning methodologies to support communication intended to assist people with hearing disabilities. The presented approach combines computer vision techniques with natural language processing methods to convert hand gestures into textual or spoken outputs in real time. The system aimed to bridge the communication gap and promote inclusivity through automated sign interpretation.

This method significantly improves accessibility for individuals within the deaf community. Although the system demonstrates reduced accuracy under varying illumination conditions and in the presence of complex backgrounds, it remains a limitation. Siddique et al. (2023) [20] proposed a Bangla SL detection system using DL models like Detectron2, EfficientDet-D0, and YOLOv7, deployed on a Jetson Nano-based edge device. Their system, trained on a dataset of 3,760 images, showed high detection accuracy, with Detectron2 achieving mAP@.5 of 94.915 and YOLOv7 achieving up to 97% for specific classes.

The YOLOv7 Tiny model was selected for real-time application due to its low training time and good performance. This system provides a low-cost and portable solution for real-time Sign Language detection in Bangladeshi contexts. A key limitation is reduced detection accuracy under rapid hand movements and low-resolution input.

Prabhu et al. (2023) [20] proposed an ML-based Tamil Sign Language Recognition tool using CNNs to interpret hand gestures into Tamil alphabets or words. The system includes video-based data collection and pre-processing before training the CNN. It obtained a recognition accuracy of 85% on the test dataset, outperforming existing tools. The tool supports deployment on mobile and web platforms for accessibility. However, its performance may be limited by gesture variability and lighting conditions in real-time use.

Priya et al. (2023) [21] developed a TSL translation system using image processing and edge detection techniques. The system processes 31 Tamil alphabets using Canny-based edge extraction and gesture recognition through scale-invariant feature transformation techniques. It maps finger positions to decimal values for effective symbol recognition. This system aims to reduce communication barriers for individuals with hearing and speech impairments. A key limitation is that it may struggle with complex or overlapping hand gestures.

Buttar et al. (2023) [22] introduced a hybrid DL model for real-time American SL Recognition, combining LSTM with MediaPipe holistic landmarks for continuous signs and YOLOv6 for static sign detection. Their system achieved 92% accuracy for dynamic signs and 96% for static signs, supporting signer-independent recognition. The model effectively handles gesture variation in real-time applications, aiding both users and learners. Their dual-algorithm approach ensures precise classification across various sign types. However, the system's performance may degrade with complex background noise and varying lighting conditions.

Sharma et al. (2022) [23] proposed a CNN-based SLRS framework for recognizing ISL gestures using images from 65 users in uncontrolled environments. Data augmentation was performed using affine transformations to enhance generalization. The model achieved accuracies of 92.43%, 88.01%, and 99.52% on three different datasets. Performance was further validated using precision, recall, F-score, and time efficiency. However, the model may face challenges in real-time implementation due to variations in lighting and background conditions.

Table 1 presents a concise comparison of recent studies on Sign Language Recognition, highlighting their methods, strengths, and limitations.

Table 1. Related works summary

S. No.	Author(s) & Year	Methodology	Advantage	Disadvantage
1	Gobhinath & Sophia (2025)	AGNN with GLCM-WAA and Savitzky–Golay denoising; optimized using Golden Eagle Optimization	Achieved 19.18–24.48% higher accuracy and reduced computation time	Limited real-time adaptability and performance in varying lighting
2	Hugar et al., 2025	CNN–HKP–LSTM–Transformer	Improves spatiotemporal recognition for low-resource languages	High complexity, limited gestures, and lighting sensitivity
3	Myagila et al., 2025	CNN–GRU with ELU activation	Effective temporal feature learning with a large dataset	Lower performance in signer-independent scenarios
4	Arunkumar et al. (2024)	Real-time sign-to-text/speech using CV and NLP integration	Promotes inclusivity with real-time translation	Struggles with accuracy in poor lighting and complex backgrounds
5	Siddique et al. (2023)	Detectron2, EfficientDet-D0, YOLOv7 trained on 3,760 images and deployed on Jetson Nano	Real-time low-cost system with high accuracy (up to 97%)	Lower accuracy during fast movements and low resolution
6	Prabhu et al. (2023)	CNN-based tool with video data preprocessing	Achieved 85% accuracy with mobile/web support	Affected by gesture variability and lighting issues
7	Priya et al. (2023)	Image processing with Canny edge and SIFT; 31 Tamil alphabets	Supports deaf-mute communication via visual processing	Poor handling of complex or overlapping gestures
8	Buttar et al. (2023)	LSTM + MediaPipe + YOLOv6 for dynamic and static signs	High accuracy (92–96%) and signer-independent recognition	Affected by background noise and lighting variations
9	Sharma & Singh (2022)	CNN with data augmentation using affine transformations	High accuracy across datasets with robust evaluation	Challenged in real-time due to environmental variations

From the literature review section, it has been observed that various researchers have already worked on recognizing sign language using ML and DL techniques. However, most of the existing techniques have limitations in that they are either computationally expensive or are based on American Sign Language and general sign language datasets.

Moreover, most techniques have limitations in that they are designed to improve recognition accuracy without considering hardware deployment issues such as memory usage and computational complexity.

Therefore, there is a need to propose a resource-efficient deep learning framework that can be used to attain accurate recognition with minimal computational overhead.

The proposed TamilNets-V1 framework has been designed to overcome the limitations of existing techniques by incorporating a computationally efficient EfficientNet-Lite model and Pelican-based hyperparameter tuning and quantization techniques.

3. TamilNetV1- Suggested Framework

Figure 1 demonstrates the suggested framework, Tamil NetV1, a Tamil Sign Recognition system (TSLR). As illustrated in Figure 1, the suggested framework aimed to design the TSLR model, which can be deployed in resource-constrained edge devices such as Raspberry Pi and even smartphones. The primary purpose of this framework is to identify the words from the signing Tamil gestures used in the real-time applications.

Hence, the foremost task is to divide the video files containing the Tamil gestures for its 13 words. After subdividing the videos, image frames are generated by the sampling techniques. Image frames are then pre-processed by the Hybrid CLAHE model, followed by a data augmentation process using the SMOTE-GAN technique.

The image augmented is then passed to the EfficientNet-Lite L0, which is quantized for hardware deployment. The detailed description of each and every component is as follows:

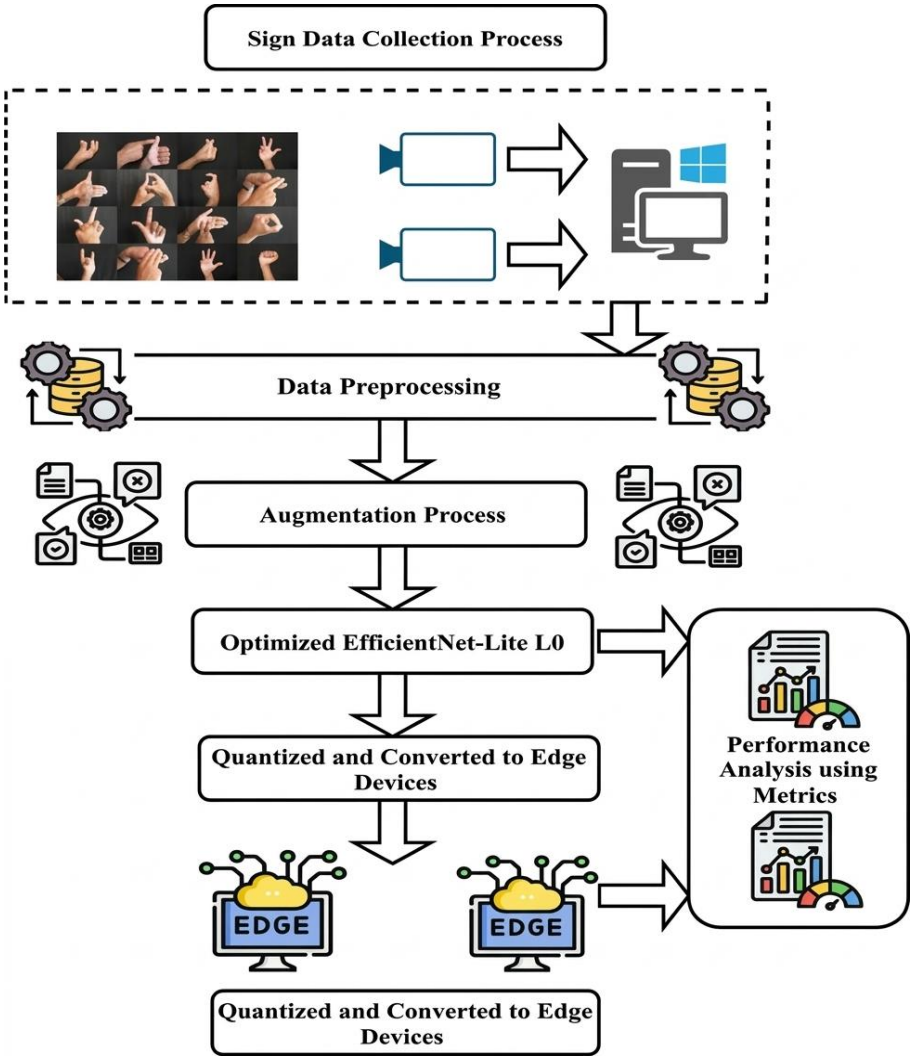

Fig. 1 TamilNet-V1 model for the tamil sign language recognition system

3.1. Tamil Sign Language Dataset Collection

The research employs the real-time Tamil Signs captured using the web-camera and stored in the SQLite. Nearly 50 volunteers participated in the experimentation process in which the 13 phrases of Tamil words were recorded using the camera.

The Tamil signs are used for this Experimentation. These recorded signs are validated using the Indian Sign Language Training and Research (ISLTR) Tamil Sign Dictionary [24] to create the corpus of Tamil Sign datasets.

Fig. 2 Sample datasets collected from the camera resembling the words in tamil sign language

3.2. Data Pre-processing and Augmentation Technique

The medical preprocessing technique is used to remove noise pixels, low -quality pixels that affect the detection of lung cancers. Adaptive Contrast Limited Adaptive Histogram Equalization (CLAHE) is utilized within this research for the pre-processing technique to remove the noise levels from the images and enhance the contrast in the images, which aids in a better classification process. Followed by a data pre-processing technique, the image augmentation process is adopted to overcome the class imbalance problem. As the class imbalance problem exists in the collected datasets, it is necessary to remove this problem since it creates the overfitting problem that affects the performance of recognition. SMOTE-GAN technique is adopted as the augmentation technique to overcome the problem of overfitting. After the Image augmentation process, 50 images have been increased to 2500 Images for each Tamil emotion.

Fig. 3 Pre-processed sign images by Adaptive -CLAHE model

3.3. Feature Extraction and Classification Networks

This section discusses the EfficientNet-B0 and B2, which act as backbone architectures, and also the EfficientLite Architecture for an effective classification of Tamil Sign Languages.

3.3.1. EffcientLite-B0 Model

Tan and Le [25] introduced an innovative convolutional neural network architecture known as EfficientNet, which enhances both performance and resource efficiency by uniformly scaling the model's depth, width, and input resolution using a compound scaling approach. Unlike traditional scaling methods, EfficientNet adjusts these three aspects simultaneously, optimizing model accuracy while maintaining manageable computational demands. The base version, EfficientNet-B0, derived from Neural Architecture Search (NAS), is designed for maximum architectural compactness and computational efficiency. Despite having only 5.3 million parameters, EfficientNet-B0 (illustrated in Figure 4) delivers outstanding classification performance, surpassing larger models like ResNet50 on the ImageNet benchmark, all while consuming less energy. Its structure includes Mobile Inverted Bottleneck Convolution (MBConv) layers, depthwise separable convolutions, and the SWISH activation function, enabling the extraction of detailed features with reduced processing overhead.

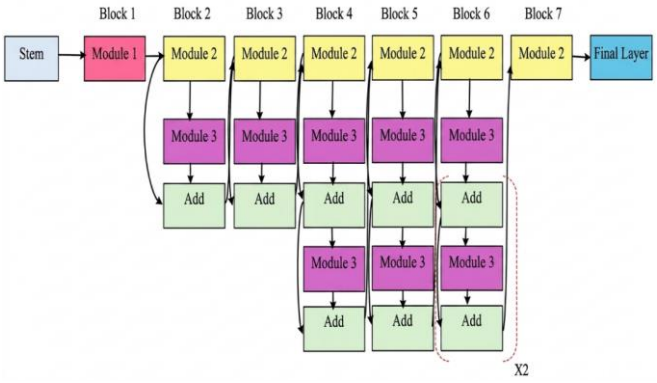

Fig. 4 EfficientNet-B0 model architecture

The EfficientNetB0 configuration outlines a streamlined and scalable network structure designed for image classification tasks, as shown in Figure 4. Serving as the foundational model in the EfficientNet series, EfficientNetB0 employs a compound scaling technique to uniformly adjust the model's depth, width, and input resolution. This design achieves a balance between performance accuracy and computational effectiveness. The architecture follows a modular format, with components sequentially labeled from Module 1 to Module 584, each representing elements such as convolution layers, activation units, or pooling mechanisms. This structured and layered setup supports optimal efficiency in terms of performance-to-resource ratio.

EfficientNetB2 builds upon the EfficientNetB0 base using the same compound scaling principle, increasing depth, width, and image resolution proportionally. Specifically, B2 processes inputs at a higher resolution of 260 × 260, expands convolutional channels, and deepens the network. These adjustments enable B2 to capture more intricate spatial features while maintaining efficiency. Though B2 contains

around 9.1 million parameters-slightly more than B0-it achieves improved accuracy, making it well-suited for applications needing precise detail recognition like medical imaging or fine-grained object classification. Despite its expanded capacity, B2 retains efficiency due to its optimized architecture and balanced scaling approach.

As depicted in Figure 5, EfficientNetB2 maintains a modular CNN framework with modules numbered from 1 to 586. Each unit reflects a distinct function within the network, such as convolution operations, activation layers, or pooling units.

This ordered module arrangement underlines the model's hierarchical and methodical structure, allowing it to deliver high accuracy while remaining computationally lean. EfficientNetB2 is essentially an enhanced version of B0, refined through extended scaling parameters to achieve greater precision and robustness.

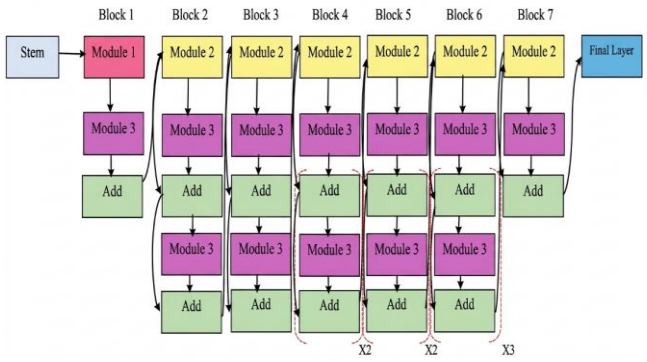

Fig. 5 EfficientNet-B2 model architecture

Recent variants of the EfficientNet series, specifically B0 and B2, have demonstrated superior performance across numerous computer vision benchmarks, particularly on datasets like ImageNet. Their notable accuracy, combined with minimal computational demand, makes them suitable for a broad spectrum of uses, ranging from lightweight mobile apps to extensive image classification systems.

Despite this, due to the nature of compound scaling, enhancements in performance remain bound to the base architecture, with any substantial improvement relying heavily on resource-intensive Neural Architecture Search (NAS). Nevertheless, EfficientNet-B0 and B2 stand out for their practical model scaling strategies and have shown success in delivering resource-conscious deep learning solutions across multiple fields.

In the suggested framework, EfficientNetB0 and EfficientNetB2 are chosen as they offer a well-balanced compromise between performance and hardware efficiency, making them ideal for constrained environments. EfficientNetB0 contributes with its lightweight, scalable structure, while EfficientNetB2 enhances prediction accuracy without imposing additional computational load.

The proposed system of TamilNets-V1 is based on a multi-step framework with the aim of achieving an efficient recognition of the sign language in the Tamil language with low computational complexity. In the first step, the resized and normalized gesture frames are input into the EfficientNet-LiteL0 architecture. EfficientNet-LiteL0 is used to effectively extract discriminative features from the hand sign input. EfficientNet-LiteL0 uses Mobile Inverted Bottleneck Convolution (MBConv) blocks that are effective in extracting spatial features from the gesture frames. Then, the feature representations are input into the classification stage, where the Pelican Optimization Algorithm is used to optimize the hyperparameters of the dense layers, thereby achieving an increased accuracy. Moreover, TensorFlow Lite is used in the proposed system, which reduces the model size and complexity. This allows the model to be deployed on platforms such as Raspberry Pi and smartphone devices.

3.3.2. EfficientLite-L0 Model

The EfficientLite-L0 model was created by slimming the EfficientNet-B0 model to increase its performance in resource-constrained devices such as edge devices and smart watches. The suggested network removes the Squeeze and Excite attention maps and replaces the Swish activation function with ReLU functions.

3.3.3. Hyperparameter Tuning Process

To find out the optimal set of parameters that can minimize the complexity as well as maximize the suitability of the models, hyperparameter tuning is carried out before moving on to the training phase. Hyperparameter tuning involves tuning various parameters such as the number of hidden layers, the number of neurons in each layer, dropout rate, total epochs, batch size, etc. To attain better classification accuracy with less computational overhead, various configurations of the network are optimized using advanced hyperparameter optimization techniques. In this study, the Pelican Optimization Algorithm is used to optimize hyperparameters for EfficientNet-LiteL0. The explanation of the proposed optimization technique is given in the following section.

3.4. Pelican Optimization Model

Pelicans are large birds recognized for their long beaks and the expandable throat pouch they use to catch and swallow prey. They usually reside in colonies of hundreds of people and are very gregarious. Adult pelicans typically have a wingspan of 0.5 to 3 meters, weigh between 2.75 and 15 kg, and stand between 1.06 and 1.83 meters tall. Although they occasionally eat frogs, turtles, crustaceans, and other seafood when they are very hungry, fish make up the majority of their diet. Pelicans frequently hunt together. Although some species hunt from lower altitudes, they may plunge from heights of 10

to 20 meters after seeing their prey. In order to force fish into shallow areas and facilitate their capture, they cooperate by spreading their wings over the water's surface. They tip their head forward to drain the extra water from their beak's pouch after catching a fish, and then they swallow their catch. The design of the proposed POA, which emulates this cooperative method, was inspired by the Pelican's intelligent and adaptable hunting activity, which is coordinated and efficient.

3.4.1. Mathematical Background of the Proposed Model POA

The suggested POA is a population-based optimization technique, with each Pelican standing in for a unique member of the population. Each person in these algorithms represents a possible solution to the optimization issue. Based on where they are in the search space, each member has a set of variable values. Equation (1) specifies the lower and upper boundaries of the problem variables, and these variables are initialized at random at the start of the operation.

$$x_{i,j} = l_j + rand \cdot (u_j - l_j), i = 1,2, \dots, N,\ j = 1,2, \dots, m, \tag{1}$$

where, '$x_{i,j}$' is the j^{th} variable specified by the i^{th} candidate solution, 'N' defines the population, 'm' is the number of problem variables, '$rand$' is a random number in the interval [0, 1], 'lj' and 'u_j' are the j^{th} lower and upper bounds of problem variables, respectively.

In the proposed POA, the pelican population is represented using a matrix referred to as the population matrix, as shown in Equation (2). Each row in this matrix represents a distinct candidate solution, while each column shows the values allocated to the variables in the optimization problem for that solution.

$$X = \begin{bmatrix} X_1 \\ \vdots \\ X_i \\ \vdots \\ X_N \end{bmatrix}_{N \times m} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,j} & \cdots & x_{1,m} \\ \vdots & \ddots & \vdots & \iddots & \vdots \\ x_{i,1} & \cdots & x_{i,j} & \cdots & x_{i,m} \\ \vdots & \iddots & \vdots & \ddots & \vdots \\ x_{N,1} & \cdots & x_{N,j} & \cdots & x_{N,m} \end{bmatrix}_{N \times m} \tag{2}$$

where 'X' → the population matrix of pelicans and 'Xi' is the *i*th Pelican. Each Pelican in the population serves as a potential solution to the optimization problem within the suggested POA. For each potential solution, the problem's objective function is assessed. Equation (3) states that the calculated objective function values are stored in a vector called the objective function vector.

$$F = \begin{bmatrix} F_1 \\ \vdots \\ F_i \\ \vdots \\ F_N \end{bmatrix}_{N \times 1} = \begin{bmatrix} F(X_1) \\ \vdots \\ F(X_i) \\ \vdots \\ F(X_N) \end{bmatrix}_{N \times 1} \tag{3}$$

where, 'F' → objective function vector and 'Fi' → the objective function value of the i_{th} candidate solution.

The suggested POA mimics pelican hunting techniques and behavior to update candidate solutions. There are two primary phases to this process:

1. Moving towards prey (phase of exploration)
2. Winging at the water's surface (phase of exploitation)

Phase 1: Exploration Phase: Approaching Prey During this phase, pelicans locate their prey by detecting its position and approaching it. This tendency is modeled to help the algorithm efficiently search the space, improving its capacity to find a variety of possible solution regions. One of POA's main characteristics is that the prey's location is determined at random inside the search space, which enhances its capacity to explore and find the best answers. Equation (4) is the mathematical explanation of this strategy for pelicans approaching prey.

$$x_{i,j}^{P_1} = \begin{cases} x_{i,j} + rand \cdot \left(p_j - I \cdot x_{i,j}\right), & F_p < F_i; \\ x_{i,j} + rand \cdot \left(x_{i,j} - p_j\right), & else, \end{cases} \tag{4}$$

where $x_{i,j}^{P_1}$ is the new status of the i_{th} Pelican in the j_{th} dimension on phase 1, 'I' is a random number (1 or 2), 'p_j' is the location of prey in the j_{th} dimension, and 'Fp' is its objective function value. For each member of the population, this parameter is selected at random in each cycle. The value of two leads the member to be more displaced, which enables it to explore new areas of the search space. Therefore, by facilitating a more exhaustive search of the solution space, parameter I significantly contributes to improving POA's exploration capability.

A pelican's updated position is only accepted in the suggested procedure if it produces a higher objective function value. This method, known as effective updating, keeps the algorithm from drifting into less-than-ideal areas. Equation (5) provides the updating mechanism's mathematical representation.

$$X_i = \begin{cases} X_i^{P_1}, & F_i^{P_1} < F_i; \\ X_i, & else, \end{cases} \tag{5}$$

where, $X_i^{P_1}$ is the new status of the i_{th} Pelican and $F_i^{P_1}$ is its objective function value on phase 1.

Phase 2: Winging on the Water Surface (Exploitation Phase)

Pelicans spread their wings to push fish toward shallower areas and then catch the prey in their throat pouch once they reach the water's surface. The probability of catching more

fish in the targeted region rises as a result of this cooperative behavior. The suggested POA can converge toward better solutions in promising areas of the search space by simulating this hunting technique. This stage improves the algorithm's capacity for local searches and for exploitation. In terms of mathematics, this phase entails assessing locations close to the Pelican's present location in order to arrive at a better answer. Equation (6) formulates the corresponding behavior.

$$x_{i,j}^{P_2} = x_{i,j} + R \cdot \left(1 - \frac{t}{T}\right) \cdot (2 \cdot rand - 1) \cdot x_{i,j}, \qquad (6)$$

Here, $x_{i,j}^{P_2}$ represents the updated position of the ith Pelican in the jth dimension during Phase 2. The constant R is set to 0.2, and the term R.(1−t/T) denotes the neighborhood radius around $x_{i,j}$, where t is the current iteration number, and T is the total number of iterations.

The local search region surrounding each population member is controlled by this coefficient. Because the coefficient value is higher at the beginning of the process, a bigger search area is possible. The neighborhood radius gets smaller as the iterations go on because the coefficient gets smaller. The algorithm's ability to converge toward the global or near-global optimal solution is enhanced by this progressive reduction, which makes it possible to do more accurate local searches close to each Pelican. Effective updating is used throughout this phase, which means that a new pelican position is only approved if it raises the value of the objective function. Equation (7) provides a mathematical expression for this technique.

$$X_i = \begin{cases} X_i^{P_2}, & F_i^{P_2} < F_i; \\ X_i, & else, \end{cases} \qquad (7)$$

where $X_i^{P_2}$ represents the new status of the i$_{th}$ Pelican and $F_i^{P_2}$ is its objective function value during phase 2.

3.5. Pseudo-Code of the Proposed Model POA

The algorithm assesses the new population states and their associated goal function values once every member of the population has been changed during the first and second stages. Then, the best solution discovered thus far is modified appropriately. Until the maximum number of iterations is achieved, the algorithm repeats the steps specified in Equations (4)-(7) in the subsequent iteration. The best solution found during the iterations is ultimately returned as the quasi-optimal solution for the specified optimization problem.

Algorithm 1:
Start POA

1. Get details of the optimization problem.
2. Set the population size N and the maximum number of iterations T.
3. Randomly initialize the positions of pelicans and compute their objective function values.
4. For each iteration t=1 to T:
5. Randomly generate the position of the prey.
6. For each Pelican i=1to N:
7. Phase 1 – Moving toward prey (Exploration):
8. For each dimension j=1to m:
9. Compute the updated value of the j$_{th}$ dimension using Equation (4).
10. End For
11. Update the position of pelican ‘i’ using Equation (5).
12. Phase 2 – Winging on the water surface (Exploitation):
13. For each dimension j=1to m:
14. Compute the updated value of the j$_{th}$ dimension using Equation (6).
15. End For
16. Update the position of pelican ‘i’ using Equation (7).
17. End For
18. Update the best solution found so far.
19. End For
20. Return the best solution obtained as the output of POA.

End POA

3.6. Pelican Tuned Deep Learning Networks (PTDLN)

The Pelican Optimization Model (POM) optimizes the weights of the dense neural networks that are used to construct the classification layers. The network initializes with hyperparameters chosen at random at the start of training. Equation (30) is employed to determine the fitness value for the suggested methodology. Algorithm 1 is utilized to update the hyperparameters during each iteration. When the calculated fitness value meets the requirement specified by Equation (8), the iteration is stopped.

$$Fitness\ Function\ < MSE(Predicted - Actual) \qquad (8)$$

The PTDLN layers allow for the quick and computationally effective identification of both normal and pathological cardiac states. The classification layers' operating flow is depicted in Algorithm 2. After extracting time-series features using self-attention-based GRU units, the model optimizes dense networks using the Pelican Optimization Model (POM) to achieve efficient multi-attack classification. 150 training epochs, 140 hidden nodes, and a momentum value of 0.01 are employed in the suggested configuration.

1. Input: Bias weights, hidden units, epochs, learning rate
2. Target: Prediction of Normal / Attack conditions
3. Randomly initialize bias weights, hidden units, epochs, and learning rate
4. Set the three parameters
5. While (True) do
6. Compute GRU cell outputs using Equation (9)
7. Evaluate the fitness function using Equation (8)

8. For t = 1 to Maximum Iterations do
9 Update bias weights and input layers using Equations (4)-(7)
10 Recalculate the fitness function using Equation (8).
11 If (fitness function == threshold) then
12 Go to Step 17
13 Else
14 Go to Step 8
15 End For
16 End While
17 Compute the output function from dense layers using Equation (10)
18 If (output function == 1)
19 // Normal condition detected
20 Else If (1 < output function <= 2)
21 // Heart disease detected
22 Else
23 Go to Step 9
24 End If
25 Stop

3.7. Hardware Deployment Process

The Pelican optimized EfficientNet-Lite L0 is quantized to convert it into a hardware deployable model so that it can be implemented in the different edge devices. The quantization technique is utilized in this research to scale the suggested networks for the hardware-supported models. The execution of the TamilNets-V1 framework is based on certain assumptions. The images of the gestures are assumed to be captured by a front-facing camera under moderate lighting conditions with a clear hand region. The system primarily focuses on recognizing static Tamil sign gestures for isolated words. Apart from that, it is also assumed that the dataset is representative of typical variations of gestures performed by different users. These assumptions are made for simplifying the recognition of the images and for the efficient execution of the model on resource-constrained edge devices.

4. Results and Discussions

4.1. Experimental Implementation

The system is implemented using an NVIDIA Tesla K40 GPU, TensorFlow v4 as the backend, and Keras as the high-level API, which was used to fully train the TamilNet V1 learning models. A CPU with 32 GB of RAM, a 2 TB hard drive, and an AMD Radeon processor operating at 3.0 GHz. Moreover, TensorFlow Lite is utilized to transform the trained model into a lightweight version suitable for deployment on resource-constrained devices. A Raspberry Pi Model 3B+ was used for deployment. To ensure a fair and unbiased evaluation of the suggested framework, the dataset was split into training and testing sets. For this purpose, an 80%-20% split of the dataset was performed. The proposed framework was trained for 150 epochs with a batch size of 32 using the TensorFlow framework. The training was performed using the stochastic gradient optimization technique. The Pelican Optimization Algorithm is employed to determine the optimal number of hidden nodes, learning rate, and dropout values. The Python-based application created for data gathering and testing is shown in Figure 6. Evaluation measures like accuracy, sensitivity, specificity, recall, and F1-score were calculated to gauge the efficacy of the suggested framework. The Tamil words utilized in the trials are listed in Table 2, and the mathematical methods for determining the metrics employed to assess the suggested architecture are given in Table 3.

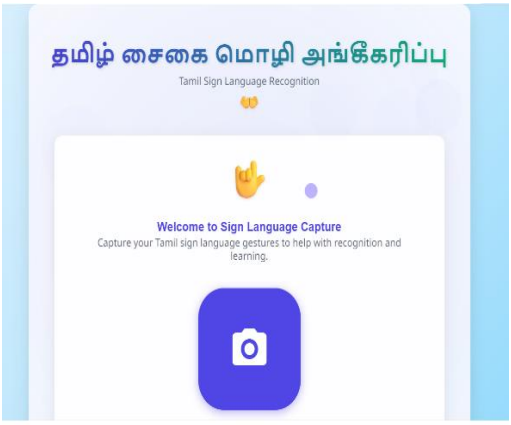

Fig. 6 Python application tool for gathering and testing the input data

Table 2. Tamil Sign Words Used for Experimental Analysis

words	Meaning
AFRAID	பயம்
APPRECIATE	பாராட்டுகிறேன்
BEAUTIFUL	அழகான
TRUST	நம்பிக்கை
WORRY	கவலை
CONGRATULATION	பாராட்டுகள்
DISAPPOINTED	ஏமாற்றம்
GRATEFUL	நன்றியுள்ள
PROMISE	வாக்குறுதி
SPEAK	பேசு
STUBBORN	பிடிவாதமான
TAKE CARE	கவனித்துக் கொள்ளுங்கள்
TRUTH	உண்மை

Table 3. Performance metrics used for evaluation of proposed framework

SL.NO	Performance Metrics	Mathematical Expression
1	Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$
2	Sensitivity or recall	$\frac{TP}{TP+FN}$ x100
3	Specificity	$\frac{TN}{TN+FP}$
4	Precision	$\frac{TN}{TP+FP}$
5	F1-Score	$2.\frac{Precison*Recall}{Precision+Recall}$

'TP is True Positive Values, TN is True Negative Values, FP is False Positive, and FN is False Negative Values'. By taking into account true positives, true negatives, false positives, and false negatives, these equations provide numerical metrics to assess the performance of stroke prediction models. Each parameter draws attention to a distinct facet of the model's performance, facilitating an analysis of its overall efficacy, accuracy, and dependability in forecasting stroke outcomes.

4.2. Experimental Result Outcomes

To evaluate the process of any Machine and Deep Learning process, confusion matrix and ROC-AUC characteristics are more often used. The figure shows the confusion matrix of the suggested model in detecting the Tamil Sign Language Signs, which belong to 13 Tamil words. From Figure 7, it is clear that the suggested framework has predicted high true positives and less false negatives. From this, it is clear that the suggested model has produced an accuracy of 99.75% in recognising the Tamil Sign Language. Further, the Experimental results demonstrate the efficacy of the proposed model by achieving an accuracy of 99.67%, a precision of 99.5%, a recall of 99.37%, and an F1-score of 99.6%, respectively. Moreover, the TamilNets-V1 consumes only 2.1 MB of memory size, 287,800 parameters, and an inference time of 114.56 seconds, and outperforms the other State-of-The-Art (SOTA) learning models.

Figure 8 demonstrates the performance of the proposed optimized model and existing EfficientNet-Lite L0 models in recognizing the Tamil Sign words. Though the existing model has produced promising performance, integrating the pelican optimization model in the EfficientNet-LiteL0 model has enhanced its accuracy from 96.7% to 99.5%, precision from 96.2% to 99%, recall from 96.0% to 99.7%, specificity of 96.0% to 99.6%, and F1-score of 96.2% to 99.5%, respectively. To prove the excellence of the proposed model, the performance of the model in TamilV1-net has been compared with other low computational learning frameworks, such as InceptionV5, EfficientNetB0, EfficientNetB2, and Resnet50 models. The figure shows the comparative analysis of the different models in recognizing the Tamil Sign words.

From Figure 9, it is clearly shown that the suggested model has outperformed the other models in detecting the Tamil Sign languages. Resnet-50 has produced the least performance, whereas EfficientNet-B0 and EfficientNet-B4 have produced good performance.

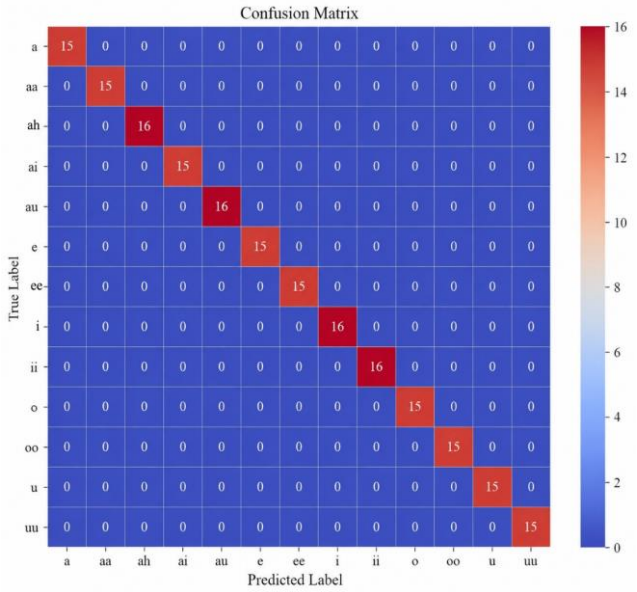

Fig. 7 Confusion matrix of the proposed model in detecting the tamil signs

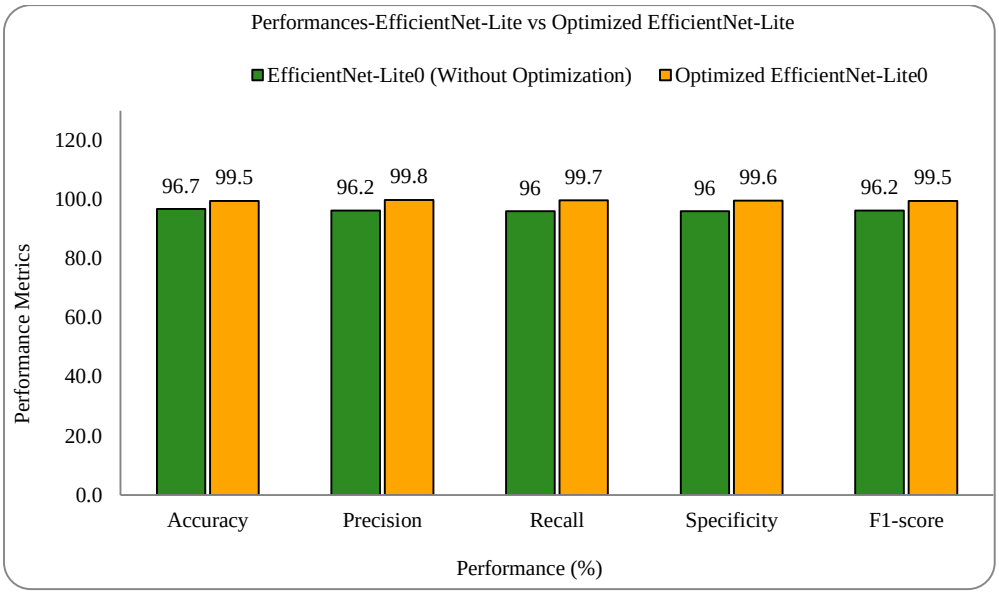

Fig. 8 Comparative analysis of the efficientnet-Lite L0 and Optimized EfficientNet-Lite model used for recognizing the tamil sign languages

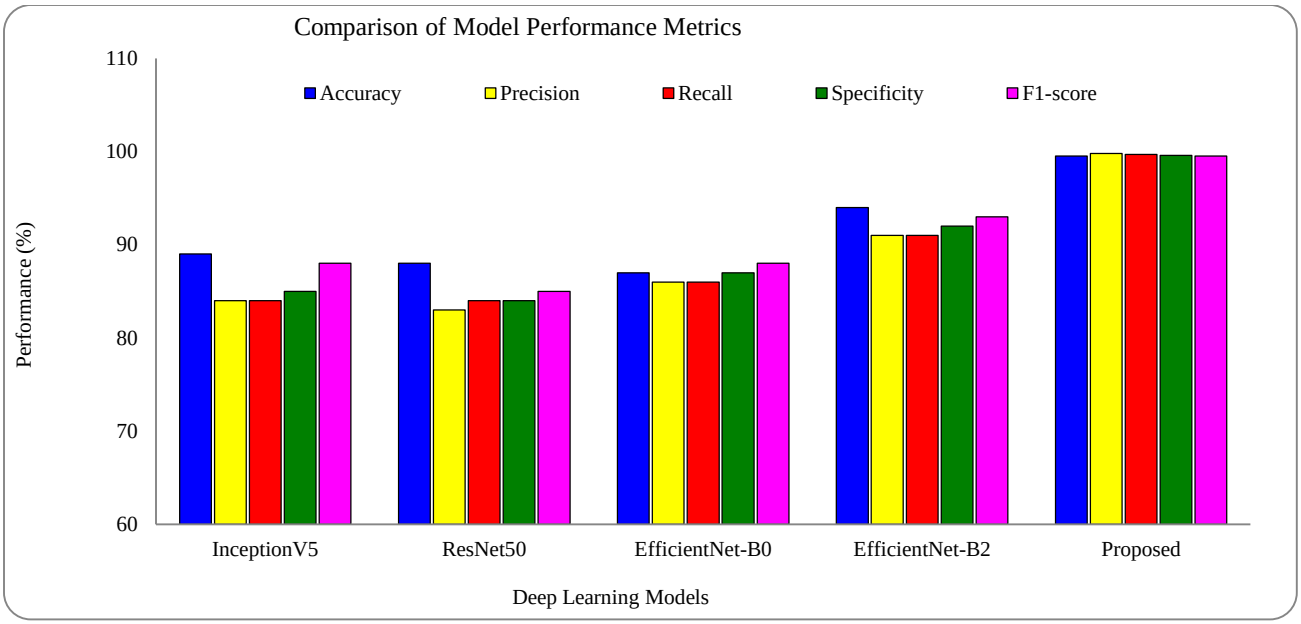

Fig. 9 Comparative analysis between the different models used for recognizing the tamil sign languages. (In CPU)

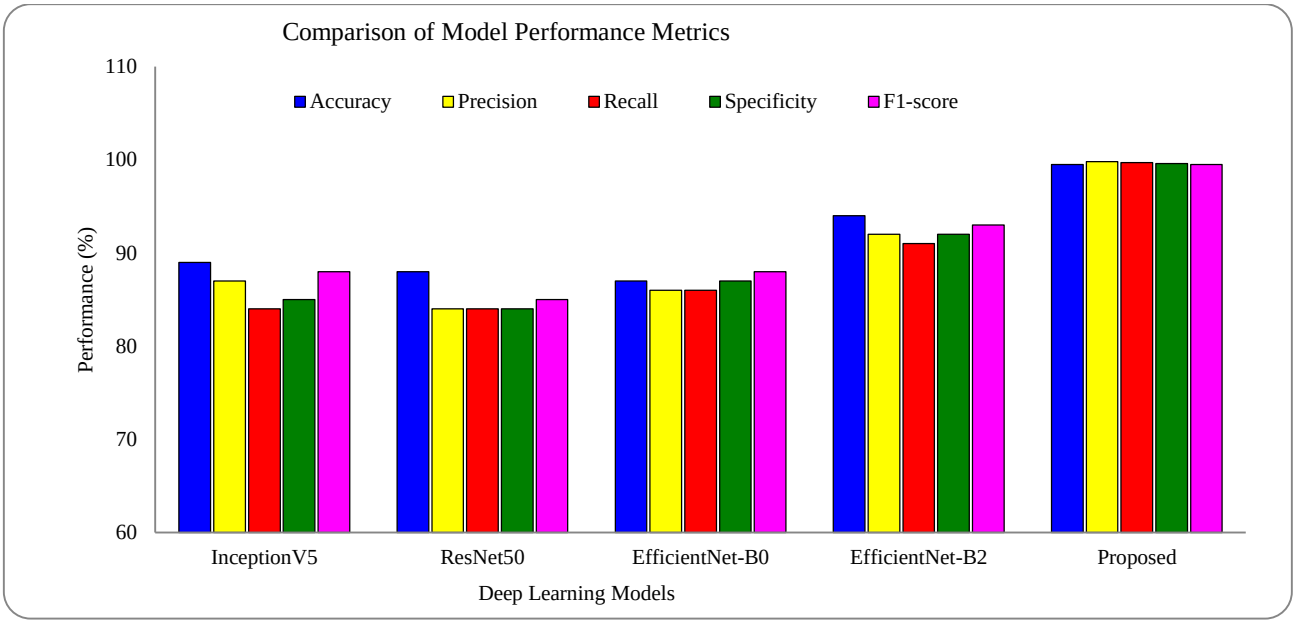

Fig. 10 Comparative analysis between the different models used for recognizing the tamil sign languages. (In Raspberry Pi Model 3B+)

Though these algorithms produce high accuracy rates, false rates are a little high when compared with the suggested framework. TamilNets-V1, which integrates the Pelican optimized EfficientNet-Lite L0 model, has given bright insights into recognizing the Tamil Sign Languages. On the other hand, the same performance metrics are calculated for the model running on an edge device (RaspBerry Pi Model B+), which is shown in Figure 10. In the context of practical and real-world scenarios, the sign language recognition system needs to address several challenges, such as lighting conditions, background complexity, hand orientation, and the speed of gestures.

In the proposed framework, the challenges are addressed by employing several preprocessing and data augmentation approaches. In particular, the Adaptive CLAHE preprocessing technique improves the contrast of the images and minimizes noise. This allows for more robust feature extraction for sign language recognition. Moreover, the SMOTE-GAN data augmentation approach increases the diversity of the dataset by generating synthetic gestures. This allows for robust feature extraction and increases the ability of the system to generalize and address different users and conditions. The EfficientNet-LiteL0 architecture also allows for robust feature extraction with low complexity.

4.3. Ablation Study

The ablation results prove the efficacy of the proposed optimization strategy as presented in Table 4. It is observed that the baseline EfficientNet-LiteL0 model performs well in recognition with a high accuracy rate of 96.7%. When Pelican Optimization is used to optimize existing models, such as lightweight models, it is observed that there is a marginal improvement in performance. The proposed TamilNets-V1 model that uses Pelican Optimization and EfficientNet-LiteL0 performs better in all evaluation metrics with a high accuracy rate of 99.5% and high precision, recall, and F1-score values. It is observed that Pelican Optimization and EfficientNet-LiteL0 together provide better results in sign language recognition for Tamil sign recognition performance.

Table 4. Baseline and comparative performance analysis

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
EfficientNet-LiteL0	96.7	96.4	96.2	96.3
Pelican Optimization+ existing Lightweight Models	97.8	97.5	97.3	97.4
Proposed	99.5	99.8	99.7	99.5

4.4 Statistical and Error Analysis

The statistical results from Tables 5 and 6 show that the proposed Lite model performs the best across all the performance metrics. Furthermore, it is also clear from the results that the proposed model possesses extremely low standard deviation values. This confirms that the recognition performance of the proposed model is stable. Based on the error analysis, it is clear from the results that the suggested Lite model possesses the lowest error rate of 0.5%, while the error rates of the InceptionV3, ResNet50, and EfficientNet-B0 models are 10.5%, 11.7%, and 12.8%, respectively. Although the error rate of the EfficientNet-B2 model is reduced to 5.9%, it is still higher than that of the proposed model. This confirms that the proposed model is effective in increasing the accuracy of the classification results while keeping the misclassification error rates low for the recognition of Tamil sign language.

Table 5. Statistical analysis between the different models used for recognizing the tamil sign languages. (In Raspberry Pi Model 3B+)

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
InceptionV3	89.5 ± 2.0	84.7 ± 2.2	83.5 ± 2.3	85.3 ± 2.1
ResNet50	88.3 ± 1.9	84.2 ± 2.1	82.4 ± 2.2	84.9 ± 2.0
EfficientNet-B0	87.2 ± 0.7	87.0 ± 0.8	86.5 ± 0.9	87.3 ± 0.7
EfficientNet-B2	94.1 ± 1.1	92.8 ± 1.2	91.4 ± 1.3	92.4 ± 1.2
Proposed Lite Model	99.5 ± 0.2	99.8 ± 0.2	99.7 ± 0.3	99.5 ± 0.2

Table 6. Statistical analysis between the different models used for recognizing the tamil sign languages (In CPU)

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
InceptionV3	89.5 ± 2.1	85.0 ± 2.3	83.2 ± 2.5	86.5 ± 2.2
ResNet50	88.3 ± 2.0	84.1 ± 2.2	82.3 ± 2.4	85.0 ± 2.1
EfficientNet-B0	87.2 ± 0.6	86.5 ± 0.7	85.6 ± 0.8	87.1 ± 0.6
EfficientNet-B2	94.1 ± 1.0	92.3 ± 1.1	91.2 ± 1.2	92.5 ± 1.1
Proposed Lite Model	99.5 ± 0.2	99.8 ± 0.2	99.7 ± 0.3	99.5 ± 0.2

Table 7 shows the computational metrics of the proposed model and other existing models in recognizing the Tamil Signs. This Experimentation was carried out by deploying the TamilNets-V1 in the CPU and Raspberry Pi Model 3 B+ with the configuration mentioned in Table 4. The parameters, such as Memory, FLOPS, and Inference time, are calculated and compared with the existing learning frameworks. From Table 7, it is found that the suggested networks have consumed less inference time, which is 10% lesser than the EfficientNet-B0 model, 50% lesser than Resnet-50,37.5% lesser than EfficientNet-B4 architectures, and 6.5% lesser than the traditional EfficientNet-Lite L0 model. Also, it is evident that the proposed model has lesser FLOPS (), less memory (), and even lesser parameters when compared to the existing model, making it ideal for edge devices. Table 4 also demonstrates the performance of the different models in Raspberry Pi Model 3B+ after applying the quantization technique suitable for the edge device. The similar performance of models in CPU is found in this edge device. As shown in the table, the proposed model consumes less inference time and a low memory size, which enhances the device's lifetime for deploying the Sign Language model in edge devices. The Experimentation clearly reveals that the proposed framework contributes substantially to the detection of Tamil Sign words with fewer computational resources and high performance-the model TamilV1-Net has been tested both in general-purpose

CPU and Embedded Raspberry Pi models. The lightweight characteristics of the proposed model, together with quantization, further help to ensure real-time deployment of the system, especially in real-world assistive scenarios.

Table 7. Computational parameters for the different models in both CPU and Raspberry Pi models

Algorithm	Computational Performance (CPU)				
	Inference Time (Secs)	**Memory (MB)**	**FLOPS(Secs)**	**Computational Time (secs)**	**No of Parameters**
ResNet-50	472.33	19.78	100	78.88	589,890
EfficientNet-B0	366.00	18.90	100	76.99	459,782
EfficientNet-B4	378.90	19.82	100	87.90	443,902
EfficientNet-Lite L0	121.80	2.19	65	34.78	322,768
InceptionV5	288.5	275.90	100	88.91	488,902
Optimized EfficientNet-L0(Proposed Model)	114.56	**2.09**	**56**	**27.89**	287,890
	Computational Performance (Raspberry Pi Model 3B+)				
ResNet-50	410.44	19.78	100	78.88	589,890
EfficientNet-B0	352.44	18.90	100	76.99	459,782
EfficientNet-B4	256.89	19.82	100	87.90	443,902
EfficientNet-Lite L0	112.9	2.10	65	34.78	322,768
InceptionV5	234.90	19.2	100	88.91	488,902
Optimized EfficientNet-L0(Proposed Model)	107.78	2.00	56	27.89	287,890

The reduced number of parameters, memory, and computational requirements of the TamilNets-V1 framework ensure that the system can operate effectively even on platforms with limited computing power, thus facilitating the creation of portable sign language recognition systems that can aid the deaf and speech-impaired community through mobile-based communication tools.

5. Conclusion and Future Direction

This research presents the TamilNets-V1, a novel optimized and Quantized EfficientNet-Lite L0 network for the recognition of Tamil Sign languages. The proposed network consists of the following components) Sign Tamil Data collection process 2) Data Pre-processing and Augmentation Process 3) Model Design 4) Quantised Model 5) Performance Evaluation. The performance of the model has been enhanced using the Pelican Optimized EfficientNet-LiteL0 algorithm. The hyperparameters of the model are tuned by using the Pelican Optimization algorithm to reduce the complexity overhead with increased performance. Finally, the model is quantized and deployed in the edge devices. The extensive Experimentation is carried out using the real-time sign gestures from 50 volunteers, and metrics such as accuracy, precision, recall, specificity, F1-score, FLOPS, memory, and inference time are calculated and analysed. To prove the superiority of the model, state-of-the-art learning models are considered for the Experimentation. Furthermore, the model is validated on CPU and Raspberry Pi Model 3B+ architectures.

Experimental results demonstrate the excellence of the proposed model in achieving the highest performance in detecting the Tamil Sign language.

As the future direction, the model needs more improvisation in terms of handling larger datasets and needs to be validated in multiple heterogeneous edge architectures.

The existing research is mainly concentrated on recognizing static Tamil sign language gestures related to individual words.

The recognition of continuous sign language, including the temporal motion of gestures, is generally carried out through sequence-based deep learning architectures like LSTM, GRU, and Transformer. Therefore, extending the proposed framework to support dynamic sign language gestures will be considered in the future.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

1. It is acknowledged with thanks that 50 volunteers have participated in the data collection of Tamil Signs.
2. It is acknowledged that the company " ABE Cognitive Solutions, Kancheepuram has supported us with data collection of Tamil Signs.

References

[1] S. Gokila, S. Rajeswari, and S. Deepa, "TAMIL-NLP: Roles and Impact of Machine Learning and Deep Learning with Natural Language Processing for Tamil," *2023 Eighth International Conference on Science Technology Engineering and Mathematics (ICONSTEM)*, Chennai, India, pp. 1-9, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[2] S. Rajendran et al., "Tamil NLP Technologies: Challenges, State of the Art, Trends and Future Scope," *Speech and Language Technologies for Low-Resource Languages: First International Conference, SPELLL*, Kalavakkam, India, vol. 1802, pp. 73-98, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[3] Hemakasiny Visuwalingam et al., "Deep Learning Model for Tamil Part-of-Speech Tagging," *The Computer Journal*, vol. 67, no. 8, pp. 2633-2642, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[4] K. Ponniah, and N.M.M. Safeek, "Exploring Theoretical and Conceptual Frameworks for the Enhancement of High-Order Thinking Skills through the Study of Thirukkural," *International Journal of Academic Research in Progressive Education and Development*, vol. 13, no. 2, 2024. [Google Scholar]

[5] Nelza Mara Pallu, Panchanan Mohanty, and Shiva Durga, "Thirukkural Translations: A Sacred Text from the Town of Peacocks—Mayilâpûr India," *International Journal of Development Research*, vol. 13, no. 5, pp. 62551-62553, 2023. [Publisher Link]

[6] B. Natarajan et al., "Development of an End-to-End Deep Learning Framework for Sign Language Recognition, Translation, and Video Generation," *IEEE Access*, vol. 10, pp. 104358-104374, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[7] Minerva Rivas Velarde, Caroline Jagoe, and Jessica Cuculick, "Video Relay Interpretation and Overcoming Barriers in Health Care for Deaf Users: Scoping Review," *Journal of Medical Internet Research*, vol. 24, no. 6, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[8] Hanaa ZainEldin et al., "Silent no More: A Comprehensive Review of Artificial Intelligence, Deep Learning, and Machine Learning in Facilitating Deaf and Mute Communication," *Artificial Intelligence Review*, vol. 57, no. 7, pp. 1-46, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[9] Ilias Papastratis et al., "Artificial Intelligence Technologies for Sign Language," *Sensors*, vol. 21, no. 17, pp. 1-25, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[10] D. Prabhu et al., "Machine Learning-based Tamil Sign Language Recognition Tool for the Deaf and Hard-of-Hearing Community," *2023 International Conference on Data Science, Agents and Artificial Intelligence (ICDSAAI)*, Chennai, India, pp. 1-6, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[11] Lihong Zheng, Bin Liang, and Ailian Jiang, "Recent Advances on Deep Learning for Sign Language Recognition," *2017 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, Sydney, NSW, Australia, pp. 1-7, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[12] Vidura Perera et al., "Tamil Alphabet Sign Language Recognition using Mediapipe and Machine Learning," *2024 9th International Conference on Information Technology Research (ICITR)*, Colombo, Sri Lanka, pp. 1-5, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[13] Pavel Trojovský, and Mohammad Dehghani, "Pelican Optimization Algorithm: A Novel Nature-Inspired Algorithm for Engineering Applications," *Sensors*, vol. 22, no. 3, pp. 1-34, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[14] Mohd Nadhir Ab Wahab et al., "EfficientNet-Lite and Hybrid CNN-KNN Implementation for Facial Expression Recognition on Raspberry Pi," *IEEE Access*, vol. 9, pp. 134065-134080, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[15] Rui-Yang Ju et al., "Efficient Convolutional Neural Networks on Raspberry Pi for Image Classification," *Journal of Real-Time Image Processing*, vol. 20, no. 2, pp. 1-11, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[16] S. Gobhinath, and S. Sophia, "Tamil Sign Language Identification using the Auto-Metric Graph Neural Network Optimized with Golden Eagle Optimization," *IETE Journal of Research*, vol. 71, no. 6, pp. 2010-2019, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[17] Gurusiddappa Hugar, Ramesh M. Kagalkar, and Abhijit Das, "Comparative Study of Hybrid Deep Learning Models for Kannada Sign Language Recognition," *International Journal of Computational Intelligence Systems*, vol. 18, no. 1, pp. 1-23, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[18] Kasian Myagila, Devotha Godfrey Nyambo, and Mussa Ally Dida, "Efficient Spatio-Temporal Modeling for Sign Language Recognition using CNN and RNN Architectures," *Frontiers in Artificial Intelligence*, vol. 8, pp. 1-12, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[19] N. Arunkumar et al., Sign Language Detection System using Machine Learning, IJCRT, vol. 12, no. 4, pp. i615-i618, 2024. [Online]. Available: https://www.ijcrt.org/papers/IJCRT2404987.pdf

[20] Sumaya Siddique et al., "Deep Learning-based Bangla Sign Language Detection with an Edge Device," *Intelligent Systems with Applications*, vol. 18, pp. 1-12, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[21] C. Bharathi Priya et al., "Tamil Sign Language Translation and Recognition System for Deaf-Mute People using Image Processing Techniques," *Applied and Computational Engineering*, vol. 8, no. 1, pp. 380-386, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[22] Ahmed Mateen Buttar et al., "Deep Learning in Sign Language Recognition: A Hybrid Approach for the Recognition of Static and Dynamic Signs," *Mathematics*, vol. 11, no. 17, pp. 1-20, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[23] Sakshi Sharma, and Sukhwinder Singh, "Recognition of Indian Sign Language (ISL) using Deep Learning Model," *Wireless Personal Communications*, vol. 123, no. 1, pp. 671-692, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[24] Indian Sign Language Research and Training Centre (ISLRTC), ISL Interpreters Directory, 2025. [Online]. Available: https://islrtc.nic.in/isl-interpreters-directory/

[25] Mingxing Tan, and Quoc Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," *Proceedings of the 36th International Conference on Machine Learning, PMLR*, vol. 97, pp. 6105-6114, 2019. [Google Scholar] [Publisher Link]