Multimedia Tools and Applications (2024) 83:45859–45880
https://doi.org/10.1007/s11042-023-17372-9

Fostering inclusivity through effective communication: Real-time sign language to speech conversion system for the deaf and hard-of-hearing community

Binwant Kaur[1] · Aastha Chaudhary[1] · Shahina Bano[1] · Yashmita[1] · S.R.N. Reddy[1] · Rishika Anand[1]

Received: 20 June 2023 / Revised: 25 August 2023 / Accepted: 1 October 2023 /
Published online: 18 October 2023

Abstract
In a world that values inclusivity, effective communication remains a cornerstone of empowerment for the deaf and hard-of-hearing community, with sign language serving as a pivotal means of expression. Yet, the limited proficiency in sign language among the general population underscores the need for innovative solutions like sign language translators. This study introduces a cutting-edge real-time sign language to speech conversion system, harnessing the power of a pre-trained InceptionResNetV2 deep learning model. To enhance accuracy, a bespoke American sign language dataset is employed, capturing 21 critical hand keypoints and sign images through Python libraries. The training dataset comprises 7200 images, categorized into 24 alphabet classes (excluding 'J' and 'Z'). Model refinement occurs over 20 epochs, each with a batch size of 16, culminating in remarkable training and validation accuracies of 98.23% and 97.07%, respectively. This impressive real-time sign language to speech conversion system, synergizing deep learning with Jetson Nano technology, paves the way for robust communication accessibility. Future advancements are envisaged, including expanding the system to support complete sentence translation and embracing diverse sign languages. By doing so, a comprehensive suite of sign language communication solutions will be offered, fostering universal understanding and inclusivity.

Keywords Sign Language Translator · Sign Language to Speech · Jetson Nano · InceptionResNetV2 · Deep Learning

1 Introduction

Effective communication is an essential aspect of human survival, enabling the conveyance of basic necessities such as clothing, shelter, food, and water. Beyond survival, communication serves as a means to socialize, express emotions, and provide mutual support. For individuals with speech and hearing impairments, sign language emerges as a vital mode

✉ Rishika Anand
rishika003phd19@igdtuw.ac.in

Extended author information available on the last page of the article

of communication, relying on intricate hand gestures and expressions. However, the majority of the population lacks proficiency in sign language, leading to a significant communication divide between these two communities.

This gap can be addressed through the use of interpreters, although this approach can be both costly and inconvenient. Alternatively, technology can provide a solution. Currently, two primary approaches are employed for recognizing sign language. One relies on sensors, involving specialized gloves and equipment to detect hand movements. The other approach is vision-based, using cameras to capture images and videos of individuals using sign language. The latter approach is more cost-effective and requires less hardware, making it the preferred choice for signers.

This study reviews the existing body of work related to sign language recognition, specifically focusing on the utilization of deep convolutional neural networks (CNNs) in conjunction with Jetson Nano. By interfacing a USB camera with Jetson Nano, sign language gestures are projected and captured. The captured sign video is then processed by Jetson Nano, which converts it into speech using a USB Sound Adapter. Jetson Nano is chosen due to its superior video processing capabilities compared to alternatives like Raspberry Pi. Balancing fast translation with hardware limitations inherent in small embedded systems is effectively achieved through Jetson Nano.

This research makes a significant contribution to the scientific community by proposing an innovative solution that integrates deep learning techniques, custom dataset creation, and real-time processing to address the critical issue of inclusivity. The system's robustness, potential for expansion, and potential societal impact all enhance the scientific significance of this research.

Challenges in employing vision-based methods for sign language recognition, as highlighted in [1], include:

- The non-universality of sign language, leading to regional variations.
- Capturing complete hand gestures within the frame during video or image capture.
- Establishing adequate background lighting to ensure clear hand movements.
- Accurately delineating sign boundaries to differentiate between adjacent signs.
- Dealing with co-articulation, where preceding or succeeding signs influence the interpretation of a sign.

1.1 The evolution of communication: Exploring sign language as a unique form of expression

Communication has been an intrinsic need since ancient times, facilitating interaction and the sharing of ideas. While verbal communication is the most prevalent method, an equally potent mode that intertwines both oral and written forms is sign language. It emerges as a natural linguistic expression, tailored for individuals facing unique challenges. The concept of sign language is far from new; historical research reveals its existence in ancient civilizations. Interestingly, it's proposed that sign language predates verbal communication, marking it as the primal human language for interaction. The signs are shown in Fig. 1.

Sign language possesses its distinct grammatical rules and structural intricacies. Its vocabulary is comprehensive and versatile, akin to any conventional language, enabling multifaceted communication. However, it's important to note that various sign languages are employed across different regions, each with its grammatical nuances and functions. Therefore, sign language isn't a universal standard. Prominent sign languages like British

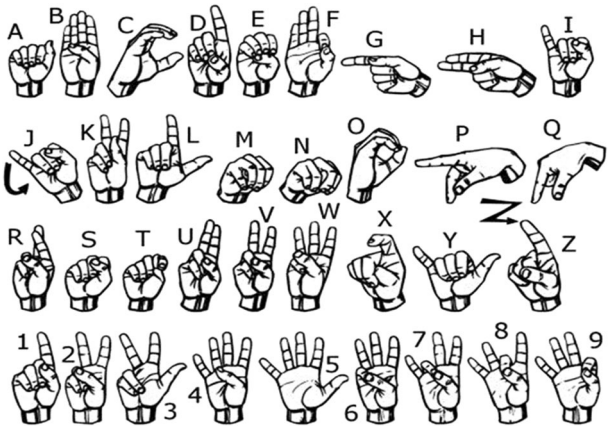

Fig. 1 English letters and numbers under American Sign Language [2]

Sign Language (BSL) and American Sign Language (ASL) have garnered recognition and development.

Comprising both manual and non-manual components such as hand orientation, shape, and motion, sign language thrives within a three-dimensional space. It can manifest through single-hand gestures, driven by the dominant hand, or more complex two-handed motions. Unlike spoken language, which follows a linear structure, sign language employs special configurations and movements, resulting in distinctive structural forms. Sign language sentences incorporate parameters like time, person, location, and predicate, albeit within constraints. This often leads to a limited vocabulary for individuals who are deaf or mute.

In summary, the rich history of communication has led to the development of sign language, a unique and intricate mode of expression. It serves as a testament to human adaptability and the diverse ways in which we connect and convey our thoughts.

1.2 The power of language: Empowering communication for the specially challenged

Language stands as the cornerstone of communication, a tool that transcends barriers and connects us all. Within India, a region with a population of approximately fifty lakhs facing hearing and speech challenges, the significance of this mode of interaction becomes even more profound. These challenges may arise from accidents or be present from birth, leaving many in this community underserved and often isolated. The inability to communicate effectively with the broader population can lead to a sense of loneliness and exclusion, impacting both personal and professional trajectories.

Sign language emerges as a pivotal tool in educating individuals with special needs. Creating targeted technology or systems that facilitate seamless communication for this community is imperative, enabling them to forge ahead in their lives. In today's rapidly evolving society, where success takes diverse forms, specially challenged individuals are carving their paths. Enabling their interaction with the general population has spurred ongoing research to develop systems capable of translating sign language into verbal or text form. This advancement is instrumental in bridging communication gaps between those who understand sign language and those who don't, fostering connections between specially challenged individuals themselves. The journey towards solving this challenge has led to ingenious solutions, including the use of sensors and sign language translators.

Such translators have proven indispensable in facilitating communication with deaf individuals. The realm of "Smart Talk" is one such innovative concept in development. This concept revolves around an electric glove embedded with sensors that discern hand gestures, ultimately leading to enhanced communication for the hearing-impaired community.

In essence, language holds the key to unlocking a world of possibilities. By fostering inclusive communication tools and technologies, we empower the specially challenged to transcend barriers, engage with society, and pave their paths towards fulfillment and success.

2 Revolutionizing sign language recognition: An overview of recent advances and challenges

In recent times, the fields of Machine Learning and Artificial Intelligence have experienced extensive applications across diverse domains. A particularly notable impact has been observed in the realm of Sign Language Recognition (SLR), profoundly enhancing communication possibilities for individuals with hearing impairments. This dynamic field has embraced various deep learning techniques, encompassing both vision-based and sensor-based methods, to identify and classify real-time sign language motions. This convergence of technology and accessibility represents both a challenge and a significant stride forward.

In the pursuit of efficient SLR systems, cutting-edge research endeavors have been instrumental. Notably, a prominent model [3] was devised employing an ASL alphabet database containing a staggering 87,000 hand gesture images with a resolution of 200×200 pixels. The meticulous process encompassed background correction, dataset partitioning, and rigorous model evaluation, culminating in an impressive accuracy of 99%. Subsequent improvements through image preprocessing could further amplify model performance.

In the context of Indian Sign Language, a breakthrough was achieved in [4] through the application of the Bag of Visual Words model. By employing skin tone-based segmentation and advanced feature extraction techniques, classification accuracies exceeding 99% were achieved using both Convolutional Neural Networks (CNN) and Support Vector Machines (SVM).

The recognition of Bangla Sign Language (BdSL) was addressed innovatively in [5], introducing real-time textual sentences and synthesized speech output. This comprehensive approach hinged on the use of YOLOv4 for object detection and a carefully curated dataset of 12.5 k BdSL images, showcasing an accuracy of 97.95%. In [6], researcher investigates using human movements to create MIDI music, exploring how this can enhance user experiences in different environments. The study presents a new algorithm implemented in a system that tracks movements in real time and converts them into music using a synthesizer. Through a user study, it shows that people's perceptions of their surroundings can be influenced by the sounds generated from their movements.

In [7], researcher aims to address the communication gap between hearing individuals and the deaf who use sign language. The researchers developed an interactive system to help people learn sign language more effectively. The study shows that this system increases user motivation and engagement in learning sign language, highlighting its potential to improve communication and awareness between these two groups.

Pioneering solutions such as an Arduino-based sign-to-speech conversion framework [8] demonstrated the fusion of hardware and software to enable hand or finger movement translation into sound. In [9], the recognition of Brazilian Sign Language words

was approached through a layered process, combining static hand shapes detection with dynamic gesture identification, achieving an accuracy of 80.19%.

Additional research has explored pose estimation [10], utilizing key joint locations to discern body posture and facilitate multi-view pose estimation and SLR. Custom CNN models have been trained from scratch, offering unique insights despite limitations with various skin colors and lighting conditions.

The quest for real-time SLR culminated in the development of a real-time American Sign Language finger-spelling translator [1], although challenges regarding dataset limitations and segmentation persist. Notably, the pioneering study [11] proficiently classified all 26 letters, including dynamic movements for the letters J and Z.

Furthermore, [12] introduced the "IISL2020" dataset for Indian Sign Language, achieving a remarkable F1 score of 97% through CNN and RNN models. The creation of custom datasets [4] has also demonstrated high accuracy in detecting ISL alphabets and numbers through tailored CNN and SVM architectures.

A multi-sign language approach [13] incorporated SVM, KNN, Random Forest, Decision Tree, Naïve Bayes, ANN, and MLP models, achieving impressive accuracies for various datasets. However, the ultimate vision extends beyond recognition, aiming to incorporate speech generation for seamless translation.

In conclusion, the strides in SLR research have ushered in a new era of accessibility, fostering connection and understanding for individuals with hearing impairments. With persistent challenges and innovative solutions on the horizon, the evolution of SLR continues to hold immense promise.

Examining the data presented in Table 1, it becomes evident that a significant portion of the research focuses on American Sign Language, predominantly involving static signs. Notably, the VGG16 pre-trained model emerges as a frontrunner, showcasing an impressive accuracy of 99.62%. Interestingly, some researchers have ventured into designing custom models, resulting in an even higher accuracy of 99.64%. An overarching trend among the existing studies is the recurrent utilization of Convolutional Neural Networks (CNN) as the primary algorithm. The appeal of CNN lies in its capacity to autonomously detect salient features, devoid of human intervention [14].

2.1 Unleashing the power of Jetson Nano: NVIDIA's AI computer for versatile AI applications

Embedded within the NVIDIA Jetson family, the Jetson Nano stands out as a compact yet robust computing solution tailored for driving entry-level AI applications and devices. This diminutive yet potent computer is uniquely equipped to facilitate the execution of multiple neural networks simultaneously. Designed with a focus on scalability and adaptability, the Jetson Nano developer kit is engineered for seamless integration into compact platforms [20].

Figuratively, Jetson Nano, depicted in Fig. 2, embodies an AI-driven powerhouse capable of orchestrating parallel deep learning neural networks while adeptly handling data inputs from numerous high-resolution sensors. Empowered by the JetPack Software Development Kit (SDK) and an array of versatile libraries, the Jetson Nano lends itself to an expansive array of applications spanning speech processing, image classification, segmentation, and object detection.

Table 1 Study of existing work on sign language recognition

Ref	Sign Language	Algorithm	Pre- Trained Models	Dataset	Sample Size	Detection	Simulated / Deployed	Accuracy
[15]	American Sign Language	CNN	VGG16	Self-Captured	40,000	Static	Simulated	99.62%
[16]	American Sign Language	CNN, RNN	Inception	American Sign Language Dataset	-	Static	Simulated	90.00%
[9]	Brazil Sign Language	HMM, HCRF, SVM, ANN	-	Self-Captured	139,154	Dynamic	Simulated	80.19%
[10]	American Sign Language	CNN	AlexNet like model	ASL image dataset(ASLID)	17,000	Dynamic	Simulated	-
[17]	American Sign Language	CNN	-	Self-Captured	25 images augmented	Static	Simulated	70.00%
[18]	American Sign Language	CNN	-	ASL Dataset from MNIST	27,455	Static	Simulated	93.00%
[1]	American Sign Language	CNN	GoogLeNet	ASL Finger Spelling Dataset from University of Surrey CVSSP	65,000	Real-time	Deployed	>70%
[11]	American Sign Language	CNN	VGG Net	Self-Captured	43,120	Real-time	Simulated	98.84%
[12]	Indian Sign Language	RNN, LSTM, GRU	inceptionResNetV2	IISL2020	11 words	Dynamic	Simulated	97.00%
[19]	Indian Sign Language	SVM, RBF, KNN	-	skin segmentation dataset from UCI	-	Static	Simulated	95.00%
[4]	Indian Sign Language	CNN, SVM	-	Self-Captured	36,000	Real-time	Deployed	99.64%
[13]	American, Indian, Italian, and Turkey Sign Language	SVM, KNN, Random Forest, ANN, MLP, Decision Tree, Naïve Bayes	-	Multiple datasets including existing and self-captured	156000, 4972, 12856, 4124	Real-time	Simulated	99.29%

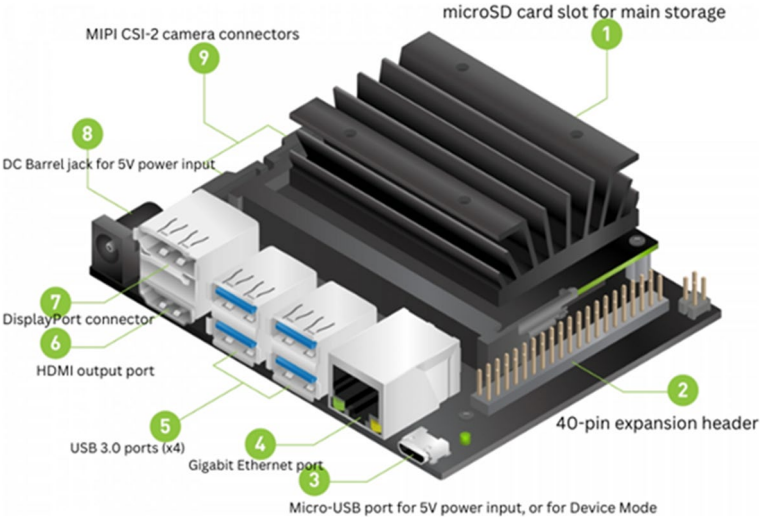

Fig. 2 Architecture of jetson nano [2]

By harnessing the prowess of CUDA cores and coupling it with Jetson Nano's GPU, a potent environment primed for application development emerges. The inherent heterogeneous architecture of the CPU-GPU tandem introduces a new dimension of capabilities. The CPU's capacity to be booted by the operating system further augments its utility. Additionally, the CPU's potential to accelerate complex machine learning tasks via programming [21] exemplifies the comprehensive potential of this integrated solution. In [22], the Jetson Nano Configuration Process Unveiled:

The process for configuring the Jetson Nano is meticulously detailed in [22], encompassing the following steps:

Operating system image flashing Begin by downloading NVIDIA's Jetpack 4.2 Ubuntu-based OS image and flashing it onto a microSD card using Bale-naEtcher. This tailored OS image empowers Jetson Nano to cater to diverse applications including deep learning, computer vision, and IoT. A 32 GB or 64 GB microSD card suffices for this task.

Initiating network connectivity For this phase, you'll need a collection of hardware components including the flashed microSD card, an HDMI screen, a mouse, the Jetson Nano dev board, a power supply, a USB keyboard, and a network connection. Internet access for Jetson Nano can be established via an Ethernet cable or a USB Wi-Fi adapter since it lacks built-in Wi-Fi capability. After assembling the components, insert the microSD card, connect the screen, keyboard, mouse, and network interface. Power up the system and configure the wired or wireless network settings upon the appearance of the NVIDIA + Ubuntu desktop.

Enabling remote access Once internet connectivity is established, establish a connection with Jetson Nano either through a terminal or via SSH.

System optimization Streamline the system by removing unnecessary programs to conserve space. Update the system and install essential system-level dependencies. Enhance the CMake precompiler tool to a newer version to facilitate the compilation of OpenCV.

OpenCV and dependencies Proceed to install system-level dependencies required for OpenCV. Employing a Python virtual environment is advisable, creating a self-contained space for Python software development and deployment. Within the virtual environment, install vital tools like TensorFlow, Keras, NumPy, and SciPy. Users can augment this environment further by incorporating other valuable libraries via pip.

Verification and testing Execute comprehensive tests to confirm the installation of libraries and system dependencies. This phase ensures the readiness of the Jetson Nano developer kit to undertake intricate computer vision and deep learning tasks [22].

This systematic approach empowers enthusiasts and developers to harness the potential of Jetson Nano for a variety of applications while ensuring optimal performance and functionality.

After a meticulous evaluation of various computational modules, as delineated in Table 2, our choice has decisively converged on the Jetson Nano. This decision stems from its adept balance between the imperatives of rapid translation and the constraints of limited memory and computational resources. Bolstered by a Quad-Core ARM®-based CPU and harnessing the formidable potential of the NVIDIA Maxwell™ architecture GPU, the Jetson Nano metamorphoses into a portable server replete with formidable capabilities.

The pivotal rationale for adopting the Jetson Nano lies in its ability to navigate the fine line between cost-effective pragmatism and an impressive CPU performance rate. Our pursuit necessitates an economically viable system that upholds robust performance standards, readily accessible as the need arises. Tailored to seamlessly integrate with the Internet of Things (IoT) paradigm, the Jetson Nano, akin in size to a mobile phone, introduces a realm of versatile deployment options, adaptable to any context where network connectivity prevails.

Significantly, this device facilitates the preliminary training of a dedicated deep learning model tailored to a specific signatory. Subsequently, this model can be seamlessly transferred onto their Jetson Nano device, culminating in a substantial enhancement of accuracy levels. The Jetson Nano embodies the convergence of practicality and capability, enabling us to advance our objectives efficiently and effectively.

Upon reviewing the contents of Table 3, it becomes evident that the majority of the equipment showcased compatibility with the Python programming language. Among the array of options, our discerning choice gravitates towards the Jetson Nano. This preference arises from its unique attribute of supporting multiple programming languages, adding a layer of versatility to its capabilities.

In addition to its language versatility, the Jetson Nano exhibits a distinct edge in terms of cost-effectiveness. While each option possesses its distinct merits, the Jetson Nano strikes an optimal balance between affordability and performance. Furthermore, its user-friendly nature enhances its appeal, as it offers a relatively uncomplicated operational experience. Ultimately, the decision to opt for the Jetson Nano aligns with our pursuit of a comprehensive solution that encompasses compatibility, affordability, and user-friendliness, enabling us to propel our endeavors forward effectively.

Table 2 Comparison of different computational modules

Equipment		Nvidia Jetson Nano [23]	Nvidia Jetson Xavier NX [24]	Raspberry Pi 4 [21]	Banana Pi BPI-M1 [25]	Orange Pi Plus [26]	Google Coral Dev Board [27]	Rock Pi N10 [28]	HiKey970 [29]	BeagleBone AI [30]
CPU	Model	ARM A57	8-core ARM	ARM Cortex-A72 64-bit	A20 ARM Cortex-A7	H3 Cortex-A7 H.265/HEVC 4 K	NXP i.MX 8 M SOC	RK3399Pro	ARM Cortex-A73	Intel Celeron J4105
	Cores	4	8	4	2	4	4	2	4	4
	Freq	1.43 GHz	2.26 Hz	1.5 GHz	1 GHz	1.5 GHz	1.5 GHz	1.8 GHz	1.8 GHz	1.5–2.5 GHz
GPU	Model	128-core Maxwell	512 Core Volta + NVDLA	Broadcom Video Core VI	A20 ARM Cortex-A7 Dual-Core2	Mali400MP2 GPU @600 MHz	Integrated GC7000 lite graphics	Mali T860MP4 GPU	ARM Cortex-A73 MPCore4	PowerVR SGX544 3D GPU
	Power	5W-10W	10/15/30 W	2.56–7.30W	1.25–2 W	1–10 W	4TOPS-2W	15–18 W	8–18 W	12.5 W
RAM		4 GB	8 GB	1 GB, 2 GB, 4 GB, 8 GB	1 GB	2 GB DDR3	1 or 4 GB LPDDR4 SDRAM	6 GB LPDDR & 32 GB eMMC 5.1	6 GB LPDDR4X 1866 MHz	6 GB LPDDR4 RAM
Ports		4 × USB 3.0 ports	(4x) USB 3.1 A(Host) USB 2.0 Micro B (Device)	2 USB 3.0 ports	USB 2.0 ports	USB 2.0 ports	USB 2.0/3.0 ports	USB Type-C PD 2	USB Type-C, USB 3.0 ports	Client Port USB 2.0
Camera		2 × MIPI CSI-2 DPHY lanes	16 lanes MIPI CSI-2 D-PHY 1.2 (40 Gbps)	MIPI CSI port	CSI Camera Connector	SM pixel camera sensor	MIPI CSI-2 camera	MIPI CSI	4 line MIPI port, 2 line MIPI port	USB Camera
Applications		Smart traffic control system	Detection of cucumber leaf diseases	IoT based smart mirror	Online test using design banana Pi and local network	Cloning vulnerability detection in driver layer of IoT devices	TensorFlow to detect objects In video streams	Early detection of brain cancer	Accelerating colorizer of shaded image for autonomous driving	Real-time management of railway administration

Table 2 (continued)

Equipment	Nvidia Jetson Nano [23]	Nvidia Jetson Xavier NX [24]	Raspberry Pi 4 [21]	Banana Pi BPI-M1 [25]	Orange Pi Plus [26]	Google Coral Dev Board [27]	Rock Pi N10 [28]	HiKey970 [29]	BeagleBone AI [30]
Connectivity	Gigabit Ethernet, Wireless networking adapter	Gigabit Ethernet, M.2 Key E (WiFi), M.2 Key M (NVMe)	Gigabit Ethernet, RJ45 cable	10/100/1000 Ethernet	10/100 Ethernet, RJ45 cable, USB WiFi adapter	Ethernet cable or Wi-Fi	Gb LAN, Ethernet port, Wi-Fi, Bluetooth	Gigabit Ethernet, GPS, PCIe gen2	Gigabit Ethernet, 2.4 GHz/5 GHz Wi-Fi
Cost	$99	$699	$55	$95	$85.99	$129.99	$127.49	$299	$58.99
OS	Linux4Tegra	Linux r35 codeline	Ubuntu Mate, Snappy Ubuntu Core, etc	Android, Linux, Beryboot, etc	ARM and Linux	Linux, Mac, Windows 10	Debian and Android 8.1	Android and Linux	Angstrom
Flash Memory	microSD card	microSD card	8 GB eMMC	8 GB eMMC	8 GB eMMC	8 GB eMMC	64 GB eMMC	64 GB UFS	16 GB eMMC

Table 3 Programming languages supported by different computational modules

S.No	Equipment	Supported Languages
1	Nvidia Jetson Nano	C/C++, Python, Java, JavaScript, Go, and Rust
2	Nvidia Jetson Xavier NX	C/C++, Python, Java
3	Raspberry Pi 4	Python
4	Banana Pi BPI-M1	Java, Python
5	Orange Pi Plus	Python, Scratch, C/C++
6	Google Coral Dev Board	C++, Python
7	Rock Pi N10	Python
8	HiKey970	Python
9	BeagleBone AI	C, C++, Python, Perl, Ruby, Java

3 Proposed work

The pivotal role of the NVIDIA Jetson Nano kit underpins the proposed architecture for sign language recognition. Unlike previous studies centered around Thai or Hong Kong sign languages, this research delves into American sign language. The objective is to craft a user-friendly system catering to individuals of diverse skill levels. Leveraging an existing CNN-based model, the system's development was streamlined. The model's accuracy was further refined through the meticulous creation and curation of an expansive dataset, encompassing diverse variations, to fuel rigorous training.

3.1 Methodology

This research is bifurcated into two core segments: sign language to text translation and text-to-speech conversion. With the aim to heighten accuracy beyond previous benchmarks, the chosen model is calibrated using the American Sign Language.

The progression of our model is delineated through the following steps [31]:

- Data Collection: Curating a comprehensive dataset of ASL signs and gestures.
- Data Pre-processing: Employing diverse machine learning techniques to preprocess the collected dataset.
- Model Implementation: Utilizing a pre-trained model and employing transfer learning for training. This phase encompasses model training and ASL symbol detection.
- Text Generation: The outcome of the aforementioned process culminates in a text rendition of the sign language.
- Text-to-Speech Conversion: Subsequently, the generated text is seamlessly transformed into speech or audio form.

3.2 Sign language to speech

Our endeavor encompassed the creation of a bespoke American Sign Language (ASL) image dataset, meticulously compiled through the real-time capture of hand gestures and signs. This dataset mirrors the complete spectrum of sign language structures,

encompassing a diverse range of variations. A paramount consideration was to accentuate differences, recognizing their role in bolstering model training and improving outcomes. The dataset was thoughtfully assembled by involving individuals with differing skin tones, hand sizes, and backgrounds. This multiplicity augments the dataset's robustness, which is further enriched by the inclusion of various backgrounds and graphical nuances, facilitating comprehensive model training.

Post-compilation, the dataset was judiciously divided into three distinct classes: training, validation, and testing. In the training and validation phases, meticulous labeling of images was executed. Leveraging the knowledge gleaned from these sets, the model underwent real-time testing, with accuracy as the quantifiable metric. Our model harnessed the potential of InceptionV2.

The outcome of the model materialized as a text representation of the provided signs, promptly displayed on the screen. A Python library named 'pyttsx3' played a pivotal role in transforming the predicted letters into audible speech. To enable auditory output, an audio card and speakers were seamlessly integrated with the Jetson Nano, allowing users to experience the final voice output firsthand.

3.3 Algorithms used in previous research

To equip a machine with the ability to learn, training becomes imperative. To ensure optimal performance, intelligent classification architectures are paramount. Below are some prevalent algorithms utilized for sign language detection:

- **CNN (Convolutional Neural Network):** This algorithm processes images, optimizing computational efficiency without compromising pertinent features essential for predictions [32]. Comprising three layers [33], the convolutional layer identifies pixel features, the pooling layer reduces dimensions while abstracting these features, and the fully-connected layer leverages an activation function to facilitate prediction. CNN has found application in various works, such as [1, 10, 17, 18], for sign language recognition.
- **SVM (Support Vector Machine):** For datasets with n dimensions, SVM plots each observation as a point in an n-dimensional space, aiming to discern a hyperplane that optimally classifies data points into respective categories [34]. In datasets with multiple dimensions and continuous features, SVM often exhibits superior performance [33]. Previous works, including [9, 19], and [13], have employed SVM for sign language recognition.
- **KNN (K-Nearest Neighbors):** KNN operates on the principle that proximate data points share similar attributes and consequently belong to the same class [23]. When predicting a new object's class, KNN selects k nearest neighbors to the object. For instance, with $k = 4$, KNN identifies the four closest neighbors to the current data point, thus predicting its class [33] [19] and [13] incorporate KNN in their endeavors.
- **HMM (Hidden Markov Model):** HMM is a probabilistic model featuring states, generating new states upon input. Transitions between states remain concealed, aligned with the model's name. Transition probabilities hinge on training and the model's proficiency, determined by the training sets. Effective training across all classes is pivotal for HMM's efficacy [33]. Many researchers have harnessed HMM to enhance recognition accuracy, including [9] and [35], especially in the context of continuous sign language recognition.

- **ANN (Artificial Neural Network):** An ANN comprises nodes and connections. Initial random weights assign to connections, impacting hidden node activations based on input signals and connections [36]. Output node activations are subsequently computed, and errors drive adjustments between hidden and output nodes. This iterative process persists until convergence criteria are met. [9, 13], and [37] have proposed ANN-based systems for recognizing a spectrum of sign language alphabets, spanning Brazilian, American, Indian, Italian, Turkish, and British.

In summary, these algorithms play pivotal roles in sign language recognition, each contributing distinctive capabilities to the broader field of research.

3.4 Architecture

Figure 3 illustrates the sequential progression adopted in this project. Commencing with data collection and compilation, an exclusive image dataset was meticulously curated. Subsequent to image preprocessing, a pre-trained model facilitated the training phase. Leveraging the trained model, real-time testing was conducted within the project's live environment. This holistic process transpired on the NVIDIA Jetson Nano, synergistically integrated with a web camera, audio card, and speakers.

3.5 Workflow

The commencement, depicted in Fig. 4, involves the activation of the web camera. Users will subsequently exhibit hand motions within its field of view. Real-time data from these gestures will be fed into the pre-trained model for immediate recognition.

In the event of accurate recognition, the corresponding output will be displayed on the screen, and the associated letter will be audibly articulated through the audio card and speakers. Conversely, if recognition falls short, the model initiates a fresh cycle of sign recognition for precise identification.

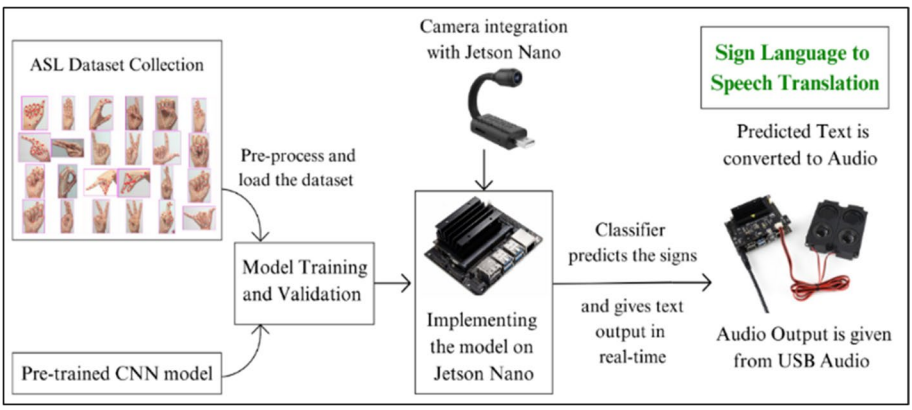

Fig. 3 Proposed Architecture

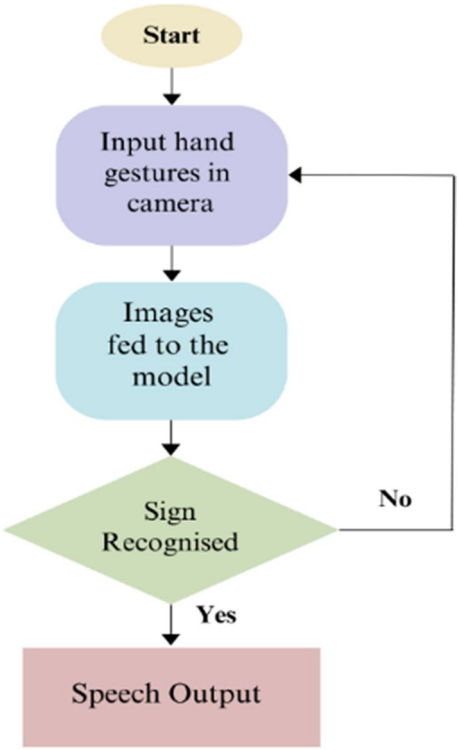

Fig. 4 Workflow of the system

4 Data and implementation

This section delineates the approach adopted for dataset collection and the subsequent implementation of the deep learning model. To enhance accuracy, the system underwent rigorous training and testing using a bespoke dataset. The pre-trained MobileNet V1 model was selected for implementation. The hardware deployment encompassed Jetson Nano equipped with the KHD CPU, renowned for its compatibility with both embedded applications and AIoT endeavors.

4.1 Data collection

The dataset was meticulously curated through manual collection and compilation efforts. A diverse group of individuals, each possessing unique attributes and hand characteristics, participated in this data collection process to enhance dataset variety. A distinctive aspect of our dataset lies in the inclusion of hand skeletons within the images. Utilizing the Python libraries CVZone and MediaPipe, the hand's detection was accomplished, along with the mapping of its key points and skeleton. The dataset comprises images depicting ASL symbols accompanied by the corresponding hand skeleton, a feature that significantly contributes to the model's improved learning capability. This dataset encompasses a total of 7200 images, distributed across 24 classes representing English alphabet symbols, excluding the letters 'J' and 'Z'.

4.2 Implementation

Based on various prior studies, Convolutional Neural Networks (CNNs) emerge as the prevalent choice for sign language detection. Employing transfer learning, the system harnesses pre-trained models and retrains them with freshly amassed datasets. The advantage of transfer learning lies in the ability to leverage a model's past learning while adapting it to new data, ultimately fostering heightened accuracy.

The adopted pre-trained model in this context is "Inception-ResNet-v2" [38], which has undergone training with the extensive ImageNet dataset encompassing over a million images. With 164 layers, this convolutional neural network has mastered the classification of images across 1000 distinct object categories, presenting intricate feature representations across diverse visual content. Blending the Residual connection and Inception structure, the Inception-ResNet architecture unites multiple convolutional filters of varying dimensions with residual connections, effectively addressing the challenges of deep network performance degradation and accelerating training processes [39].

The Inception-ResNet model ingests a dataset of 7200 images, categorized into 24 classes, and these data are partitioned into 80% for training and 20% for validation. Through training, the model optimizes its accuracy using the Adam optimizer and categorical cross-entropy loss. The training employs a batch size of 16 and spans 20 epochs.

Upon successful model training, the trained model is implemented on the Jetson Nano, establishing a real-time detection system. The system recognizes signs, displaying the corresponding alphabetic representation. The subsequent phase entails converting the text output into speech. Leveraging the Python library "pyttsx3," the model's text output post-symbol prediction transforms into real-time audio output. Integration with speakers connected to the Jetson Nano facilitates the immediate delivery of audio output. Consequently, the system proficiently accomplishes the conversion of sign language into spoken language.

5 Result

By utilizing a pre-trained Inception-ResNetV2 model and training it on a dataset comprising 7200 images across 24 classes, commendable training and validation accuracies of 98.23% and 97.07%, respectively, were attained. The application of the Categorical Cross Entropy Loss Function resulted in a calculated loss of 0.550%, reinforcing the promising outcomes within the context of dataset size and time limitations. The model's accuracy and loss progression throughout its training course are visually represented in Figure 5(a) and (b), which demonstrates performance over 20 epochs with a batch size of 16.

The study further presents the confusion matrix, depicted in Figure 6, which offers insights into the model's classification performance. Additionally, the system's ability for real-time detection of sign gestures corresponding to letters 'A,' 'B,' and 'X' is visually showcased in Figures 7, 8, and 9, respectively. These visualizations collectively highlight the robustness and efficacy of the developed system in recognizing and interpreting sign language gestures.

Table 4 below presents a comparative analysis of previous and prospective research endeavors concerning the utilization of diverse machine learning models for the

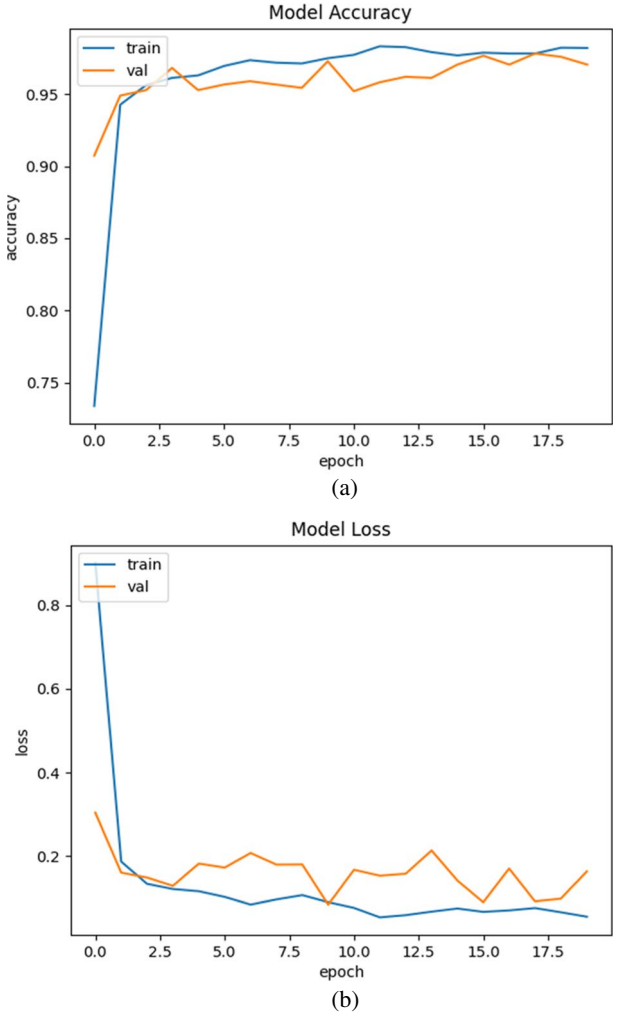

Fig. 5 (**a**) Accuracy graph of the proposed model. (**b**) Loss graph of the proposed model

conversion of sign language to text. In the current study, the journey commences with the application of a pre-trained InceptionResNetV2 model, subsequently fine-tuned through the integration of a self-captured sign language dataset to enhance recognition accuracy.

Of notable significance is the prevalent deficiency of racial diversity within the datasets employed in earlier investigations, influencing the predictive precision of the models. To address this limitation, the proposed study integrates a hand tracking module during dataset collection. This strategic addition empowers the model to adeptly interpret hand gestures by comprehending pivotal hand keypoints, thereby enabling real-time communication of their intended meanings.

Operating on the compact yet potent Jetson Nano embedded system, the proposed system embodies the pinnacle of convenience for sign language recognition applications. The

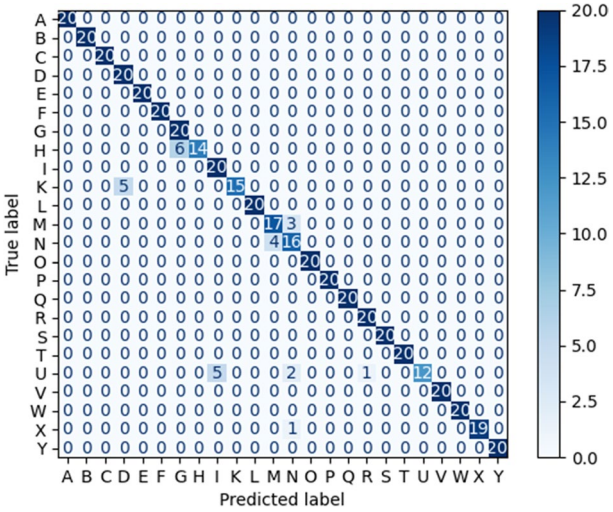

Fig. 6 Confusion matrix

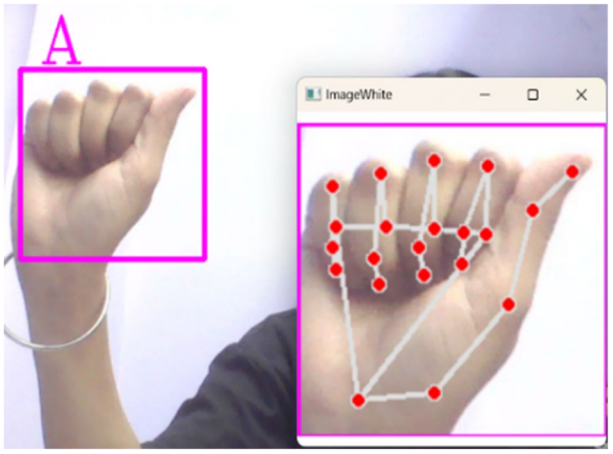

Fig. 7 Recognition of the alphabet 'A'

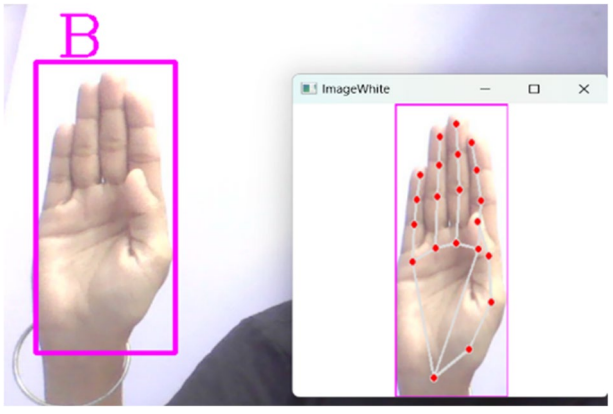

Fig. 8 Recognition of the alphabet 'B'

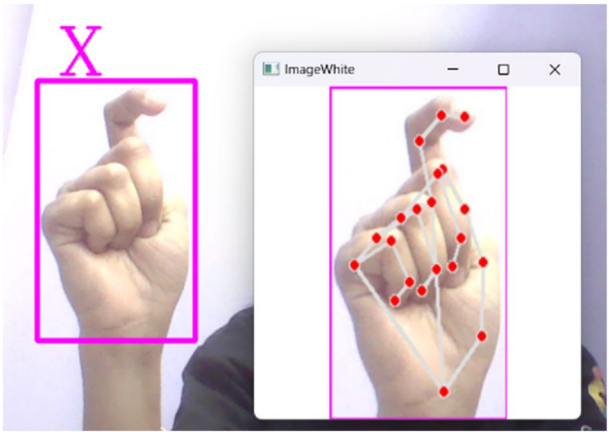

Fig. 9 Recognition of the alphabet 'X'

Jetson Nano's robust processing capabilities serve as the central processing unit for real-time sign language recognition, facilitating rapid and accurate interpretation.

6 Conclusion

In this study, a novel approach to convert sign language into speech is presented, leveraging the capabilities of a CNN-based deep learning model. The model exhibits remarkable proficiency by accurately classifying 24 distinct alphabetic symbols, yielding an impressive training accuracy of 98.23% and a validation accuracy of 97.07%. A noteworthy advantage of the proposed system is its elimination of the laborious and error-prone manual feature extraction process, replacing it with an automated feature extraction mechanism via CNNs.

Utilizing the portable embedded system Jetson Nano, the system attains a versatile performance across models of varying complexity. The integration of a USB camera, audio card, and speakers empowers real-time translation of sign language. The dataset collection process, facilitated by the CVZone package and OpenCV library, ensures robust 3D hand keypoints within the images. By addressing previous research gaps, including lighting variations and dataset diversity, the system achieves enhanced robustness.

With approximately 70 million individuals globally relying on sign language as their primary means of communication, this system emerges as a vital tool to bridge the communication divide between signers and non-signers. Its adoption of a lightweight deep learning model with advanced convolutional layers enhances efficiency and cost-effectiveness. Moving forward, there exists potential for expanding the dataset to encompass a broader array of sign language gestures, further refining the model's accuracy and usability.

Funding This research received no financing from any commercial, public or non-profit organization.

Table 4 Previous vs proposed research on Sign Language Recognition

Ref	Sign language	Algorithm	Pre-trained models	Dataset	Sample size	Accuracy
[18]	American Sign Language	CNN	GoogLeNet	ASL FingerSpelling Dataset from the University of Surrey's CVVSP	65,000	> 70%
[1]	American Sign Language	CNN	VGG Net	Self-Captured	43,120	98.84%
[11]	Indian Sign Language	CNN, SVM	-	Self-Captured	36,000	99.64%
[19]	American, Indian, Italian, and Turkey Sign Language	SVM, KNN, Random Forest, Decision Tree, Naïve Bayes, MLP, ANN	-	Multiple datasets including existing and self-captured	156,000, 4972, 12,856, and 4124	99.29%
[12]	Bengali Sign Language	CNN	-	Self-captured	1000	98.2%
[4]	American Sign Language	ANN, K-convex hull	-	Self-captured	1850	94.32%
[13]	Turkish Sign Language	LR, KNN, RF, ANN	-	Self-captured	87,000	98.97%
[20]	American Sign Language	ANN	-	Self-captured	520	96.15%
Proposed	American Sign Language	CNN	InceptionResNetV2	Self-captured	2400	97.07%

Data availability On request, the dataset used to support the findings of this study can be obtained from the corresponding author.

Declarations

Conflict of interest There are no potential conflicts of interest for the authors to disclose.

References

1. Alarcon G, Brandon VS (2016) Real-time american sign language recognition with convolutional neural networks 2. Convolutional Neural Netw Vis Recogn 8:225–232
2. Bukhari J, Rehman M, Malik SI, Kamboh AM, Salman A (2015) American sign language translation through sensory glove; SignSpeak. Int J u- e-Serv Sci Technol 8(1):131–142. https://doi.org/10.14257/ijunesst.2015.8.1.12
3. Triwijoyo BK, Karnaen LYR, Adil A (2023) Deep learning approach for sign language recognition. JITEKI: Jurnal Ilmiah Teknik Elektro Komputer dan Informatika 9(1)
4. Katoch S, Singh V, Tiwary US (2022) Indian Sign Language recognition system using SURF with SVM and CNN. Array 14. https://doi.org/10.1016/j.array.2022.100141
5. Talukder D, Jahara F (2020) Real-time bangla sign language detection with sentence and speech generation. In ICCIT 2020 - 23rd International Conference on Computer and Information Technology, Proceedings, Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ICCIT51783.2020.9392693
6. Breve B, Cirillo S, Cuofano M, Desiato D (2022) Enhancing spatial perception through sound: mapping human movements into MIDI. Multimed Tools Appl 81(1):73–94. https://doi.org/10.1007/s11042-021-11077-7
7. Pietro B, Di Gregorio M, Romano M, Sebillo M, Vitiello G, Solimando G (2020) Sign language interactive learning-measuring the user engagement. In Learning and Collaboration Technologies. Human and Technology Ecosystems: 7th International Conference, LCT 2020, Denmark: Springer, pp 3–12
8. Yash J, Pooja S, Pradnya P, Jyoti W (2017) Sign language to speech conversion using arduino. Int J Innov Eng Res 2(1):37–44. https://www.researchgate.net/publication/339973280_SIGN_LANGUAGE_TO_SPEECH_CONVERSION_USING_ARDUINO
9. de Souza CR, Pizzolato EB (2013) Sign language recognition with support vector machines and hidden conditional random fields: going from fingerspelling to natural articulated words. In: Machine learning and data mining in pattern recognition: 9th International Conference, MLDM 2013, New York, NY. Proceedings, vol 9. Springer Berlin Heidelberg, pp 84–98
10. Gattupalli S, Ghaderi A, Athitsos V (2016) Evaluation of deep learning based pose estimation for sign language recognition. In ACM International Conference Proceeding Series, Association for Computing Machinery.https://doi.org/10.1145/2910674.2910716
11. Amer Kadhim R, Khamees M (2020) A real-time american sign language recognition system using convolutional neural network for real datasets. TEM Journal:937–943. https://doi.org/10.18421/TEM93-14
12. Kothadiya D, Bhatt C, Sapariya K, Patel K, Gil-González AB, Corchado JM (2022) Deepsign: Sign language detection and recognition using deep learning. Electronics (Switzerland), 11(11). https://doi.org/10.3390/electronics11111780
13. Tayade A, Halder A (2021) Real-time vernacular sign language recognition using mediapipe and machine learning. Int J Res Publ Rev 2(5). https://doi.org/10.13140/RG.2.2.32364.03203
14. Dertat A (2013) Applied deep learning - part 4: convolutional neural networks, Medium. https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2. Accessed 16 May 2023
15. Thakur A, Budhathoki P, Upreti S, Shrestha S, Shakya S (2020) Real time sign language recognition and speech generation. J Innov Image Process 2(2):65–76. https://doi.org/10.36548/jiip.2020.2.001
16. Bantupalli K, Xie Y (2018) American sign language recognition using deep learning and computer vision. In 2018 IEEE International Conference on Big Data (Big Data), IEEE, pp 4896–4899. https://doi.org/10.1109/BigData.2018.8622141
17. Bheda V, Radpour D (2017) Using deep convolutional networks for gesture recognition in American sign language. arXiv preprint arXiv:1710.06836

18. Sabeenian RS, Sai Bharathwaj S, Mohamed Aadhil M (2020) Sign language recognition using deep learning and computer vision. J Adv Res Dyn Control Syst 12(5 Special Issue):964–968. https://doi.org/10.5373/JARDCS/V12SP5/20201842
19. Shirbhate RS et al (2020) Sign language recognition using machine learning algorithm. Int Res J Eng Technol. [Online]. Available. http://www.irjet.net. Accessed 15 May 2023
20. Nano J (2022) Developer kit. NVIDIA Developer. https://developer.nvidia.com/embedded/jetson-nano-developer-kit. Accessed 23 May 2023
21. Suzen AA, Duman B, Sen B (2020) Benchmark analysis of jetson TX2, Jetson Nano and raspberry PI using deep-CNN. In 2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), IEEE, pp 1–5 https://doi.org/10.1109/HORA49412.2020.9152915
22. Rosebrock A (2020) How to configure your Nvidia jetson Nano for computer vision and deep learning. physiol image search. https://pyimagesearch.com/2020/03/25/how-to-configure-your-nvidia-jetson-nano-for-computer-vision-and-deep-learning/. Accessed 16 May 2023
23. Shin DJ, Kim JJ (2022) A deep learning framework performance evaluation to use YOLO in Nvidia jetson platform. Appl Sci (Switzerland) 12(8). https://doi.org/10.3390/app12083734
24. Jetson modules, support, ecosystem, and lineup. NVIDIA Developer (2023) https://developer.nvidia.com/embedded/jetson-modules. Accessed 16 May 2023
25. Banana Pi B-M, Banana Pi (2023) Wiki. https://en.wikipedia.org/wiki/Banana_Pi. Accessed 17 May 2023
26. Lencse G, Répás S (2016) Benchmarking further single board computers for building a mini supercomputer for simulation of telecommunication system. Int J Adv Telecommun Electrotech Signals Syst 5(1). https://doi.org/10.11601/ijates.v5i1.138
27. Coral, Dev Board, Google (2020) https://coral.ai/products/dev-board/. Accessed 25 May 2023
28. Rock Pi, 10 designed for AI apps and solutions based on, Aliexpress (2020) https://www.aliexpress.com/item/1005002921148955.html. Accessed 25 May 2023
29. Linaro HIKEY970 (2023) https://www.96boards.org/product/hikey970/. Accessed 26 May 2023
30. TI.com, BEAGLE-3P-BBONE-AI BeagleBone® AI AM5729 development board for embedded Artificial Intelligence (2023) https://www.ti.com/tool/BEAGLE-3P-BBONE-AI. Accessed 26 May 2023
31. Zhou Z, Neo Y, Lui KS, Tam VWL, Lam EY, Wong N (2020) A portable hong kong sign language translation platform with deep learning and jetson Nano. in ASSETS 2020 - 22nd International ACM SIGACCESS Conference on Computers and Accessibility, Association for Computing Machinery, Inc. https://doi.org/10.1145/3373625.3418046.
32. Gavrilova Y (2021) What are convolutional neural. Serokell Software Development Company. https://serokell.io/blog/introduction-to-convolutional-neural-networks. Accessed 31 May 2023
33. Adeyanju IA, Bello OO, Adegboye MA (2021) Machine learning methods for sign language recognition: a critical review and analysis. Intell Syst Appl 12:56. https://doi.org/10.1016/j.iswa.2021.20
34. Saxena S (2021) Beginner's guide to support vector machine (SVM), analytics vidya. https://www.analyticsvidhya.com/blog/2021/03/beginners-guide-to-support-vector-machine-svm/. Accessed 30 May 2023
35. Aloysius N, Geetha M (2020) A scale space model of weighted average CNN ensemble for ASL fingerspelling recognition. Int J Comput Sci Eng 22(1):154. https://doi.org/10.1504/IJCSE.2020.107268
36. Srivastava T (2020) How does Artificial Neural Network (ANN) algorithm work? Simplified!, Analytics Vidya. https://www.analyticsvidhya.com/blog/2014/10/ann-work-simplified/. Accessed 30 May 2023
37. Raj RD, Jasuja A (2018) British sign language recognition using HOG. In 2018 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS), IEEE, pp 1–4. https://doi.org/10.1109/SCEECS.2018.8546967
38. Elhamraoui Z (2020) InceptionResNetV2 – Simple introduction, Medium. https://medium.com/@zahraelhamraoui1997/inceptionresnetv2-simple-introduction-9a2000edcdb6. Accessed 16 May 2023
39. Szegedy C, Ioffe S, Vanhoucke V, Alemi AA. Inception-v4, Inception-ResNet and the impact of residual connections on learning." [Online]. Available. http://www.aaai.org

Authors and Affiliations

Binwant Kaur[1] · Aastha Chaudhary[1] · Shahina Bano[1] · Yashmita[1] · S.R.N. Reddy[1] · Rishika Anand[1]

Binwant Kaur
binwant030btcse19@igdtuw.ac.in

Aastha Chaudhary
aastha040btcse19@igdtuw.ac.in

Shahina Bano
shahina032btcse19@igdtuw.ac.in

Yashmita
yashmita027btcse19@igdtuw.ac.in

S.R.N. Reddy
srnreddy@igdtuw.ac.in

[1] Computer Science and Engineering, IGDTUW, Delhi, India 110006