KEC Conference 2024
ISSN: 3021-940X (Print)
DOI: https://doi.org/10.3126/injet.v1i2.66704

American Sign Language Classification using CNNs: A Comparative Study

Samrat Kumar Adhikari[1,*], Pranav Neupane[2], Shyama Mainali[3], Utsarga Regmi[4], Pralhad Chapagain[5]

[1]*Department of Computer and Electronics Engineering, Kantipur Engineering College, Dhapakhel, Lalitpur, Nepal, samratmetaladhikari@gmail.com*
[2]*Department of Computer and Electronics Engineering, Kantipur Engineering College, Dhapakhel, Lalitpur, Nepal, neupanepranav27@gmail.com*
[3]*Department of Computer and Electronics Engineering, Kantipur Engineering College, Dhapakhel, Lalitpur, Nepal, mainalishyama@gmail.com*
[4]*Department of Computer and Electronics Engineering, Kantipur Engineering College, Dhapakhel, Lalitpur, Nepal, regmiutsarga7@gmail.com*
[5]*Department of Computer and Electronics Engineering, Kantipur Engineering College, Dhapakhel, Lalitpur, Nepal, pralhadchapagain@kec.edu.np*

Abstract

American Sign Language (ASL) classification is crucial in facilitating communication for individuals with hearing impairments. Traditional methods rely heavily on manual interpretation, which can be time-consuming and error-prone. Inspired by the success of deep learning techniques in image processing, the paper explores the application of Convolutional Neural Networks (CNNs) for ASL classification. The paper presents a CNN architecture tailored specifically for this task and investigates the effectiveness of transfer learning by leveraging four pre-trained models: VGG16, InceptionV3, ResNet50, and DenseNet121. A comparative analysis of these architectures has been presented in this paper. The experimental results show that the customized CNN model outperformed other models with a testing accuracy of 99.93% when provided with testing set images. Consequently, it is concluded that customized CNN outshines other models in accurately classifying sign languages.

Keywords: American Sign Language, Deep Learning, Convolutional Neural Network, Transfer Learning, Image Classification, Image Processing

1. Introduction

Communication is crucial for everyone, but individuals with speech and hearing challenges often struggle to express themselves and understand others, leading to feelings of isolation. Sign language is manual communication commonly used by people who are deaf. Sign language is not universal. People who are deaf from different countries speak different sign languages (AccessComputing, n.d.). To address this issue, researchers have developed hand sign recognition systems, enabling communication through hand gestures Initially relying on basic features like color and shape for sign and gesture detection, these systems have evolved significantly since their inception in the late 20th century. The advancements in computer vision and deep learning have propelled hand sign recognition forward. The Customized Neural Networks (CNNs) excel at learning patterns from visual data, enabling them to interpret hand gestures accurately by analyzing both shape and movement. Unlike older methods that were manual and less adaptable, CNNs learn and improve with training, making hand sign recognition more efficient. These advancements promise to enhance the quality of life for individuals with speech and hearing impairments by providing them with more accessible means of communication. This paper presents a comprehensive study of a customized CNN architecture and four pre-trained CNN models. The primary objective of this study is to evaluate the performance of these

**Corresponding Author*

models and determine the model with the highest accuracy in classifying hand signs from the video frame from a live camera.

2. Related Works

The research on Hand Sign Recognition utilizing CNN (Bhavana et al., 2021) employed a tailored CNN model achieving a 75% accuracy in classifying hand signs. For background elimination, the HSV color space was utilized to ensure the model's effectiveness across diverse environments. The HSV color mode was used to extract hand regions in a paper published in 2015 (Noreem et al., 2015). HSV (Hue, Saturation & Value) color space separates color information from brightness, enabling the extraction of the hand region based on its distinctive color characteristics, such as skin tone, while disregarding variations in lighting conditions.

The paper published in 2021 (Harris et al., 2021) with the title illustrated the use of Mediapipe library to isolate hand region from an image based on the hand-landmarks. Mediapipe provides a framework for machine learning-based hand gesture recognition, enabling the extraction of hand regions from images through sophisticated computer vision techniques, thereby enhancing interactivity and usability in applications like user guides.

The paper published in 2018 (Mesbahi et al., 2018) presents a method for hand gesture recognition that combines background subtraction with convexity defect detection. By eliminating irrelevant information through background subtraction and utilizing convexity defects for feature extraction, the proposed approach ensures reliable gesture recognition invariant to translation and rotation.

Transfer learning is gaining popularity in image classification, where a pre-trained neural network is repurposed for a new task. This method leverages the learned features of the pre-trained model, typically trained on a large dataset like ImageNet, and fine-tunes it for a specific classification task with a smaller dataset. By utilizing this existing knowledge, the model can adapt more quickly and effectively to the new task, leading to improved performance and reduced training time compared to starting training from scratch.

A research paper titled “Sign language recognition: A comparative analysis of deep learning models” published in 2022 (Premkumar et al., 2022), demonstrated a comparative study of a customized CNN and VGG16 models. The paper concluded that VGG16 was better where it delivered an accuracy of 99.56%, followed by customized CNN with an accuracy of 99.38%.

Hand Gesture Recognition (HGR) dataset was used in a paper (Hussain et al., 2023), where InceptionV3 and EfficientNet-B0 models were implemented. In this experimental study, InceptionV3 model achieved 90% accuracy with precision, recall and f1-score of 0.93%, 0.91%, and 0.90% respectively whereas EfficientNet-B0 achieved 99% accuracy with precision, recall and f1-score of 0.98%, 0.97%, 0.98% respectively.

A 2-level ResNet50 was used in a paper title “Sign Language Recognition Using ResNet50 Deep Neural Network Architecture” (Rathi et al., 2020), where the model achieved an accuracy of 99.03% on 12,048 test images.

DenseNet is a CNN architecture that has become widely popular and is extensively used in image processing tasks. A research paper published in 2021 (Marjusalinah et al., 2021) performed comparative study of ResNet50 and DenseNet121 on American sign language dataset. The dataset was split into training and testing set with the ratio of 80:20. ResNet50 achieved an accuracy of 0.999913 with recall, precision, and f1-score of 0.998966, 0.998958, and 0.999913 respectively. The same train-test ratio was used to train and test the DenseNet121 model where it achieved an accuracy of 0.998872 with recall, precision, and f1-score of 0.987116, 0.986458 and 0.998872 respectively.

The ASL dataset sourced from the Modified National Institute of Standards and Technology (MNIST) database was used for a research paper (Mohsin et al., 2024), where detailed comparison between VGG16, ResNet50, MobileNetV2, InceptionV3 and customized CNN. The paper illustrated that VGG16 and InceptionV3 stand out among the stated models with an accuracy of 0.95 and 0.96 respectively.

Despite the notable progress in the field, there exists a significant research gap that demands attention. The proposed study, titled "**American Sign Language Classification: A Comparative Study**" seeks to address this gap by investigating the effectiveness of transfer learning in ASL recognition, leveraging pre-trained models to improve accuracy. This research aims to contribute by analyzing how transfer learning can enhance

ASL recognition using both pre-existing and custom models, comparing their performance in testing scenarios. Additionally, the study aims to tackle key challenges in ASL recognition, particularly in real-time scenarios, dataset diversity, and variations in hand movements and positions. By exploring the complexities of real-time ASL recognition, the study aims to develop methodologies ensuring precise and timely interpretation in practical settings. Through these efforts, the proposed study aims to advance the efficiency and applicability of ASL recognition systems, ultimately fostering greater accessibility and inclusivity for individuals with hearing impairments.

3. Methodology

The dataset titled “ASL Alphabet” was acquired from Kaggle (AKASH, 2018), a renowned website for data science and machine learning notebooks and datasets. The model was trained using Google Colab due to its provision of free GPU services which provide cost-effective access to accelerated computing resources.

3.1. Dataset Description

The acquired dataset has a training set and a testing set subfolders. The training set subfolder was used for training, validation, and testing. The training folder is composed of 29 classes, including 3 classes designated for 'del', 'space' and 'nothing', and 26 classes representing the English alphabet. Each class contains 3000 images, resulting in a total of 87,000 images. However, ‘del’ and ‘nothing’ classes were removed for this research. The 'nothing' class was excluded as empty images were deemed unnecessary as Mediapipe is utilized to track hand movements. Similarly, the 'del' class was excluded from the experiment to maintain focus on the specific objectives of the research in American Sign Language (ASL) recognition, avoiding unnecessary gestures that could introduce noise or confusion into the training and evaluation process.
The training subfolder was divided into sets for training, validation, and testing, with a ratio of 68:17:15. The split ratio allocates more data to validation and testing sets, mitigating the risk of overfitting, a common concern in deep learning tasks.

Figure 1. Image samples from each class

Figure 1 depicts images of hand signs across different classes, showcasing variations in brightness and distance from the camera. Upon visual inspection of the sample images in the dataset, notable similarities between a few signs have been observed. For instance, signs like V and K exhibit visual resemblances in their hand configurations.

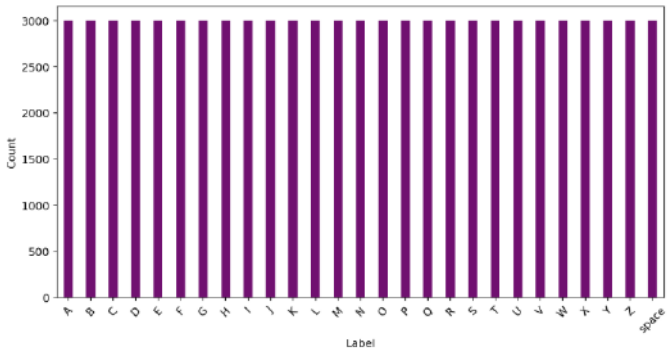

Figure 2. Distribution of images in each class

The above figure indicates that each of the 27 classes comprises 3000 images, ensuring a balanced dataset. Balanced datasets are essential as they mitigate the risk of overfitting and bias towards classes with many images.

3.2. Preprocessing

Images in the training set were reduced to 128x128x3 to minimize computational complexity while maintaining the necessary information. By standardizing pixel values between 0 and 1, normalization was used to improve data integrity and decrease redundancy.
Preprocessing techniques were applied to live images taken from a camera to classify hand signs in real-time. The Hand-tracking module from Mediapipe was used to extract the hand region from every frame. Furthermore, hand contours were utilized to generate background subtraction masks, guaranteeing adaptability to different environments. The resulting images were resized to 128x128x3, and pixel values underwent normalization to match the training set.

3.2.1. Mediapipe library for hand region extraction

Mediapipe is an open-source framework developed by Google which is used to build pipelines that perform computer vision inference over arbitrary sensory data such as video or audio. Media pipe is used to detect hand-landmarks with the help of its hand-tracking module. The hand-tracking model detects 21 key points on the hand. These key points track the hand and crop the hand region from the frame. The model was trained on more than 30k real-world images over varying backgrounds. (MediaPipe)

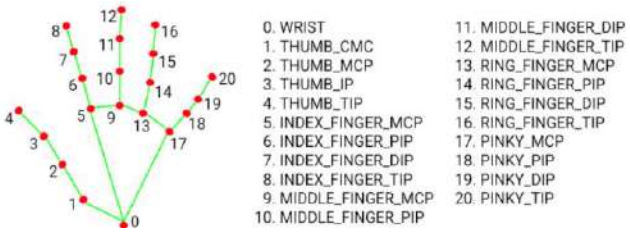

Figure 3. Illustration of hand-landmarks using Mediapipe

3.2.2. Background subtraction using hand contours

Contours are continuous lines formed by connecting points along the boundary of the hand, determined by its color or intensity in the HSV color mode, a technique initially applied to the image. These contours serve as a description of the hand's shape and can be employed as a mask for background subtraction, isolating the hand from its surroundings. Utilizing contours as a guide, background subtraction algorithms can accurately segment the hand from the background, facilitating tasks such as gesture recognition or object tracking in

computer vision applications. The mask derived from contours can be seamlessly overlaid onto the cropped hand image, eliminating the background. This process ensures a clean separation between the hand and its surroundings, enhancing the focus on the hand region for further analysis or visualization purposes.

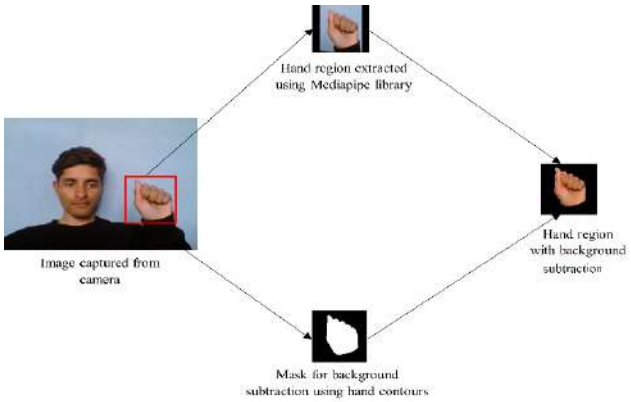

Figure 4. Hand region segregation using Mediapipe and hand contours

3.3 Model Selection

Convolutional Neural Networks (CNNs) are the preferred choice for image-processing tasks. CNNs are feed-forward neural networks that are well known for their capacity to learn feature representations on their own by optimizing filters or kernels. The basic architecture of a CNN consists of multiple layers, including convolutional layers, pooling layers, and fully connected layers. In the convolutional layers, the network applies filters to the input image to extract features such as edges, corners, and textures. The pooling layers are used to reduce the spatial dimensions of the feature maps while the fully connected layers are used to classify the input image based on the extracted features. CNNs are trained using large datasets of labeled images, and they use backpropagation to adjust the network weights to minimize the error between the predicted and actual labels. The training process involves several iterations of forward and backward propagation, where the network learns to recognize patterns and features in the input images. CNNs are exceptional at extracting complex patterns and characteristics from images, which makes them very useful for a variety of tasks like object identification, image segmentation, and image classification. (Gurucharan, 2022)

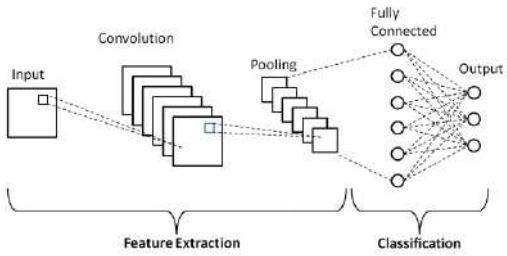

Figure 5. CNN architecture

3.3.1 Customized CNN

Customized CNNs are tailored neural network architectures for specific tasks or datasets. These networks are often modified by adjusting the number of layers, the types of layers used, or the layer parameters to improve performance on a particular problem. Customization can involve adding, removing, or altering layers such as convolutional, pooling, or fully connected layers. Additionally, the network's hyperparameters, such as learning rate, batch size, and regularization, may be adjusted to optimize performance. Customized CNNs are widely used in various applications, including image classification, object detection, and facial expression recognition, to achieve superior results on specific tasks or datasets.

Table 1. Applied Customized CNN architecture

Layer (type)	Input Shape	Output Shape	Param #
Conv2D	(None, 128, 128, 3)	(None, 124, 124, 32)	2432
MaxPooling2D	(None, 124, 124, 32)	(None, 62, 62, 32)	0
Dropout	(None, 62, 62, 32)	(None, 62, 62, 32)	0
Conv2D	(None, 62, 62, 32)	(None, 60, 60, 64)	18496
MaxPooling2D	(None, 60, 60, 64)	(None, 30, 30, 64)	0
Dropout	(None, 30, 30, 64)	(None, 30, 30, 64)	0
Conv2D	(None, 30, 30, 64)	(None, 28, 28, 64)	36928
MaxPooling2D	(None, 28, 28, 64)	(None, 14, 14, 64)	0
Dropout	(None, 14, 14, 64)	(None, 14, 14, 64)	0
Flatten	(None, 14, 14, 64)	(None, 12544)	0
Dense	(None, 12544)	(None, 128)	1605760
Dense	(None, 128)	(None, 27)	3483

The architecture of the model has been developed through experimentation with various numbers of layers and parameters to achieve optimal performance. It consists of three convolutional layers, followed by three pooling layers and two fully connected layers. The input layer takes input images of size 128x128x3 and applies 32 filters with a 5x5 kernel, resulting in feature maps of size 124x124x32. Max-pooling with a 2x2 kernel is then applied to down-sample the feature maps by half. Dropout regularization is employed to prevent overfitting. Subsequent convolutional layers follow a similar pattern, increasing the number of filters and reducing the spatial dimensions of the feature maps. After the final max-pooling layer, the feature maps are flattened into a 1D vector and passed through a dense layer with 128 neurons. For the output layer, a softmax activation function is used, while for other layers, a ReLU activation function is employed. The output layer consists of 27 neurons representing the classes in the ASL classification. This architecture effectively extracts hierarchical features from input images and utilizes fully connected layers for classification.

3.3.2 VGG16

VGG16 is a 16-layer deep convolutional neural network (CNN) architecture proposed by the Visual Geometry Group at the University of Oxford. The network consists of 13 convolutional layers and three fully connected layers. The input size is 224x224x3 (RGB images). The convolutional layers use 3x3 filters with a stride of 1 and padding to maintain the spatial dimensions. The network has a total of approximately 138 million parameters. The VGG16 architecture is known for its simplicity and effectiveness. (Datagen, Understanding VGG16)

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 6. Detailed VGG16 architecture

3.3.3 InceptionV3

The InceptionV3 is a neural network architecture developed by Google's researchers. This complex network consists of 48 layers and is intricately designed to analyze images and extract features in a hierarchical manner. The standout feature of InceptionV3 lies in its unique Inception modules, which redefine feature extraction by utilizing filters of varying sizes, from 1x1 to 5x5, in combination with strategic pooling techniques. Thanks to its comprehensive design and sophisticated feature extraction methods, InceptionV3 achieves exceptional accuracy in tasks such as image classification and various computer vision applications. (Pytorch, INCEPTION_V3)

type	patch size/stride or remarks	input size
conv	3×3/2	299×299×3
conv	3×3/1	149×149×32
conv padded	3×3/1	147×147×32
pool	3×3/2	147×147×64
conv	3×3/1	73×73×64
conv	3×3/2	71×71×80
conv	3×3/1	35×35×192
3×Inception	As in figure 5	35×35×288
5×Inception	As in figure 6	17×17×768
2×Inception	As in figure 7	8×8×1280
pool	8 × 8	8 × 8 × 2048
linear	logits	1 × 1 × 2048
softmax	classifier	1 × 1 × 1000

Figure 7. Detailed InceptionV3 architecture

3.3.4 ResNet50

ResNet50 is a neural network created by Microsoft researchers. It is designed especially for applications such as image recognition. It is a 50-layer convolutional neural network consisting of 48 convolutional layers, one max pooling layer, and one average pooling layer. Residual neural networks are a type of artificial neural network (ANN) that forms networks by stacking residual blocks. (Kaushik, A.)

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 8. Detailed ResNet architecture

3.3.5 DenseNet121

DenseNet121 is a robust neural network architecture that stands out for its dense connectivity pattern and is highly effective in tasks like image recognition. Unlike traditional networks, where each layer is connected only to the next layer, DenseNet121 connects each layer to every other layer in a feed-forward fashion. This

dense connectivity promotes feature reuse, enhances gradient flow, and combats the vanishing gradient problem, leading to more efficient training and improved accuracy. (Pytorch, INCEPTION_V3)

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv, 3 × 3 conv] × 24	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48	[1 × 1 conv, 3 × 3 conv] × 64
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv, 3 × 3 conv] × 16	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Figure 9. Detailed DenseNet architecture

3.4 Evaluation Metrics

After constructing the system model, it undergoes training with the standardized training dataset. Following this, the model's performance is validated by testing it with the testing dataset. The effectiveness of the system's performance is evaluated based on metrics such as Accuracy, Precision, F1 score, and Recall. The confusion matrix is a table that is used to assess a classification model's performance. It provides a matrix-format summary of the model's predictions based on a dataset.

3.4.1 Accuracy

The accuracy of a model is defined as the ratio of true positives and true negatives to all positive and negative observations.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad \text{(Equation 1)}$$

3.4.2 Precision

The precision of a model is defined as the percentage of labels that were correctly predicted positively.

$$\text{Precision} = \frac{TP}{TP + FP} \quad \text{(Equation 2)}$$

3.4.3 Recall

The recall or sensitivity of a model is defined as the ratio of true positives to all the positive instances.

$$\text{Recall} = \frac{TP}{TP + FN} \quad \text{(Equation 3)}$$

3.4.3 F1-score

The f1-score is a metric that calculates the harmonic mean of precision and recall.

$$\text{f1-score} = \frac{2 * precision * recall}{precision + recall} \quad \text{(Equation 4)}$$

3.5 Experimental results

Table 2. Hyper parameters used

Parameter	Value
Epoch	30
Batch size	32
Optimizer	Adam
Learning rate	0.001

The hyper parameters for the model training were determined through experimentation to optimize performance. An epoch value of 30 was selected to train the model over a sufficient number of iterations without excessively prolonging training time or risking overfitting. A batch size of 32 was chosen to balance computational efficiency and model convergence. The Adam optimizer with a learning rate of 0.001 was employed due to its effectiveness in training deep neural networks, providing a balance between rapid convergence and fine-grained parameter updates. These values were found to yield the best performance in terms of accuracy and convergence during experimentation.

3.5.1 Customized CNN

The accuracy of the model for testing data: 99.93%
The loss of the model for testing data: 0.0023

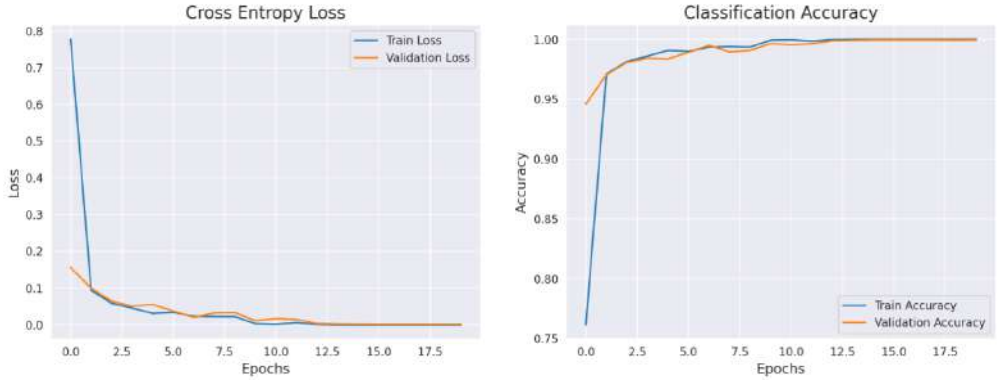

Figure 10. Training and validation set loss and accuracy for customized CNN

3.5.2 VGG16

The accuracy of the model for testing data: 99.34%
The loss of the model for testing data: 0.035

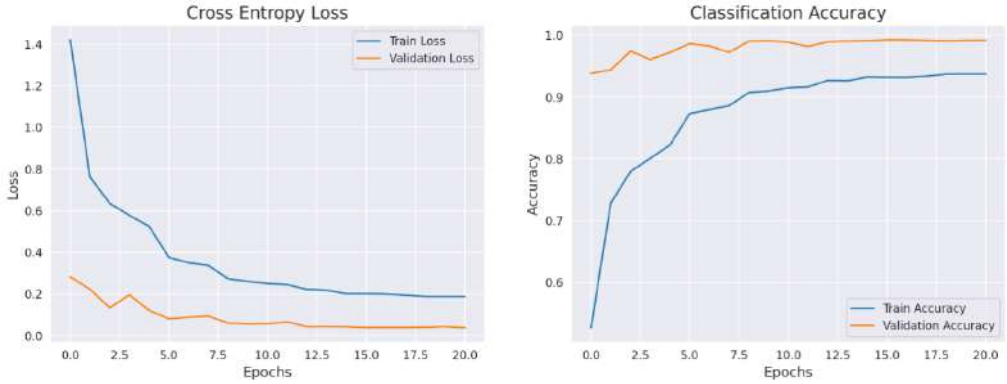

Figure 11. Training and validation set loss and accuracy for VGG16

3.5.3 InceptionV3

The accuracy of the model for testing data: 81.62%
The loss of the model for testing data: 0.56

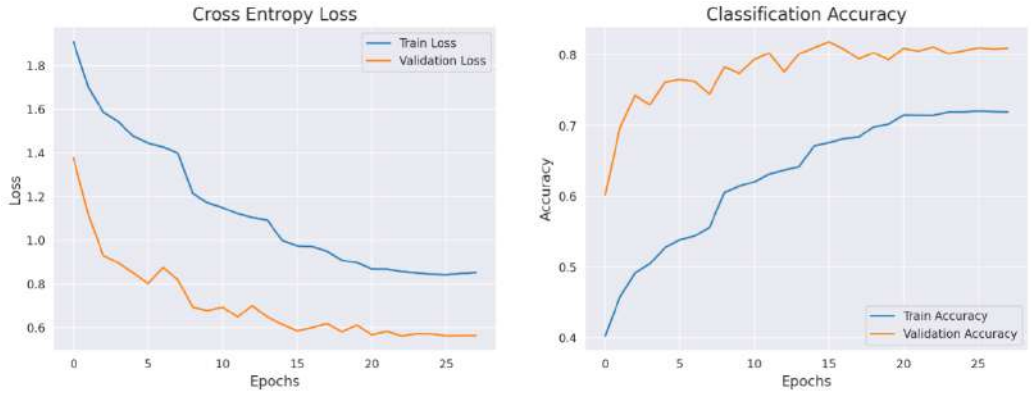

Figure 12. Training and validation set loss and accuracy for InceptionV3

3.5.4 ResNet50

The accuracy of the model for testing data: 99.70%
The loss of the model for testing data: 0.008

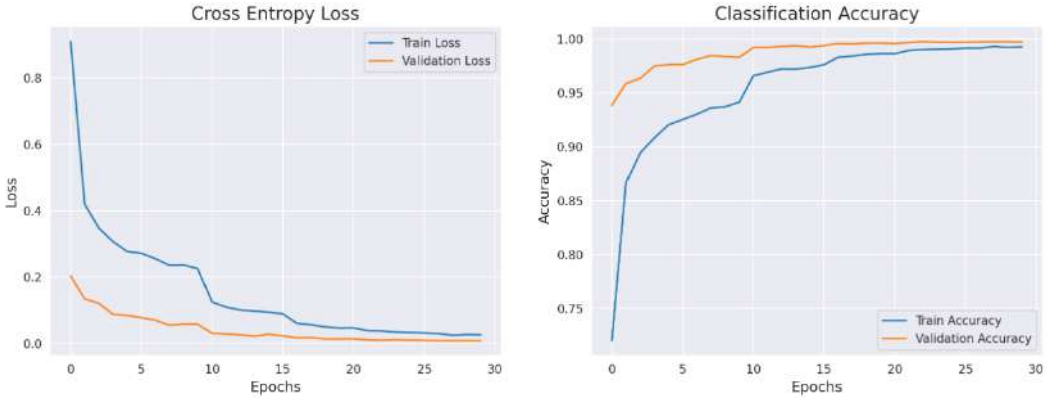

Figure 13. Training and validation set loss and accuracy for ResNet50

3.5.4 DenseNet121

The accuracy of the model for testing data: 99.80%
The loss of the model for testing data: 0.006

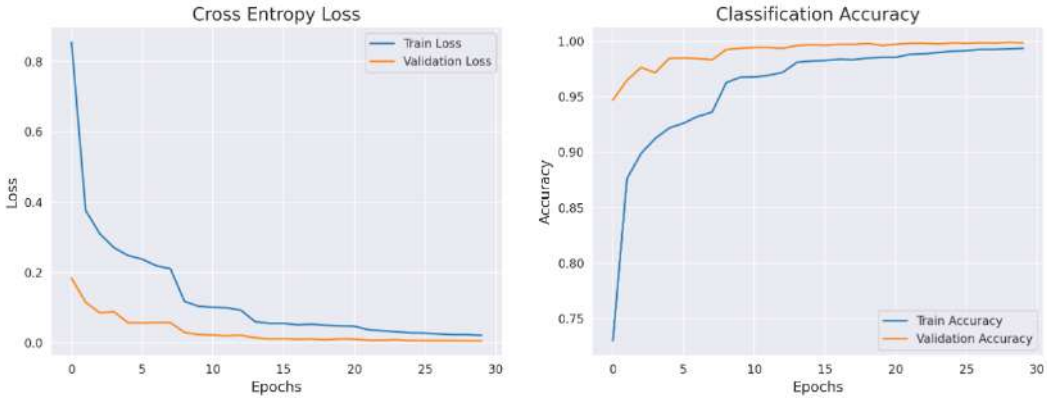

Figure 14. Training and validation set loss and accuracy for DenseNet121

3.5 Model comparison

Table 3. Model performance comparison on test set

Model	Accuracy (%)	Recall	Precision	F1-score
Cust. CNN	99.93%	1.00	1.00	1.00
VGG16	99.34%	0.99	0.99	0.99
InceptionV3	81.62%	0.87	0.82	0.81
ResNet50	99.70%	1.00	1.00	1.00
DenseNet121	99.80%	1.00	1.00	1.00

The table above presents the performance metrics of various convolutional neural network (CNN) models on a classification task. Each model, including Customized CNN, VGG16, InceptionV3, ResNet50V2, and DenseNet121, is evaluated based on accuracy, recall, precision, and F1-score. The customized CNN achieves outstanding results across all metrics, with nearly perfect scores close to 1.00, indicating exceptional performance in correctly classifying instances. VGG16 also demonstrates high accuracy and robust performance across all metrics, while ResNet50V2 and DenseNet121 exhibit similar results. However, InceptionV3 falls short in comparison, with lower accuracy and performance metrics overall. These findings suggest that Customized CNN, VGG16, ResNet50V2, and DenseNet121 are well-suited for the classification task, while InceptionV3 may require further optimization or may be less suitable for this specific dataset or task.

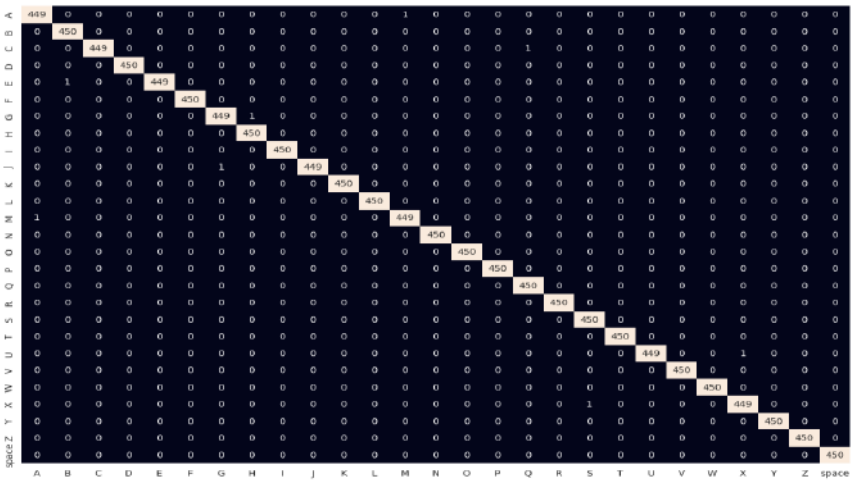

Figure 15. Confusion Matrix for each class in the testing set using customized CNN

4. Real-time implementation

Figure 16. Real-Time ASL Image Classification from Live Camera Feed

The image frames captured by the camera undergo hand detection before being fed into the CNN model for classification. Upon detection, the model predicts the corresponding hand sign, displaying the predicted alphabet. A continuous monitoring mechanism is implemented, wherein if the same sign is predicted for two consecutive seconds, the corresponding alphabet is printed in the text section. This functionality enhances communication through ASL, enabling users to seamlessly construct phrases and sentences.

5. Conclusion and Future Enhancements

In conclusion, the research on American Sign Language (ASL) recognition involved training several models, including a customized CNN, VGG16, InceptionV3, ResNet50, and DenseNet121. Through extensive experimentation, it was discovered that the customized CNN demonstrated superior performance in the testing dataset, highlighting its robustness in effectively recognizing ASL gestures across various environments. This emphasizes the significance of selecting suitable models specifically designed to address the distinct challenges of ASL recognition, thereby laying the foundation for the development of more precise and dependable ASL recognition systems. Furthermore, integrating background subtraction using Media pipe and hand contouring significantly contributed to the research findings. These techniques proved instrumental in mitigating environmental variations, enhancing the robustness of the ASL recognition system across diverse settings. Leveraging Media pipe for background subtraction and hand contour extraction allowed for more accurate localization of hand gestures, thereby improving the overall performance and reliability of the ASL recognition models.

The current model exhibits occasional misclassification of images in real-time scenarios. However, there are promising avenues for improving its accuracy. By fine-tuning hyper parameters and adjusting the number of dense layers following the customized CNN architecture, the model's performance can be enhanced, ensuring more precise recognition of American Sign Language (ASL) gestures. Additionally, optimizing computational efficiency is essential for real-time applications. Addressing the potential slowness of the present model in specific environments could involve refining model architecture, leveraging hardware acceleration, or exploring lightweight alternatives to accommodate faster inference speeds without compromising accuracy. These enhancements can elevate the effectiveness and usability of the ASL recognition system, fostering more significant accessibility and inclusivity for individuals with hearing impairments.

Acknowledgment

The authors would like to extend their heartfelt gratitude to the Department of Computer and Electronics Engineering at Kantipur Engineering College for their unwavering support and resources, which have been invaluable throughout the duration of this research endeavor. The authors are deeply grateful for their contributions, without which this research would not have been possible.

References

AccessComputing,What is sign language?. [Online] Available at:

https://www.washington.edu/accessco mputing/what-sign-language [Accessed 28 March 2024].

AKASH, 2018. ASL Alphabet. [Online] Available at: https://www.kaggle.com/datasets/grassknoted/asl-alphabet [Accessed 28 March 2024].

Bhavana, D., Kumar, K.K., Chandra, M.B., Bhargav, P.S.K., Sanjanaa, D.J. and Gopi, G.M., 2021. Hand sign recognition using CNN. International Journal of Performability Engineering, 17(3), p.314.

Datagen, Understanding VGG16: Concepts, Architecture, and Performance. [Online] Available at: https://datagen.tech/guides/computer-vision/vgg16/ [Accessed 28 March 2024].

Gurucharan, M., 2022. Basic CNN Architecture: Explaining 5 Layers of Convolutional Neural Network. [Online] Available at: https://www.upgrad.com/blog/basic-cnn-architecture/ [Accessed 28 March 2024].

Harris, M. and Agoes, A.S., 2021, November. Applying hand gesture recognition for user guide application using MediaPipe. In 2nd International Seminar of Science and Applied Technology (ISSAT 2021) (pp. 101-108). Atlantis Press.

Hussain, A., Ul Amin, S., Fayaz, M. and Seo, S., 2023. An Efficient and Robust Hand Gesture Recognition System of Sign Language Employing Finetuned Inception-V3 and Efficientnet-B0 Network. Computer Systems Science & Engineering, 46(3).

Kaushik, A., Understanding ResNet50 architecture. [Online] Available at: https://iq.opengenus.org/resnet50-architecture/ [Accessed 28 March 2024].

Marjusalinah, A.D., Samsuryadi, S. and Buchari, M.A., 2021. Classification of finger spelling American sign language using convolutional neural network. Computer Engineering and Applications Journal, 10(2), pp.93-103.

MediaPipe, Hand landmarks detection guide. [Online] Available at: https://developers.google.com/mediapipe/solutions/vision/hand_landmarker [Accessed 28 March 2024].

Mesbahi, S.C., Mahraz, M.A., Riffi, J. and Tairi, H., 2018, April. Hand gesture recognition based on convexity approach and background subtraction. In 2018 International Conference on Intelligent Systems and Computer Vision (ISCV) (pp. 1-5). IEEE.

Mohsin, S., Salim, B.W., Mohamedsaeed, A.K., Ibrahim, B.F. and Zeebaree, S.R., 2024. American Sign Language Recognition Based on Transfer Learning Algorithms. International Journal of Intelligent Systems and Applications in Engineering, 12(5s), pp.390-399.

Noreen, U., Jamil, M. and Ahmad, N., 2015. Hand detection using HSV model. Hand, 6(12).

Premkumar, A., Hridya Krishna, R., Chanalya, N., Meghadev, C., Varma, U.A., Anjali, T. and Siji Rani, S., 2022. Sign language recognition: A comparative analysis of deep learning models. In Inventive Computation and Information Technologies: Proceedings of ICICIT 2021 (pp. 1-13). Singapore: Springer Nature Singapore.

Pytorch, DENSENET. [Online] Available at: https://pytorch.org/hub/pytorch_vision_densenet/ [Accessed 28 March 2024].

Pytorch, INCEPTION_V3. [Online] Available at: https://pytorch.org/hub/pytorch_vision_inception_v3/ [Accessed 28 March 2024].

Rathi, P., Kuwar Gupta, R., Agarwal, S. and Shukla, A., 2020, February. Sign language recognition using resnet50 deep neural network architecture. In 5th International Conference on Next Generation Computing Technologies (NGCT-2019).