Sign Language Recognition using SVM, CNN, RF, and Xception Models

by

Mohammad Abbas Adil

Bachelor of Engineering, Osmania University, India, 2022

A Report Submitted in Partial Fulfillment of the Requirements for the Degree of

MASTER OF ENGINEERING

in the Department of Electrical and Computer Engineering

Sign Language Recognition using SVM, CNN, RF, and Xception Models

by

Mohammad Abbas Adil

Bachelor of Engineering, Osmania University, India, 2022

Supervisory Committee

Dr. T. Aaron Gulliver, Supervisor

(Department of Electrical and Computer Engineering)

Dr. Mihai Sima, Departmental Member

(Department of Electrical and Computer Engineering)

Abstract

Sign language is an essential means of communication for individuals with hearing and speech impairments. It enables them to express thoughts and emotions through hand gestures. With advances in computer vision and machine learning, recognizing these gestures through automated systems has become an active research area. The goal of this study is to develop an efficient system for static hand gesture recognition using supervised machine learning models and compare their performance.

This study uses two distinct datasets of hand gesture images that are openly accessible on Kaggle. The first dataset, called gestures (hand), has 16,000 preprocessed grayscale images in eight different gesture classes: fist, five, okay, peace, rad, straight, thumbs, and none. The second dataset, hand gesture recognition, contains an additional 4,000 preprocessed grayscale images for the same gesture classes. These datasets collectively provide 20,000 images for this study. The first dataset is used for training and validation, and the additional dataset is used for testing. Image augmentation techniques are applied to improve the diversity of training samples and enhance generalization.

Four models are implemented: Support Vector Machine (SVM), Convolutional Neural Network (CNN), Random Forest (RF), and Xception. The SVM model is trained using an RBF kernel with different regularization values (C = 2, 4, 6, 8, 10, 12, 14). The CNN and Xception models are evaluated with early stopping *patience* values ranging from 1 to 7. All models are implemented and tested using Python in the Kaggle notebook environment.

The performance of each model is evaluated using accuracy, precision, recall, F1-score, and training time. The results show that the CNN model achieves the best overall performance with an overall accuracy of 99.20% and a training time of 5.89 min for a *patience* value of 5. The Xception model has 99.08% overall accuracy with the same *patience* value, but with a higher training time of 11.02 min. The SVM classifier achieves a maximum overall accuracy of 90.83% at C = 10 with a training time of 20.35 min. The RF model achieves a maximum overall accuracy of 83.68% for *n_estimators* = 200 with a training time of 0.46 min. These results highlight the effectiveness of deep learning approaches, especially CNN, in real-time gesture recognition.

Contents

List of Tables

List of Figures

Abbreviations

AI	Artificial Intelligence
ASL	American Sign Language
BSL	Bengali Sign Language
CNN	Convolutional Neural Network
DHH	Deaf and Hard of Hearing
DL	Deep Learning
ML	Machine Learning
NLP	Natural Language Processing
OpenCV	Open Source Computer Vision Library
RBF	Radial Basis Function
ReLU	Rectified Linear Unit
RF	Random Forest
SL	Sign Language
SLR	Sign Language Recognition
SVM	Support Vector Machine
TSL	Tanzanian Sign Language
WHO	World Health Organization

Acknowledgement

I sincerely appreciate my supervisor, Dr. T. Aaron Gulliver, for his continued support and wise counsel throughout this project. I am also deeply grateful to him for providing me with the chance to be a part of the University of Victoria (UVic). This work was greatly influenced by his encouragement and guidance. I also want to convey thanks to Dr. Mihai Sima for serving on my supervisory committee.

A special thanks goes to my elder brothers, Mohammed Adnan Adil, Mohammed Ammar Adil, and Mohammad Affan Adil, for helping me apply to UVic and for always being there for me. Their help with important updates and procedures in Canada saved me a lot of time and stress, and allowed me to focus fully on my studies.

Last but not least, I want to thank my sister-in-law, Amreen, for looking after me with such care. Even with her own job and responsibilities to manage, she always made sure I was well and stayed healthy during my time at UVic.

Dedication

To my parents, brothers, and younger sister for their continuous support, affection, and encouragement during my journey.

A special dedication to my mother, whose selfless deeds and restless nights have meant more than words can convey. She has always been there for me when I had difficulties with health growing up, taking care of me, and encouraging me in my studies and in life. Her perseverance and commitment helped shape me into the person I am today.

Chapter 1 Introduction

Sign Language (SL) is an essential parallel to audible languages. It is considered the only language that connects vocal people with the hearing-impaired community. According to the World Health Organization (WHO), this community globally numbers around 430 million with deafness, and 1.5 billion are partially hearing impaired [1]. Recently, awareness of the importance of SL has increased worldwide [2]. The emergence of computers and Natural Language Processing (NLP) has made automated translation systems for audible speech commonplace [4]. However, SL has received less attention since it is structurally different from natural languages and requires more sophisticated techniques [5]. Artificial Intelligence (AI) has become a significant part of our daily lives. Its expansion has exciting implications for the hearing-impaired community [6], and developing SL translation systems has become realistic [7]. Machine translation and image recognition have interested researchers for many decades [8]. Recognizing SL is closely related to machine translation, but with important challenges to solve [8]. There has been significant research on SL translation to create a communication environment for the Deaf and Hard of Hearing (DHH).

The goal of this project is to compare the performance of four supervised Machine Learning (ML) models: Support Vector Machine (SVM), Convolutional Neural Network (CNN), Random Forest (RF), and Xception for Sign Language Recognition (SLR). These models are evaluated on a vision-based static gesture recognition task using two publicly available image datasets sourced from Kaggle. Each model is trained and tested with different parameters to assess its effectiveness in terms of recognition performance and training time.

1.1 Motivation

According to the United Nations, the number of deaf people in 2050 will be 900 million [9], so SLR is an important problem. It can help close the communication gap between people who use SL and those who do not. This reduces frustration and makes conversations easier and more effective. Developing an SLR system is challenging and involves various components such as hand movement, shape, position, orientation, palm posture, finger movement, facial expression, and body posture. In addition to these components, lip shapes and eyebrow positions are often used to distinguish visually similar signs.

1.2 Related Work

Early SLR studies often used small or custom-built datasets, which limits their relevance and comparability. Many researchers have developed systems to recognize hand gestures for SL using ML and Deep Learning (DL) models. In [10], a CNN was used to recognize static hand gestures in American Sign Language (ASL). The model was trained on over 1,800 images and achieved a validation accuracy of 94.34%, showing that CNNs can perform well on clean, labeled image datasets. It was found in [11] that CNN-based models are widely used and often give the best results for static gesture recognition. However, the performance of these models can be affected by challenges such as lighting conditions, background noise, and image quality [13]. In [12], a CNN model was developed for the Bengali Sign Language (BSL) using a custom dataset of 1850 images (37 images for each of the 50 letters in the alphabet) collected from just one signer. While the model performed well, the small dataset size reduces confidence in its effectiveness for other users. An Xception-based model was proposed in [13] for BSL recognition, but only a small dataset obtained in a controlled setting was used which limits the real-world applicability. In contrast, there has been limited use of models like SVM. In [14], SVM and CNN were compared on a custom Tanzanian Sign Language (TSL) image dataset consisting of 3000 images (100 images for each of 30 words), and it was found that CNN outperformed SVM. As the dataset was small, it remains unclear whether the results hold at scale. Very few studies include training time in their evaluation, even though it is important for real-time applications and devices with limited resources [14].

Previous research indicates models are often tested on small, custom datasets, making broad conclusions difficult. This project addresses the limitations of dataset size by evaluating the SVM, CNN, RF, and Xception models on two publicly available Kaggle datasets totaling 20,000 images comparing both evaluation metrics (Section 4.1) and training time. This larger-scale study, using consistent datasets and evaluation methods, provides a better understanding of how the models perform under the same conditions.

1.3 Report Outline

The structure of this report is as follows.

Chapter 1 introduced SLR and gave the motivation for this project. It also provided an overview of the related work in this field.

Chapter 2 presents the ML techniques used in this study. It includes a discussion on supervised learning, classification, the CNN, Xception, RF, and SVM models, and the tools used such as Kaggle notebook and Python libraries.

Chapter 3 gives the system design for SLR. It includes details about the dataset, preprocessing and augmentation methods, and the design and configuration of the four models used.

Chapter 4 presents the evaluation metrics used, the performance of each model for different parameters, and a comparison and analysis of the results.

Chapter 5 summarizes the results, identifies limitations, and provides suggestions for future research on SLR systems.

Chapter 2 Machine Learning

ML is a branch of AI that enables computers to learn from data without the need for explicit programming. It uses algorithms to analyze data, identify patterns, and perform prediction or classification tasks. In contrast to traditional programming, where rules are manually defined, ML systems adapt and improve their performance automatically as additional data becomes available, which can result in higher accuracy. DL is a specialized subset of ML inspired by the structure and function of the human brain [15]. It utilizes artificial neural networks with multiple layers to process and represent information at various levels of abstraction. These networks can extract complex and hierarchical features from large datasets such as images, text, or audio, making them particularly effective for visual recognition tasks such as hand gesture classification [16].

2.1 Supervised Learning

With supervised learning, ML models learn from labeled data, so every input has a known correct result. For the model to produce predictions on unseen data, a pattern or mapping between inputs and outputs is learned. This is similar to a student learning under a teacher's supervision [17]. Supervised learning is used for problems like regression (e.g., predicting property values) and classification (e.g., determining if an email is spam or not). Classification here means categorizing data and forming groups based on similarities or features. The amount and quality of the training data have a significant impact on model performance [20].

2.2 Support Vector Machine (SVM)

SVM is a popular ML algorithm [21]. It is a supervised ML algorithm that can be used for both classification and regression. In terms of classification, it finds the optimal hyperplane that separates data points belonging to different classes. In two-dimensional space, this hyperplane can be thought of as a line dividing the space into two parts: each containing data points that belong primarily to one class [21]. However, this line can only separate the data correctly if the data is linearly separable. For datasets that are not linearly separable, SVM can use kernel functions to map the data into a higher-dimensional space where an optimal separating boundary can be established [22].

2.3 Convolutional Neural Network (CNN)

A CNN is a neural network used to process data that has a grid-like topology, such as an image. A digital image is a binary representation of visual data. It contains a series of pixels arranged in a grid-like fashion with values to denote how bright and what color they should be [23]. The convolutional layer is the core component of a CNN, responsible for most of the computation [23]. It operates using three elements: input data, a filter (or kernel), and a feature map. The input image is a three-dimensional matrix with height, width, and depth corresponding to the RGB color channels. The filter, typically a small matrix, slides over the image, performing a dot product between its values and the input pixels in the receptive field [24]. This operation, known as convolution, generates a feature map that highlights the presence of specific patterns or features in the image. As the network deepens, successive convolutional layers learn increasingly complex representations by combining low-level features like edges and textures into higher-level patterns [24]. The stride determines how far the filter moves across the image, and zero-padding is used to preserve spatial dimensions when the filter does not perfectly fit the input. Pooling layers further reduce the size of feature maps, lowering computational cost and mitigating overfitting [24]. Activation functions introduce non-linearity, enabling the network to model complex relationships. Fully connected layers integrate the extracted features and generate the final output for classification tasks.

2.4 Xception

Xception (short for Extreme Inception) is a DL model introduced as an extension of the inception architecture [25]. This architecture utilizes inception modules that use multiple convolutional and pooling operations in parallel to capture information at different spatial scales. The Xception model refines this concept by replacing the inception modules with depthwise separable convolutions, enabling the network to learn spatial and cross-channel features independently [25]. In a standard convolution, each filter simultaneously learns relationships across the spatial dimensions and depth (channels). In contrast, depthwise separable convolution divides this process into two operations: depthwise convolution and pointwise convolution. Depthwise convolution applies a single filter to each input channel separately, and pointwise convolution combines the resulting outputs to form a unified feature map. This separation improves computational efficiency while maintaining representation capability. As illustrated in Figure 1, the Xception architecture

can be divided into three main parts: input processing layer, depthwise feature mapping layer, and prediction layer. The input processing layer performs the initial convolution operations to extract low-level spatial features from the input image [12]. It also reduces the image dimensions gradually through pooling. The depthwise feature mapping layer consists of a sequence of depthwise separable convolution blocks, each followed by ReLU activation [12]. This block is repeated eight times to progressively learn higher-level and more abstract feature representations. In the prediction layer, additional separable convolution and pooling operations further refine the extracted features. The output feature maps are then aggregated using a global average pooling layer and passed to fully connected layers for final prediction [25]. Projection blocks (skip connections) are used to maintain dimensional consistency, facilitate efficient gradient flow, and improve learning stability.

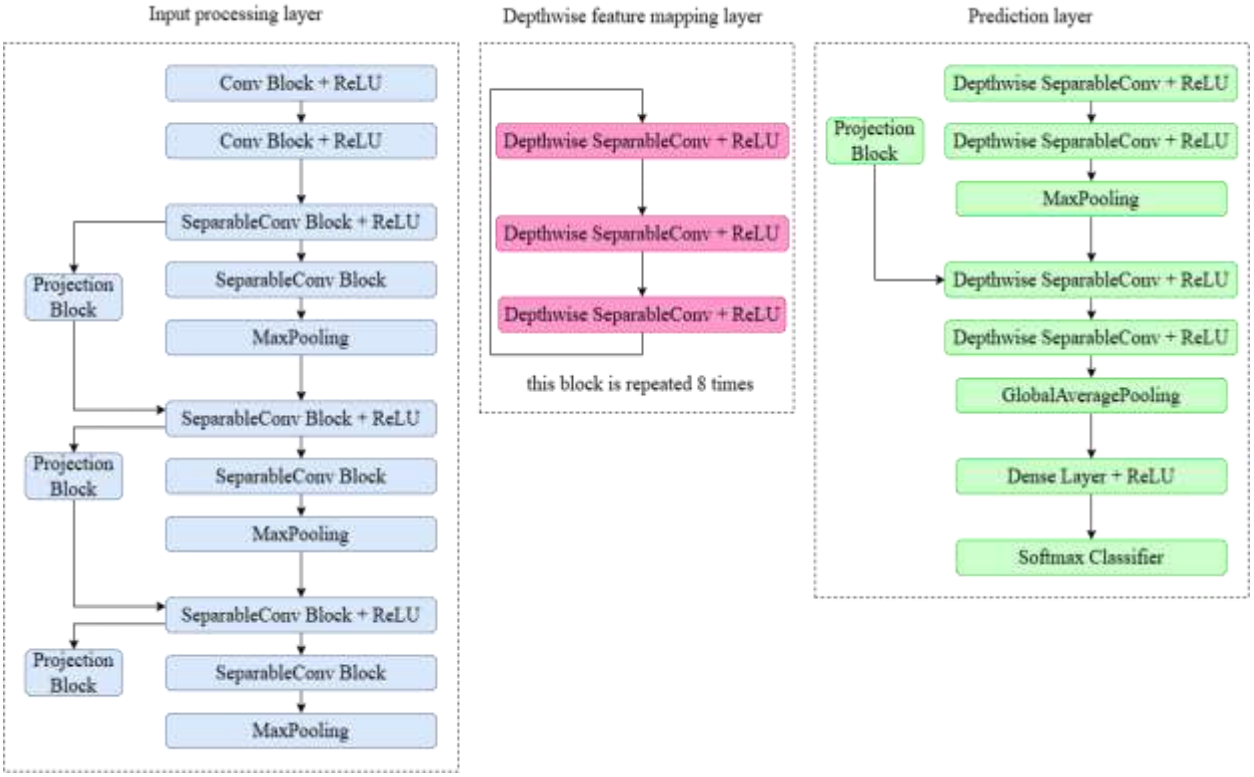


Figure 1 The Xception architecture.

2.5 Random Forest (RF)

RF is an ensemble ML model that combines multiple decision trees [26]. Each tree in the forest is trained on a random subset of the data using bootstrap sampling and considers only a random

subset of features when making splits, a process known as feature randomization [18]. In a decision tree, a split divides the data at a node based on a feature and threshold that separates the classes to minimize prediction error. Each tree has a training set created by sampling the original data with replacement. During node selection, each tree only considers a random subset of features (typically the square root of the features), to determine the best split. Each tree grows using only its bootstrap sample and selected features, making splits until it reaches a stopping point (such as pure groups or minimum sample size). All trees vote to obtain the final prediction. For classification, a majority vote of class predictions is made, and for regression, the average of the predicted values is used [26].

2.6 Kaggle Notebook and Python Tools

This project was built using the Kaggle notebook environment. This is a cloud-based platform that makes it easy to develop and run ML models. Kaggle provides powerful GPUs to pre-installed libraries, all accessible through a browser. The coding is done in Python 3.11.13. For DL, TensorFlow 2.18.0 and Keras 3.8.0 were used to build and train the ML models. These models were trained on NVIDIA Tesla T4 GPUs, each with 15 GB of VRAM, which significantly speeds up the process. The GPUs are supported by CUDA 12.6 and an Intel Xeon CPU.

The following Python libraries were used in this study. Scikit-learn is a Python library for ML that provides tools for data preprocessing, model training, and evaluation [19]. TensorFlow is an open-source framework for DL and numerical computation that supports GPU acceleration [19]. Keras is a high-level neural network API built on TensorFlow that simplifies the design and training of DL models [24]. NumPy is a library for scientific computing that provides support for multi-dimensional arrays and mathematical operations [26]. Matplotlib is a library for data visualization used to create static and interactive plots [19]. Seaborn is a statistical data visualization library built on Matplotlib for creating clear and structured graphics [19]. Pandas is a library for data manipulation and analysis that provides Series and DataFrame structures for managing tabular data [25]. Open Source Computer Vision Library (OpenCV) is a computer vision library that provides tools for image processing and feature extraction [12].

Chapter 3 Sign Language Recognition System Design

Figure 3.1 illustrates the architecture of the proposed SLR system. In the subsequent sections, the individual components are explained in detail.

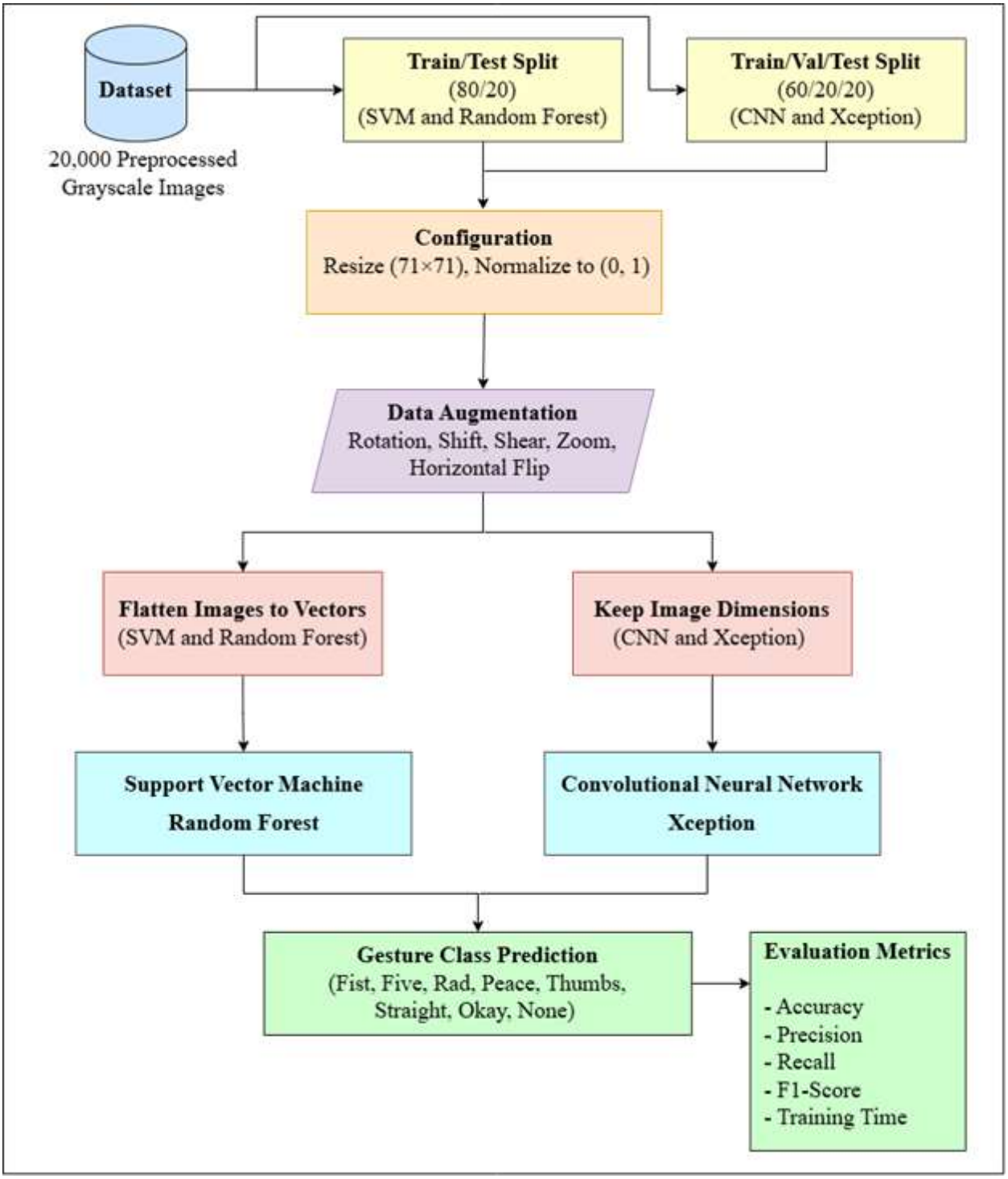


Figure 2 The sign language recognition system.

3.1 The Sign Language Dataset

This study uses two distinct datasets of hand gesture images that are openly accessible on Kaggle. The first dataset, called gestures (hand), has 16,000 preprocessed grayscale images in eight different gesture classes as shown in Table 1: fist, five, okay, peace, rad, straight, thumbs, and none (plain background/no gesture). There are 4,000 additional preprocessed grayscale images for the same gesture classes in the second dataset, hand gesture recognition. These datasets collectively provide 20,000 images for this study. The first dataset is used for training and validation, and the additional dataset is used for testing.

Gesture Image	Gesture Name	Description
	Fist	A closed hand with all fingers curled in, like making a punch or holding something tightly.
	Five	Open palm, all five fingers spread out, like saying hi or showing the number five.
	Okay	Index finger touching the thumb to make an O shape, a gesture for okay or all good.
	Peace	Index and middle fingers raised to form a V, a symbol of peace or victory.
	Rad	Thumb and pinky extended, other fingers folded, often used for cool or rock on.

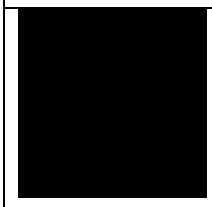
	Straight	Hand extended with fingers held together and straight, like a blade or karate chop.
	Thumbs	Thumb raised, other fingers curled, commonly used to show approval or good job.
	None	No hand gesture shown, these are plain background images used to detect silence or idle state.

Table 1 Gesture names and descriptions.

3.2 Dataset Preprocessing and Augmentation

An input shape of 71×71 is the minimum required for the Xception model, so the images were resized to 71×71 pixels to ensure a uniform input shape for the four models [29]. The images were normalized to (0, 1) to improve gradient stability and model convergence during training [28]. For the SVM and RF models, each image was flattened into a one-dimensional vector containing 5,041 features. The data was split into 80% training (16,000 images) and 20% testing (4,000 images). Data augmentation was applied only to the training set, with each training image augmented three times, generating an additional 48,000 images and resulting in a total of 64,000 training images. 5-fold cross-validation was applied on the training data with 51,200 images used for training and 12,800 images for validation in each fold. Thus, every image served once as validation and four times as training. Cross-validation was used for hyperparameter tuning and validation within the training data, serving an equivalent role to the validation set used for the CNN and Xception models. For the CNN and Xception models, the images were directly processed without flattening. The data was split into 60% training (12,000 images), 20% validation (4,000 images), and 20% testing (4,000 images) using the Keras image data generator. Data augmentation was applied only to the training set, with each training image augmented three times, generating an additional 36,000 images and resulting in a total of 48,000 training images. The augmentation transformations

included rotation, width and height shifts, shear, zoom, and horizontal flipping to increase data diversity and improve model generalization. For each augmented copy of an image, two transformations from this set were applied.

3.3 Support Vector Machine (SVM)

The values of the regularization parameter considered using the scikit-learn library to train the SVM model were C = 2, 4, 6, 8, 10, 12, and 14. A lower value enables the model to better adapt and generalize, while a larger value makes the model fit the training data more closely. A Radial Basis Function (RBF) kernel was used which is common for non-linear classification tasks. The RBF kernel maps the data into a higher-dimensional space where a better decision boundary can be found. The criterion that determines when SVM finishes training is *tol* (tolerance), which is the minimum improvement required to continue training. The model stops training once the improvement is smaller than this value. The default value *tol* = 0.001 was used to train the model.

3.4 Random Forest (RF)

The RF model in the scikit-learn library was used. The number of trees (*n_estimators*) considered was 200, 300, 400, and 500. In general, the model produces more stable predictions with more trees by reducing the variance. To prevent overfitting, the maximum depth (*max_depth*) of the trees was set to 30. The parameter *min_samples_split* was fixed at 2, so a node can split only when at least two classes are present, ensuring the model captures decision boundaries without hindering tree growth.

3.5 Convolutional Neural Network (CNN)

The sequential API in TensorFlow was used to develop the CNN model. There are three convolutional layers. The first has 32 filters of size 3×3 with ReLU activation, while the second and third layers have 64 and 128 filters, respectively. Following each convolutional layer, the output was stabilized using batch normalization, and the spatial dimensions were reduced using a max pooling layer. The dropout was 0.2, 0.3, and 0.4 after each layer, respectively, to randomly turn off neurons during training to avoid overfitting. The feature maps were reduced to a vector using a global average pooling layer following the last convolutional layer. Next is a dense layer with 256 neurons and ReLU activation, and an additional dropout layer with a rate of 0.5. Eight

neurons with softmax activation were used in the final output layer to produce probability scores, one for each gesture class. The Adam optimizer with a learning rate of 0.0008 was used to construct the model. Categorical cross-entropy, a loss function appropriate for multi-class classification problems, was employed. The model was trained for a maximum of 40 epochs with a batch size of 32. To avoid overfitting, early stopping was used to monitor the validation loss with *patience* = 1, 2, 3, 4, 5, 6, and 7.

3.6 Xception

The functional API in TensorFlow was used to develop the Xception-based model. Since Xception expects a 3-channel input, a Conv2D layer was used to project the single channel grayscale images into a 3-channel representation (resulting in an input shape of 71×71×3). The convolutional layers of the Xception model were loaded without the pre-trained ImageNet weights, as the SL dataset used in this study differs from the natural-image content of ImageNet. The original fully connected layers were removed and replaced with new layers specifically designed for classifying the eight gesture classes, as described below. The output feature maps from the convolutional layers were reduced to a vector using a global average pooling layer. A dropout layer with rate 0.5 was added to mitigate overfitting. This was followed by a dense layer with 256 neurons using the ReLU activation function. Another dropout layer with rate 0.5 was added before the final output layer. The final classification layer had 8 neurons with softmax activation to provide probability scores for each of the gesture classes. The Adam optimizer with a learning rate of 0.0008 was used to construct the model. Categorical cross-entropy was employed as the loss function. The model was trained for a maximum of 40 epochs with a batch size of 32. To avoid overfitting, early stopping was used to monitor the validation loss with *patience* = 1, 2, 3, 4, 5, 6, and 7.

Chapter 4 Performance Evaluation of Models

In this chapter, the performance of the four supervised ML models is evaluated for SLR. The results were obtained using Kaggle notebooks. The software and environment details used for model training and testing were described in Section 2.5.

4.1 Evaluation Metrics

The performance metrics used are as follows.

Accuracy is the percentage of the predictions that are correct for a given gesture [30]

$$\text{Accuracy} = \frac{tp+tn}{tp+tn+fp+fn}$$

where true positive (tp) is the number of gesture images correctly predicted for the gesture, true negative (tn) is the number of images correctly predicted as not belonging to the gesture, false positive (fp) is the number of images wrongly predicted as the gesture, and false negative (fn) is the number of images that were incorrectly predicted for the gesture.

Overall accuracy is the percentage of the predictions that are correct for the gestures [30]

$$\text{Overall accuracy} = \frac{\text{number of correct predictions for the gestures}}{\text{total number of gesture images}}$$

Precision is the ratio of correctly predicted images of a gesture to the total images predicted as that gesture [19]

$$\text{Precision} = \frac{tp}{tp+fp}$$

Recall is the ratio of correctly predicted images of a gesture to all images for that gesture [19]

$$\text{Recall} = \frac{tp}{tp+fn}$$

F1-Score is the harmonic mean of precision and recall [19]

$$\text{F1-Score} = \frac{2\times\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}}$$

Training Time is the time taken to train a model. A shorter training time is preferred, especially in real-time or resource-constrained environments [19].

4.2 SVM Results with Different *C* Values

The SVM model was trained by splitting the data into 80% for training and 20% for testing. It was trained using C = 2, 4, 6, 8, 10, 12, and 14. This section gives the test results and confusion matrices for each C value. The training times are also included. Figure 3 presents the results for C = 2. The numbers of correctly predicted images for the eight gesture classes fist, five, none, okay, peace, rad, straight, and thumbs out of 500 were 418, 345, 500, 451, 436, 417, 497, and 464, respectively. Table 2 gives the average precision, recall, F1-score, and accuracy which are 88.80%, 88.20%, 88.08%, and 97.05%, respectively, with a training time of 20.29 min. Figure 4 presents the results for C = 4. The numbers of correctly predicted images for the eight classes were 434, 368, 500, 447, 449, 436, 500, and 469. Table 3 gives the average test results which are 90.53%, 90.08%, 90.02%, 97.52%, respectively, with a training time of 21.60 min. Figure 5 presents the results for C = 6. The numbers of correctly predicted images for the eight gesture classes were 441, 373, 500, 448, 452, 439, 500, and 465, respectively. Table 4 gives the average test results which are 90.84%, 90.45%, 90.40%, and 97.61%, respectively, with a training time of 21.29 min. Figure 6 presents the results for C = 8. The numbers of correctly predicted images for the eight gesture classes were 444, 375, 500, 448, 455, 439, 500, and 464, respectively. Table 5 gives the average test results which are 90.98%, 90.62%, 90.57%, and 97.66%, respectively, with a training time of 21.89 min. Figure 7 presents the results for C = 10. The numbers of correctly predicted images for the eight gesture classes were 450, 366, 500, 435, 469, 439, 497, and 477, respectively. Table 6 gives the average test results which are 91.10%, 90.82%, 90.75%, and 97.71%, respectively, with a training time of 20.35 min. Figure 8 presents the results for C = 12. The numbers of correctly predicted images for the eight gesture classes were 450, 365, 500, 435, 469, 439, 497, and 476, respectively. Table 7 gives the average test results which are 91.04%, 90.78%, 90.69%, and 97.69%, respectively, with a training time of 20.81 min. Figure 9 presents the results for C = 14. The numbers of correctly predicted images for the eight gesture classes were 451, 365, 500, 435, 469, 439, 497, and 476, respectively. Table 8 gives the average test results which are 91.06%, 90.80%, 90.72%, and 97.70%, respectively, with a training time of 20.84 min.

The highest overall accuracy of 90.83% was obtained with C = 10 and a training time of 20.35 min. The lowest overall accuracy of 88.20% was obtained with C = 2 and a training time of 20.29 min. The model performance gradually improved with increasing C values up to 10 and then remained approximately constant, indicating that C = 10 is the best value.

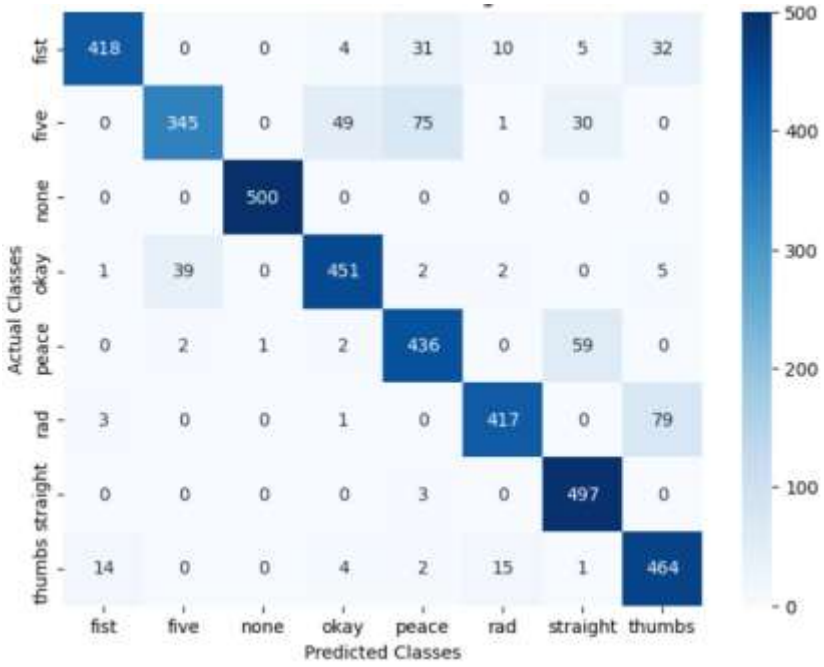


Figure 3 SVM confusion matrix for C = 2.

Gesture Class	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)	Overall Accuracy (%)	Training Time (min)
Fist	95.87	83.60	89.32	97.50	88.20	20.29
Five	89.38	69.00	77.88	95.10		
None	99.80	100	99.90	99.98		
Okay	88.26	90.20	89.22	97.28		
Peace	79.42	87.20	83.13	95.58		
Rad	93.71	83.40	88.25	97.22		
Straight	83.95	99.40	91.03	97.55		
Thumbs	80.00	92.80	85.93	96.20		
Average	88.80	88.20	88.08	97.05		

Table 2 SVM test results for C = 2.

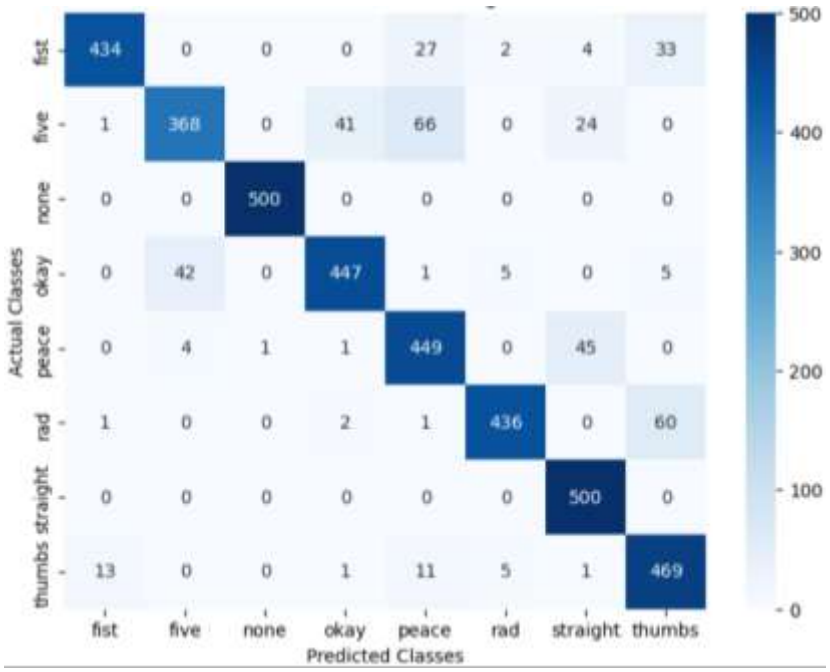


Figure 4 SVM confusion matrix for $C = 4$.

Gesture Class	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)	Overall Accuracy (%)	Training Time (min)
Fist	96.66	86.80	91.46	97.98	90.08	21.60
Five	88.89	73.60	80.53	95.55		
None	99.80	100	99.90	99.98		
Okay	90.85	89.40	90.12	97.55		
Peace	80.90	89.80	85.12	96.08		
Rad	97.32	87.20	91.98	98.10		
Straight	87.11	100	93.11	98.15		
Thumbs	82.72	93.80	87.91	96.78		
Average	90.53	90.08	90.02	97.52		

Table 3 SVM test results for $C = 4$.

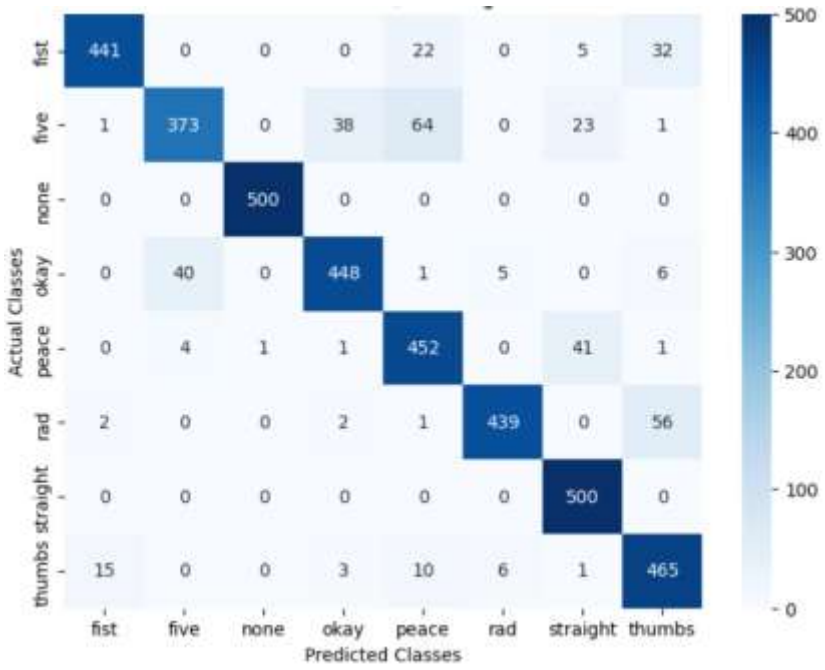


Figure 5 SVM confusion matrix for $C = 6$.

Gesture Class	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)	Overall Accuracy (%)	Training Time (min)
Fist	96.08	88.20	91.97	98.08	90.45	21.29
Five	89.45	74.60	81.35	95.72		
None	99.80	100	99.90	99.98		
Okay	91.06	89.60	90.32	97.60		
Peace	82.18	90.40	86.10	96.35		
Rad	97.56	87.80	92.42	98.20		
Straight	87.72	100	93.46	98.25		
Thumbs	82.89	93.00	87.65	96.72		
Average	90.84	90.45	90.40	97.61		

Table 4 SVM test results for $C = 6$.

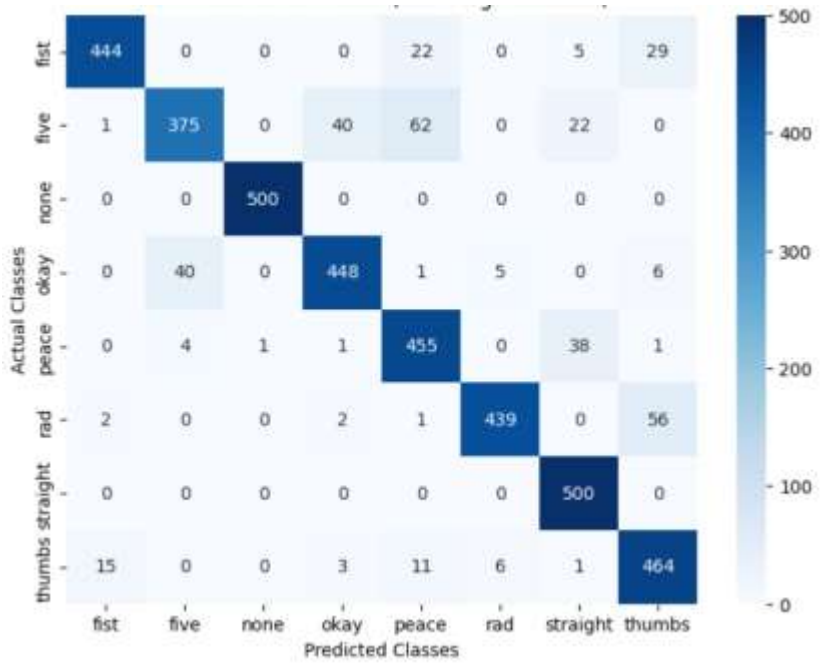


Figure 6 SVM confusion matrix for $C = 8$.

Gesture Class	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)	Overall Accuracy (%)	Training Time (min)
Fist	96.10	88.80	92.31	98.15	90.62	21.89
Five	89.50	75.00	81.61	95.78		
None	99.80	100	99.90	99.98		
Okay	90.69	89.60	90.14	97.55		
Peace	82.43	91.00	86.50	96.45		
Rad	97.56	87.80	92.42	98.20		
Straight	88.34	100	93.81	98.35		
Thumbs	83.45	92.80	87.88	96.80		
Average	90.98	90.62	90.57	97.66		

Table 5 SVM test results for $C = 8$.

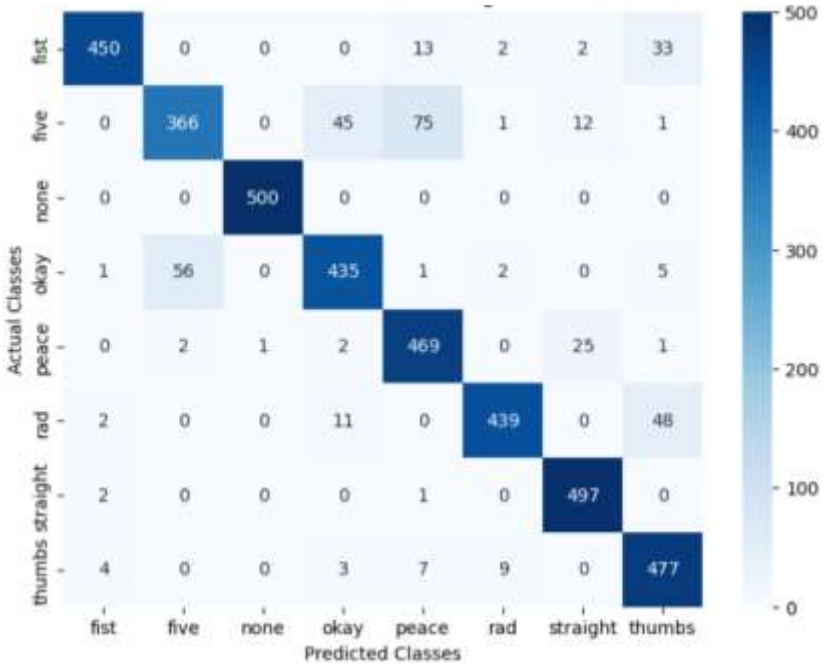


Figure 7 SVM confusion matrix for C = 10.

Gesture Class	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)	Overall Accuracy (%)	Training Time (min)
Fist	98.04	90.00	93.85	98.52	90.83	20.35
Five	86.32	73.20	79.22	95.20		
None	99.80	100	99.90	99.98		
Okay	87.70	87.00	87.35	96.85		
Peace	82.86	93.80	87.99	96.80		
Rad	96.91	87.80	92.13	98.12		
Straight	92.72	99.40	95.95	98.95		
Thumbs	84.42	95.40	89.58	97.22		
Average	91.10	90.82	90.75	97.71		

Table 6 SVM test results for C = 10.

Figure 8 SVM confusion matrix for C = 12.

Gesture Class	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)	Overall Accuracy (%)	Training Time (min)
Fist	98.04	90.00	93.85	98.52	90.78	20.81
Five	86.29	73.00	79.09	95.18		
None	99.80	100	99.90	99.98		
Okay	87.53	87.00	87.26	96.82		
Peace	82.86	93.80	87.99	96.80		
Rad	96.70	87.80	92.03	98.10		
Straight	92.72	99.40	95.95	98.95		
Thumbs	84.40	95.20	89.47	97.20		
Average	91.04	90.78	90.69	97.69		

Table 7 SVM test results for C = 12.

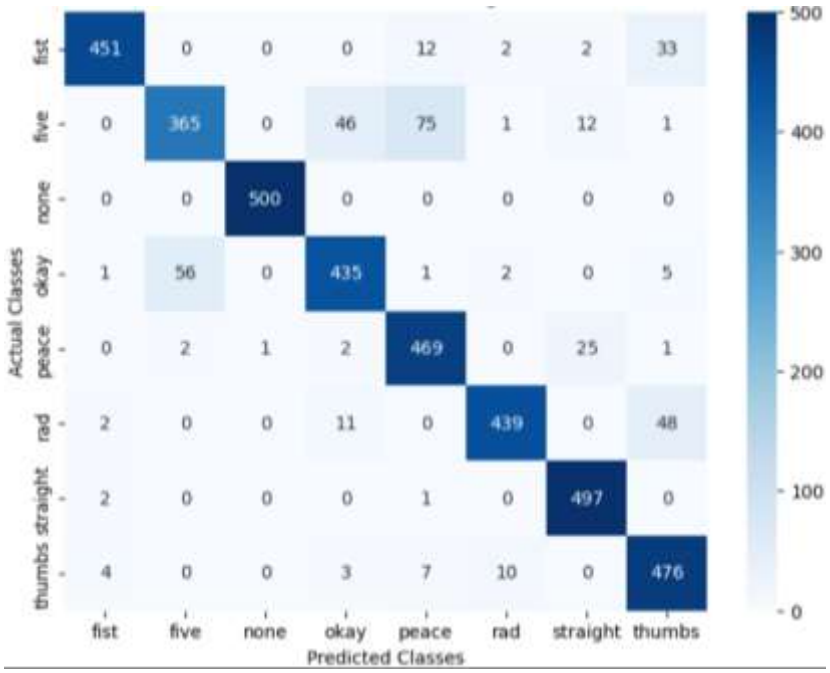


Figure 9 SVM confusion matrix for C = 14.

Gesture Class	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)	Overall Accuracy (%)	Training Time (min)
Fist	98.04	90.20	93.96	98.55	90.80	20.84
Five	86.29	73.00	79.09	95.18		
None	99.80	100	99.90	99.98		
Okay	87.53	87.00	87.26	96.82		
Peace	83.01	93.80	88.08	96.82		
Rad	96.70	87.80	92.03	98.10		
Straight	92.72	99.40	95.95	98.95		
Thumbs	84.40	95.20	89.47	97.20		
Average	91.06	90.80	90.72	97.70		

Table 8 SVM test results for C = 14.

4.3 Random Forest Results with Different Numbers of Trees

The RF model was trained by splitting the data into 80% for training and 20% for testing. It was trained using *n_estimators* = 200, 300, 400, and 500. This section gives the test results and confusion matrices for each value of *n_estimators*. The training times are also included. Figure 10 presents the results for *n_estimators* = 200. The numbers of correctly predicted images for the eight gesture classes fist, five, none, okay, peace, rad, straight, and thumbs out of 500 were 362, 346, 500, 443, 421, 369, 498, and 408, respectively. Table 9 gives the average precision, recall, F1-score, and accuracy which are 85.33%, 83.67%, 83.69%, and 95.92%, respectively, with a training time of 0.46 min. Figure 11 presents the results for *n_estimators* = 300. The numbers of correctly predicted images for the eight gesture classes were 349, 346, 500, 436, 425, 376, 498, and 408, respectively. Table 10 gives the average test results which are 84.96%, 83.45%, 83.41%, and 95.86%, respectively, with a training time of 0.72 min. Figure 12 presents the results for *n_estimators* = 400. The numbers of correctly predicted images for the eight gesture classes were 347, 348, 500, 437, 420, 373, 499, and 417, respectively. Table 11 gives the average test results which are 85.14%, 83.52%, 83.50%, and 95.88%, respectively, with a training time of 0.91 min. Figure 13 presents the results for *n_estimators* = 500. The numbers of correctly predicted images for the eight gesture classes were 345, 345, 500, 434, 421, 370, 498, and 414, respectively. Table 12 gives the average test results which are 84.80%, 83.17%, 83.14%, and 95.80%, respectively, with a training time of 1.16 min.

The highest overall accuracy of 83.68% was obtained with *n_estimators* = 200 and a training time of 0.46 min. The lowest overall accuracy of 83.18% was obtained with *n_estimators* = 500 and a training time of 1.16 min. The model performance is approximately constant across different *n_estimators* values, with slight variations in overall accuracy and training time.

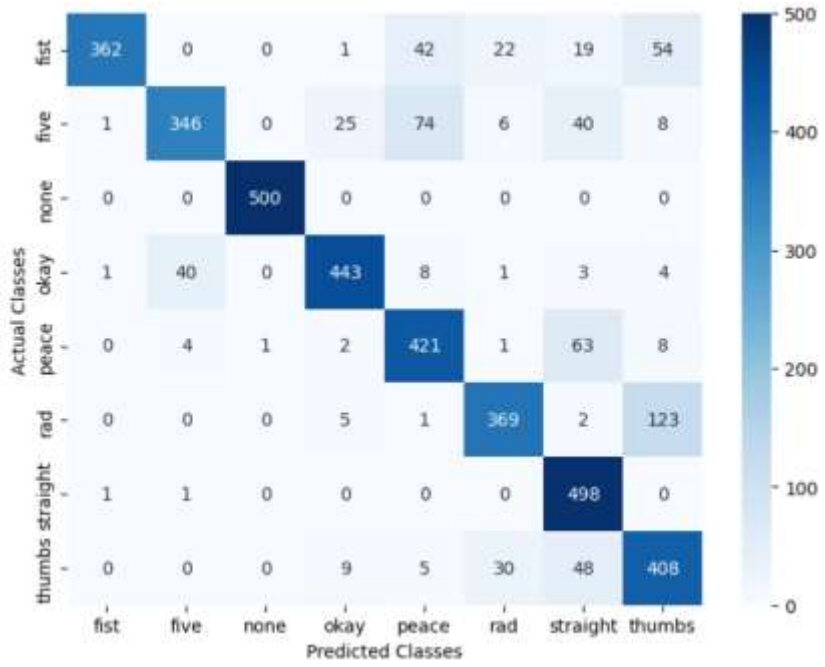


Figure 10 RF confusion matrix for *n_estimators* = 200.

Gesture Class	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)	Overall Accuracy (%)	Training Time (min)
Fist	99.18	72.40	83.70	96.48	83.68	0.46
Five	88.49	69.20	77.67	95.02		
None	99.80	100	99.90	99.98		
Okay	91.34	88.60	89.95	97.52		
Peace	76.41	84.20	80.11	94.78		
Rad	86.01	73.80	79.44	95.22		
Straight	74.00	99.60	84.91	95.58		
Thumbs	67.44	81.60	73.85	92.78		
Average	85.33	83.67	83.69	95.92		

Table 9 RF test results for *n_estimators* = 200.

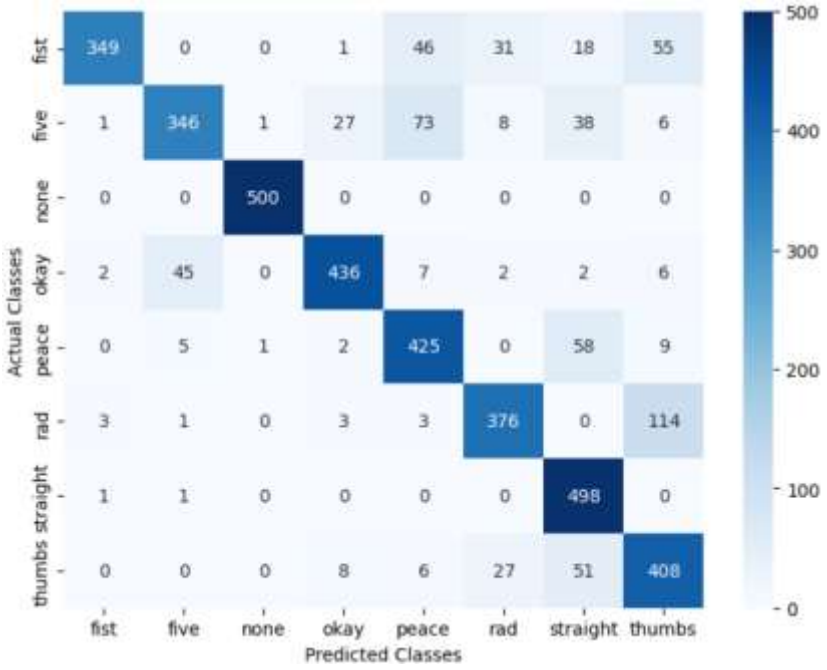


Figure 11 RF confusion matrix for *n_estimators* = 300.

Gesture Class	**Precision (%)**	**Recall (%)**	**F1-Score (%)**	**Accuracy (%)**	**Overall Accuracy (%)**	**Training Time (min)**
Fist	98.03	69.80	81.54	96.05	83.45	0.72
Five	86.93	69.20	77.06	94.85		
None	99.60	100	99.80	99.95		
Okay	91.40	87.20	89.25	97.38		
Peace	75.89	85.00	80.19	94.75		
Rad	84.68	75.20	79.66	95.20		
Straight	74.89	99.60	85.49	95.78		
Thumbs	68.23	81.60	74.32	92.95		
Average	84.96	83.45	83.41	95.86		

Table 10 RF test results for *n_estimators* = 300.

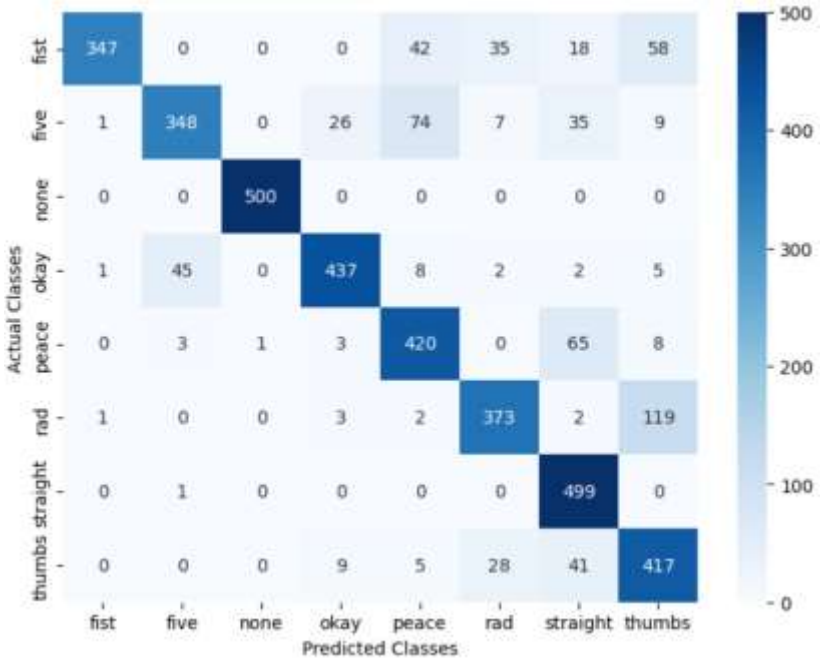


Figure 12 RF confusion matrix for *n_estimators* = 400.

Gesture Class	**Precision (%)**	**Recall (%)**	**F1-Score (%)**	**Accuracy (%)**	**Overall Accuracy (%)**	**Training Time (min)**
Fist	99.14	69.40	81.65	96.10	83.53	0.91
Five	87.66	69.60	77.59	94.98		
None	99.80	100	99.90	99.98		
Okay	91.42	87.40	89.37	97.40		
Peace	76.23	84.00	79.92	94.72		
Rad	83.82	74.60	78.94	95.02		
Straight	75.38	99.80	85.89	95.90		
Thumbs	67.69	83.40	74.73	92.95		
Average	85.14	83.52	83.50	95.88		

Table 11 RF test results for *n_estimators* = 400.

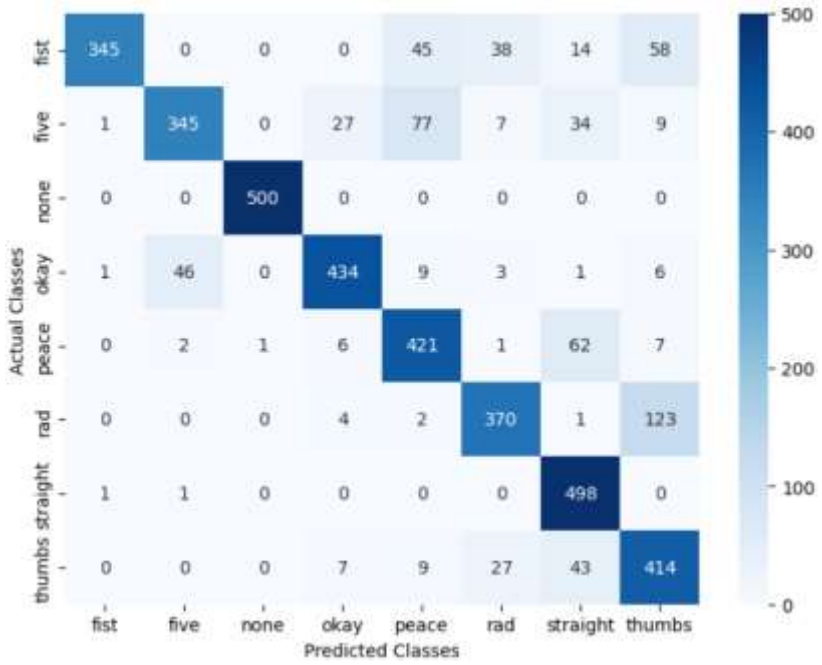


Figure 13 RF confusion matrix for *n_estimators* = 500.

Gesture Class	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)	Overall Accuracy (%)	Training Time (min)
Fist	99.14	69.00	81.37	96.05	83.18	1.16
Five	87.56	69.00	77.18	94.90		
None	99.80	100	99.90	99.98		
Okay	90.79	86.80	88.75	97.25		
Peace	74.78	84.20	79.21	94.48		
Rad	82.96	74.00	78.22	94.85		
Straight	76.26	99.60	86.38	96.08		
Thumbs	67.10	82.80	74.13	92.78		
Average	84.80	83.17	83.14	95.80		

Table 12 RF test results for *n_estimators* = 500.

4.4 CNN Results with Different Early Stopping Patience

The CNN model was trained by splitting the data into 60% for training, 20% for validation, and 20% for testing. It was trained using *patience* = 1, 2, 3, 4, 5, 6, and 7. This section gives the test results and confusion matrices for each *patience* value. The training times are also included. Figure 14 presents the results for *patience* = 1. The numbers of correctly predicted images for the eight gesture classes fist, five, none, okay, peace, rad, straight, and thumbs out of 500 were 499, 497, 500, 496, 462, 486, 456, and 472, respectively. Table 13 gives the average precision, recall, F1-score, and accuracy which are 96.93%, 96.70%, 96.72%, and 99.18%, respectively, with a training time of 3.72 min. Figure 15 presents the results for *patience* = 2. The numbers of correctly predicted images for the eight gesture classes were 479, 500, 500, 498, 479, 461, 490, and 467, respectively. Table 14 gives the average test results which are 96.90%, 96.85%, 96.86%, and 99.21%, respectively, with a training time of 4.28 min. Figure 16 presents the results for *patience* = 3. The numbers of correctly predicted images for the eight gesture classes were 491, 500, 500, 480, 494, 490, 491, and 451. Table 15 gives the average test results which are 97.50%, 97.42%, 97.42%, and 99.36%, respectively, with a training time of 4.37 min. Figure 17 presents the results for *patience* = 4. The numbers of correctly predicted images for the eight gesture classes were 495, 498, 500, 493, 484, 497, 475, and 474. Table 16 gives the average test results which are 97.98%, 97.90%, 97.91%, and 99.48%, respectively, with a training time of 5.52 min. Figure 18 presents the results for *patience* = 5. The numbers of correctly predicted images for the eight gesture classes were 497, 500, 500, 500, 498, 491, 495, and 487. Table 17 gives the average test results which are 99.20%, 99.20%, 99.19%, and 99.80%, respectively, with a training time of 5.89 min. Figure 19 presents the results for *patience* = 6. The numbers of correctly predicted images for the eight gesture classes were 496, 500, 500, 486, 496, 494, 484, and 492. Table 18 gives the average test results which are 98.72%, 98.70%, 98.70%, and 99.68%, respectively, with a training time of 6.12 min. Figure 20 presents the results for *patience* = 7. The numbers of correctly predicted images for the eight gesture classes were 500, 500, 500, 500, 493, 495, 483, and 493. Table 18 gives the average test results which are 99.12%, 99.10%, 99.10%, and 99.78%, respectively, with a training time of 8.31 min.

The highest overall accuracy of 99.20% was obtained with *patience* = 5 and a training time of 5.89 min. The lowest overall accuracy of 96.70% was obtained with *patience* = 1 and a training time of

3.72 min. The model performance gradually improved with increasing *patience* up to 5 and then remained approximately constant, indicating that *patience* = 5 is the best value.

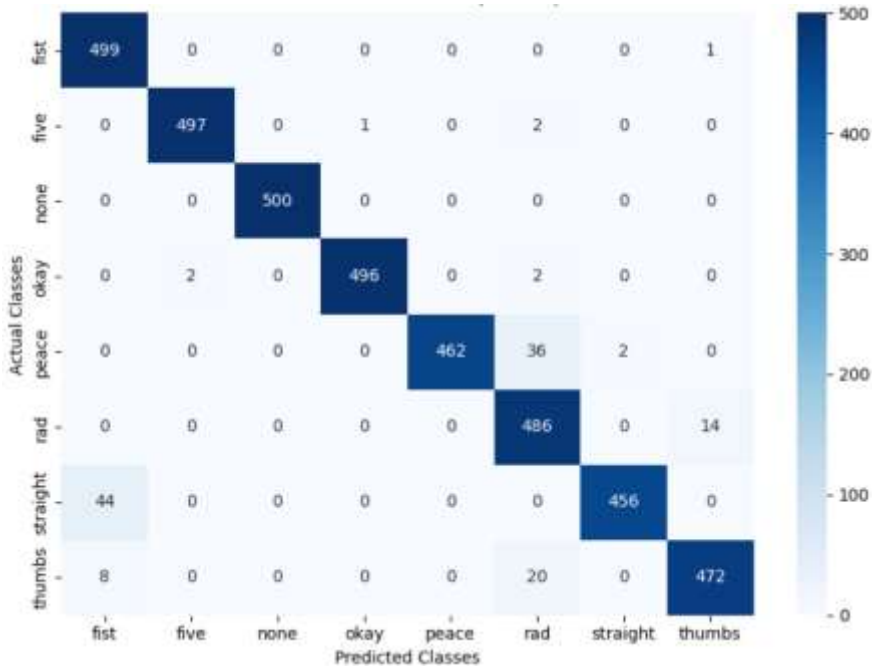


Figure 14 CNN confusion matrix for *patience* = 1.

Gesture Class	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)	Overall Accuracy (%)	Training Time (min)
Fist	90.56	99.80	94.96	98.67	96.70	3.72
Five	99.60	99.40	99.50	99.88		
None	100	100	100	100		
Okay	99.80	99.20	99.50	99.88		
Peace	100	92.40	96.05	99.05		
Rad	89.01	97.20	92.93	98.15		
Straight	99.56	91.20	95.20	98.85		
Thumbs	96.92	94.40	95.64	98.92		
Average	96.93	96.70	96.72	99.18		

Table 13 CNN test results for *patience* = 1.

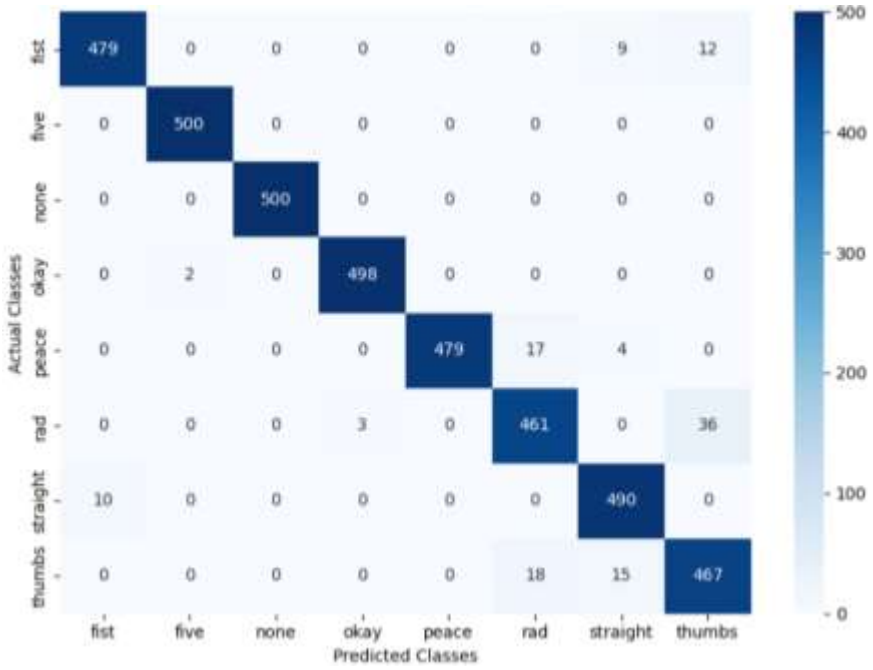


Figure 15 CNN confusion matrix for *patience* = 2.

Gesture Class	**Precision (%)**	**Recall (%)**	**F1-Score (%)**	**Accuracy (%)**	**Overall Accuracy (%)**	**Training Time (min)**
Fist	97.96	95.80	96.87	99.22	96.85	4.28
Five	99.60	100	99.80	99.95		
None	100	100	100	100		
Okay	99.40	99.60	99.50	99.88		
Peace	100	95.80	97.85	99.48		
Rad	92.94	92.20	92.57	98.15		
Straight	94.59	98.00	96.27	99.05		
Thumbs	90.68	93.40	92.02	97.97		
Average	96.90	96.85	96.86	99.21		

Table 14 CNN test results for *patience* = 2.

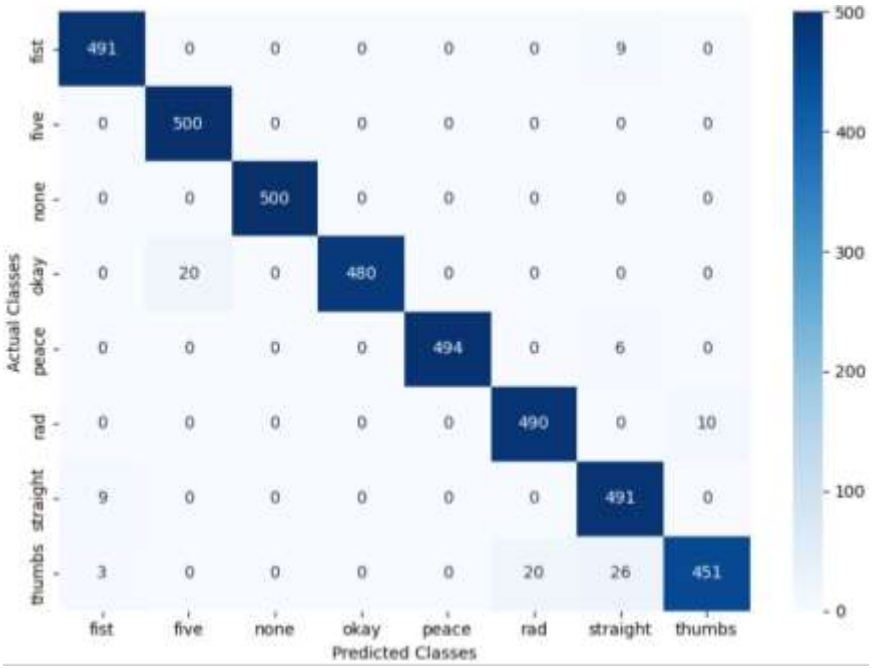


Figure 16 CNN confusion matrix for *patience* = 3.

Gesture Class	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)	Overall Accuracy (%)	Training Time (min)
Fist	97.61	98.20	97.91	99.48	97.43	4.37
Five	96.15	100	98.04	99.50		
None	100	100	100	100		
Okay	100	96.00	97.96	99.50		
Peace	100	98.80	99.40	99.85		
Rad	96.08	98.00	97.03	99.25		
Straight	92.29	98.20	95.16	98.75		
Thumbs	97.83	90.20	93.86	98.52		
Average	97.50	97.42	97.42	99.36		

Table 15 CNN test results for *patience* = 3.

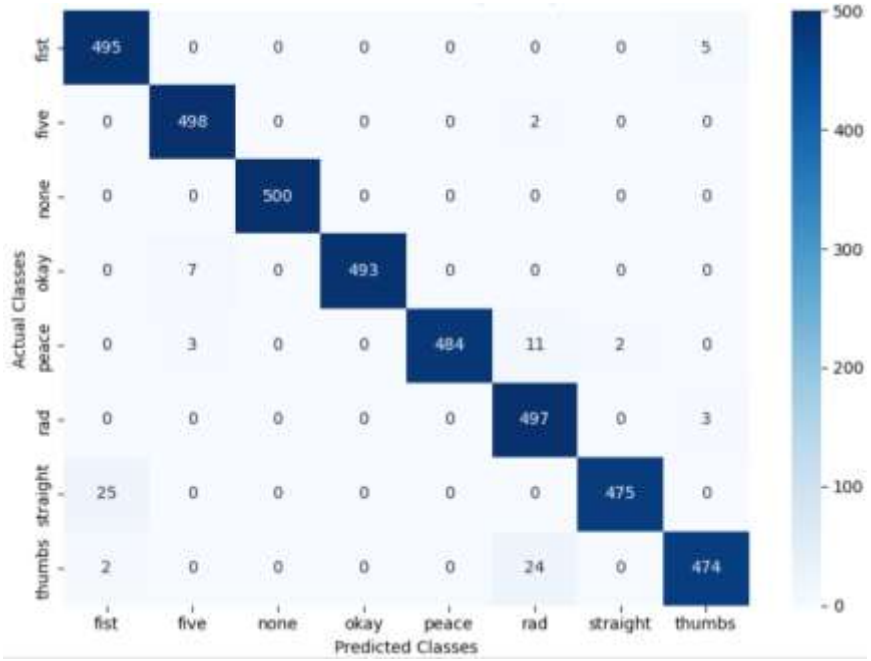


Figure 17 CNN confusion matrix for *patience* = 4.

Gesture Class	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)	Overall Accuracy (%)	Training Time (min)
Fist	94.83	99.00	96.87	99.20	97.90	5.52
Five	98.03	99.60	98.81	99.70		
None	100	100	100	100		
Okay	100	98.60	99.30	99.83		
Peace	100	96.80	98.37	99.60		
Rad	93.07	99.40	96.13	99.00		
Straight	99.58	95.00	97.24	99.33		
Thumbs	98.34	94.80	96.54	99.15		
Average	97.98	97.90	97.91	99.48		

Table 16 CNN test results for *patience* = 4.

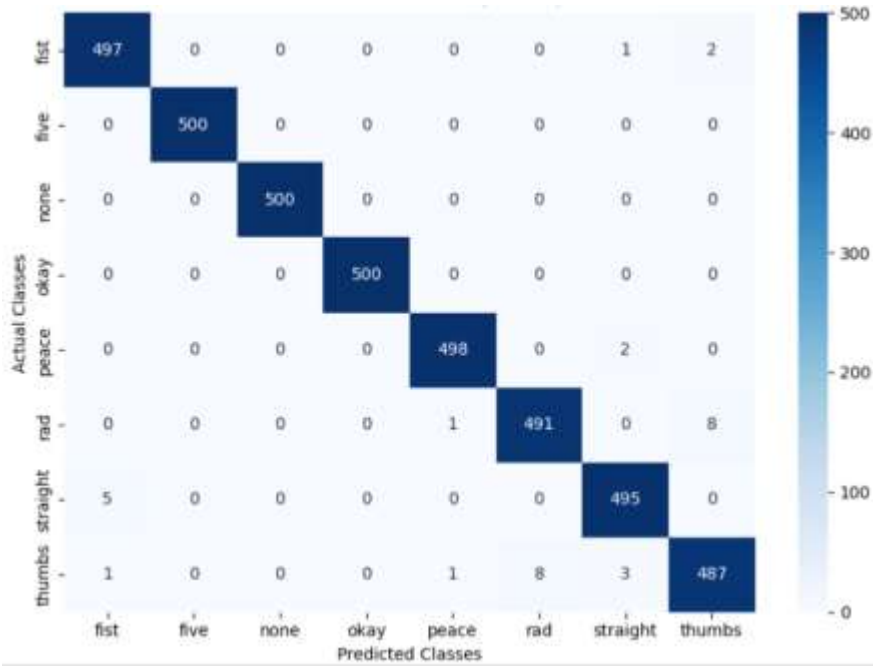


Figure 18 CNN confusion matrix for *patience* = 5.

Gesture Class	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)	Overall Accuracy (%)	Training Time (min)
Fist	98.81	99.40	99.10	99.78	99.20	5.89
Five	100	100	100	100		
None	100	100	100	100		
Okay	100	100	100	100		
Peace	99.60	99.60	99.60	99.90		
Rad	98.40	98.20	98.30	99.58		
Straight	98.80	99.00	98.90	99.72		
Thumbs	97.99	97.40	97.69	99.42		
Average	99.20	99.20	99.19	99.80		

Table 17 CNN test results for *patience* = 5.

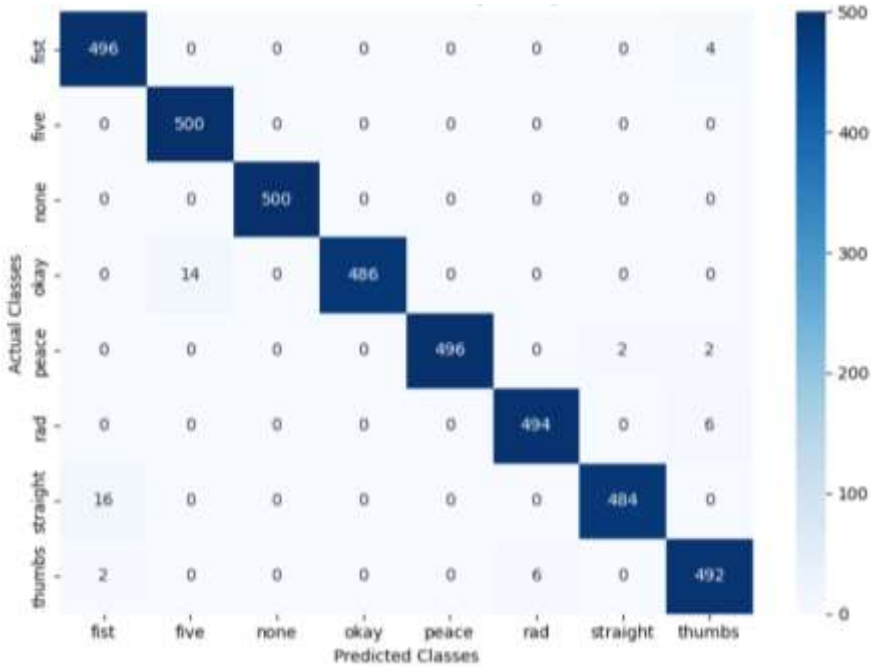


Figure 19 CNN confusion matrix for *patience* = 6.

Gesture Class	**Precision (%)**	**Recall (%)**	**F1-Score (%)**	**Accuracy (%)**	**Overall Accuracy (%)**	**Training Time (min)**
Fist	96.50	99.20	97.83	99.45	98.70	6.12
Five	97.28	100	98.62	99.65		
None	100	100	100	100		
Okay	100	97.20	98.58	99.65		
Peace	100	99.20	99.60	99.90		
Rad	98.80	98.80	98.80	99.70		
Straight	99.59	96.80	98.17	99.55		
Thumbs	97.62	98.40	98.01	99.50		
Average	98.72	98.70	98.70	99.68		

Table 18 CNN test results for *patience* = 6.

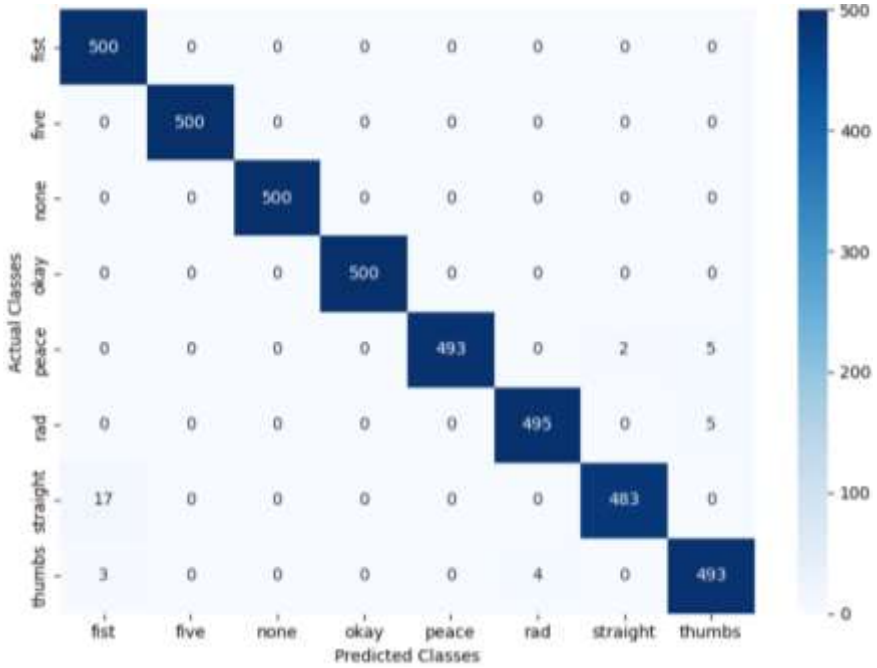


Figure 20 CNN confusion matrix for *patience* = 7.

Gesture Class	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)	Overall Accuracy (%)	Training Time (min)
Fist	96.15	100	98.04	99.50	99.10	8.31
Five	100	100	100	100		
None	100	100	100	100		
Okay	100	100	100	100		
Peace	100	98.60	99.30	99.83		
Rad	99.20	99.00	99.10	99.78		
Straight	99.59	96.60	98.07	99.52		
Thumbs	98.01	98.60	98.31	99.58		
Average	99.12	99.10	99.10	99.78		

Table 19 CNN test results for *patience* = 7.

4.5 Xception Results with Different Early Stopping Patience

The Xception model was trained by splitting the data into 60% for training, 20% for validation, and 20% for testing. It was trained using *patience* = 1, 2, 3, 4, 5, 6, and 7. This section gives the test results and confusion matrices for each *patience* value. The training times are also included. Figure 21 presents the results for *patience* = 1. The numbers of correctly predicted images for the eight gesture classes fist, five, none, okay, peace, rad, straight, and thumbs out of 500 were 443, 463, 500, 498, 498, 466, 500, and 500, respectively. Table 20 gives the average precision, recall, F1-score, and accuracy which are 97.04%, 96.70%, 96.72%, and 99.18%, respectively, with a training time of 4.34 min. Figure 22 presents the results for *patience* = 2. The numbers of correctly predicted images for the eight gesture classes were 472, 470, 500, 493, 497, 453, 500, and 491, respectively. Table 21 gives the average test results which are 97.17%, 96.90%, 96.93%, and 99.23%, respectively, with a training time of 5.48 min. Figure 23 presents the results for *patience* = 3. The numbers of correctly predicted images for the eight gesture classes were 475, 471, 500, 496, 493, 498, 500, and 485. Table 22 gives the average test results which are 98.00%, 97.95%, 97.95%, and 99.49%, respectively, with a training time of 7.82 min. Figure 24 presents the results for *patience* = 4. The numbers of correctly predicted images for the eight gesture classes were 499, 483, 500, 500, 465, 490, 500, and 498. Table 23 gives the average test results which are 98.43%, 98.38%, 98.38%, and 99.60%, respectively, with a training time of 8.49 min. Figure 25 presents the results for *patience* = 5. The numbers of correctly predicted images for the eight gesture classes were 499, 472, 500, 495, 499, 498, 500, and 500. Table 24 gives the average test results which are 99.12%, 99.07%, 99.08%, and 99.77%, respectively, with a training time of 11.02 min. Figure 26 presents the results for *patience* = 6. The numbers of correctly predicted images for the eight gesture classes were 463, 490, 500, 499, 486, 463, 500, and 471. Table 25 gives the average test results which are 97.03%, 96.80%, 96.83%, and 99.20%, respectively, with a training time of 11.79 min. Figure 27 presents the results for *patience* = 7. The numbers of correctly predicted images for the eight gesture classes were 494, 490, 500, 497, 464, 439, 500, and 500. Table 26 gives the average test results which are 97.38%, 97.10%, 97.11%, and 99.28%, respectively, with a training time of 12.13 min.

The highest overall accuracy of 99.08% was obtained with *patience* = 5 and a training time of 11.02 min. The lowest overall accuracy of 96.70% was obtained with *patience* = 1 and a training

time of 4.34 min. The model performance gradually improved with increasing *patience* up to 5 and then remained approximately constant, indicating that *patience* = 5 is the best value.

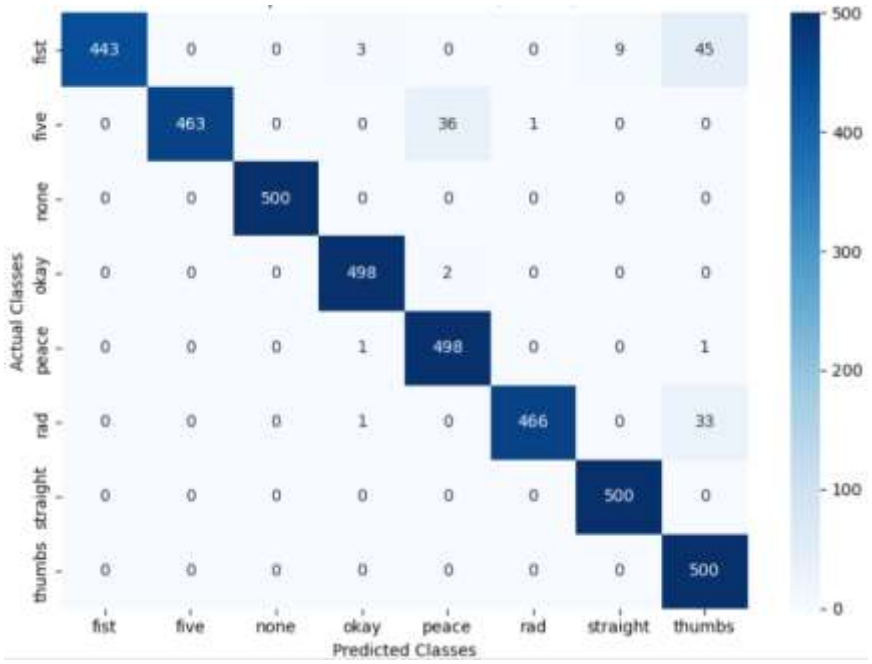


Figure 21 Xception confusion matrix for *patience* = 1.

Gesture Class	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)	Overall Accuracy (%)	Training Time (min)
Fist	100	88.60	93.96	98.58	96.70	4.34
Five	100	92.60	96.16	99.08		
None	100	100	100	100		
Okay	99.01	99.60	99.30	99.83		
Peace	92.91	99.60	96.14	99.00		
Rad	99.79	93.20	96.38	99.12		
Straight	98.23	100	99.11	99.78		
Thumbs	86.36	100	92.68	98.02		
Average	97.04	96.70	96.72	99.18		

Table 20 Xception test results for *patience* = 1.

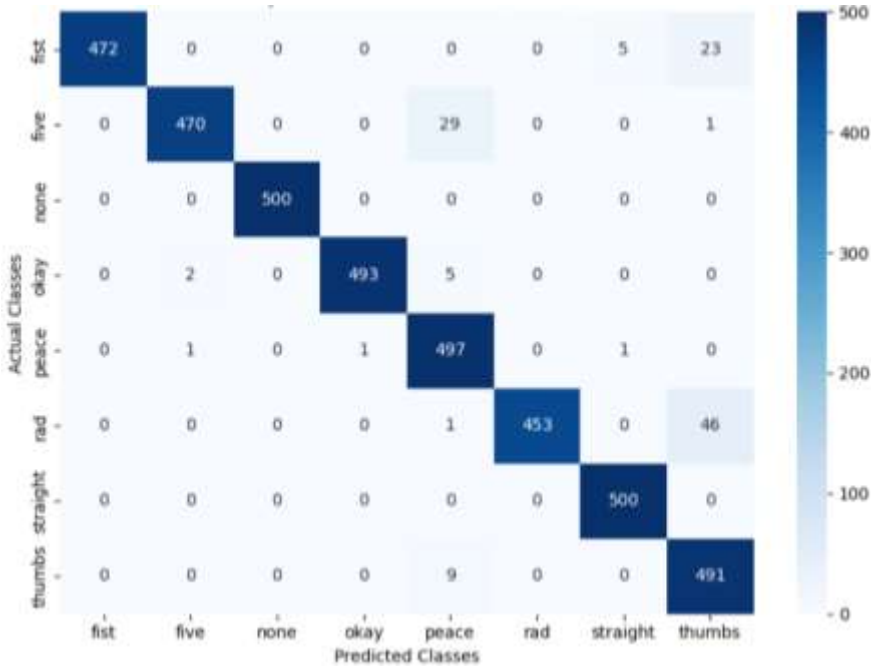


Figure 22 Xception confusion matrix for *patience* = 2.

Gesture Class	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)	Overall Accuracy (%)	Training Time (min)
Fist	100	94.40	97.12	99.30	96.90	5.48
Five	99.37	94.00	96.61	99.17		
None	100	100	100	100		
Okay	99.80	98.60	99.20	99.80		
Peace	91.87	99.40	95.49	98.83		
Rad	100	90.60	95.07	98.83		
Straight	98.81	100	99.40	99.85		
Thumbs	87.52	98.20	92.55	98.02		
Average	97.17	96.90	96.93	99.23		

Table 21 Xception test results for *patience* = 2.

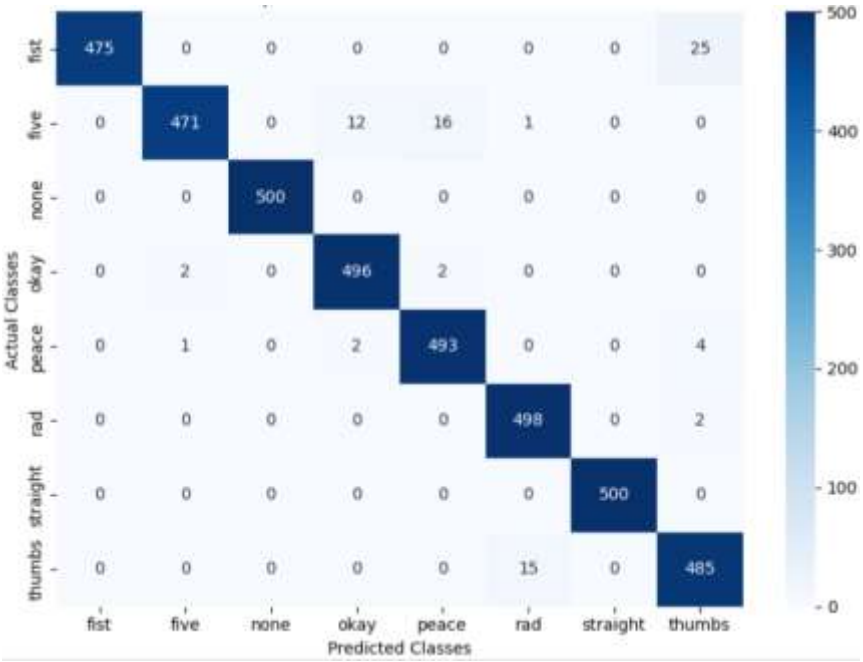


Figure 23 Xception confusion matrix for *patience* = 3.

Gesture Class	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)	Overall Accuracy (%)	Training Time (min)
Fist	100	95.00	97.44	99.38	97.95	7.82
Five	99.37	94.20	96.71	99.20		
None	100	100	100	100		
Okay	97.25	99.20	98.22	99.55		
Peace	96.48	98.60	97.53	99.38		
Rad	96.89	99.60	98.22	99.55		
Straight	100	100	100	100		
Thumbs	93.99	97.00	95.47	98.85		
Average	98.00	97.95	97.95	99.49		

Table 22 Xception test results for *patience* = 3.

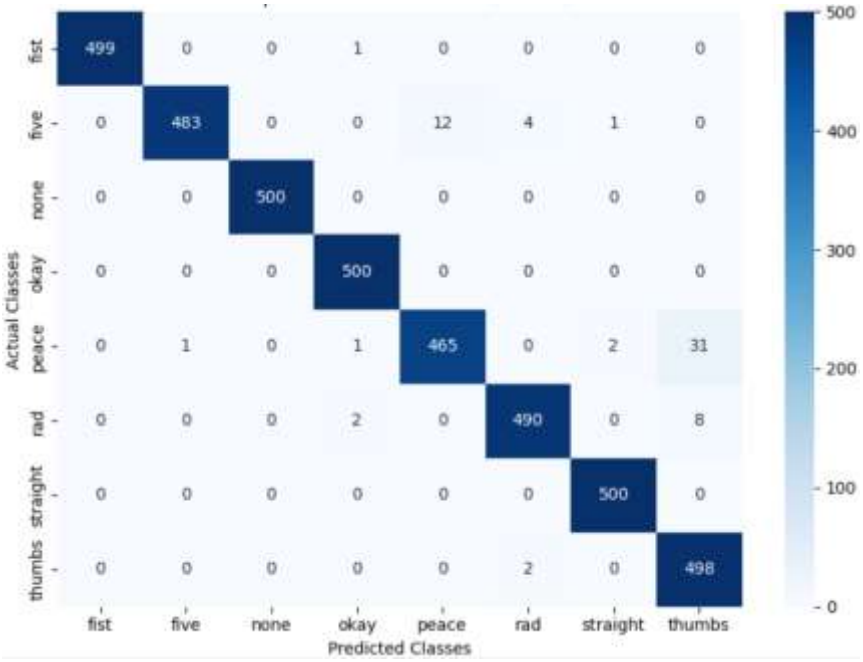


Figure 24 Xception confusion matrix for *patience* = 4.

Gesture Class	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)	Overall Accuracy (%)	Training Time (min)
Fist	100	99.80	99.90	99.98	98.38	8.49
Five	99.79	96.60	98.17	99.55		
None	100	100	100	100		
Okay	99.21	100	99.60	99.90		
Peace	97.48	93.00	95.19	98.83		
Rad	98.79	98.00	98.39	99.60		
Straight	99.40	100	99.70	99.92		
Thumbs	92.74	99.60	96.05	98.98		
Average	98.43	98.38	98.38	99.60		

Table 23 Xception test results for *patience* = 4.

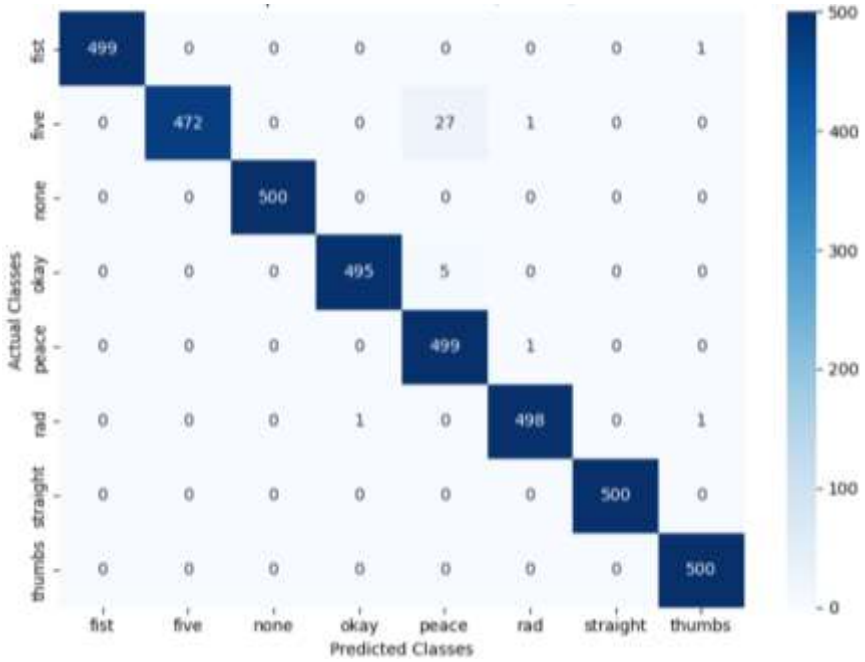


Figure 25 Xception confusion matrix for *patience* = 5.

Gesture Class	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)	Overall Accuracy (%)	Training Time (min)
Fist	100	99.80	99.90	99.98	99.08	11.02
Five	100	94.40	97.12	99.30		
None	100	100	100	100		
Okay	99.80	99.00	99.40	99.85		
Peace	93.97	99.80	96.80	99.17		
Rad	99.60	99.60	99.60	99.90		
Straight	100	100	100	100		
Thumbs	99.60	100	99.80	99.95		
Average	99.12	99.07	99.08	99.77		

Table 24 Xception test results for *patience* = 5.

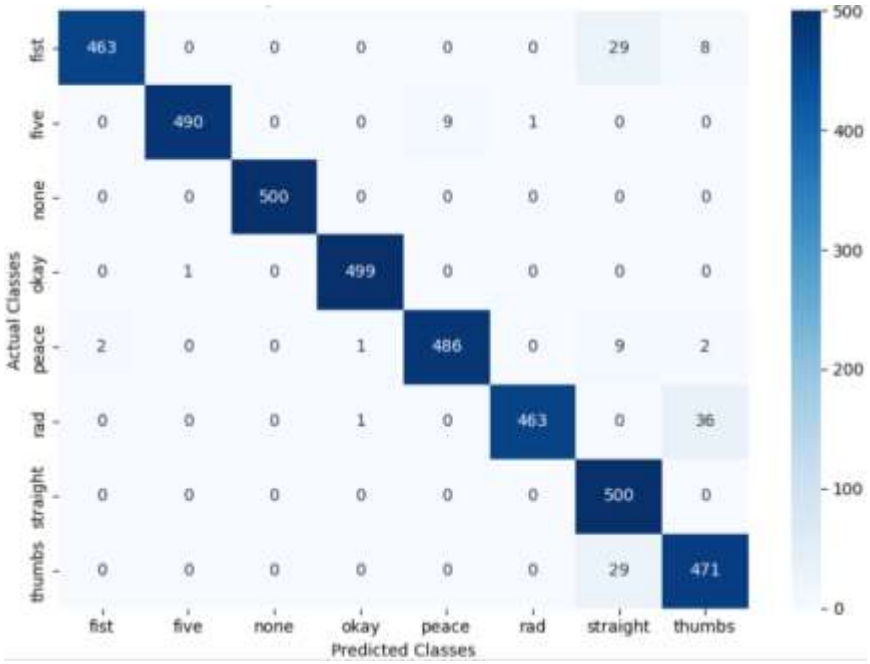


Figure 26 Xception confusion matrix for *patience* = 6.

Gesture Class	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)	Overall Accuracy (%)	Training Time (min)
Fist	99.57	92.60	95.96	99.02	96.80	11.79
Five	99.80	98.00	98.89	99.72		
None	100	100	100	100		
Okay	99.60	99.80	99.70	99.92		
Peace	98.18	97.20	97.69	99.42		
Rad	99.78	92.60	96.06	99.05		
Straight	88.18	100	93.72	98.32		
Thumbs	91.10	94.20	92.63	98.12		
Average	97.03	96.80	96.83	99.20		

Table 25 Xception test results for *patience* = 6.

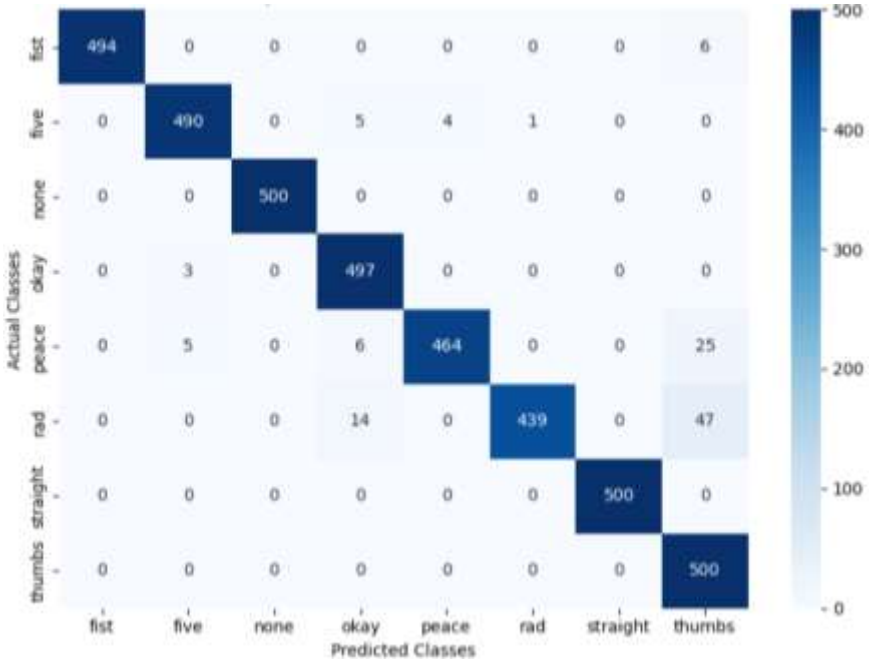


Figure 27 Xception confusion matrix for *patience* = 7.

Gesture Class	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)	Overall Accuracy (%)	Training Time (min)
Fist	100	98.80	99.40	99.85	97.10	12.13
Five	98.39	98.00	98.20	99.55		
None	100	100	100	100		
Okay	95.21	99.40	97.26	99.30		
Peace	99.15	92.80	95.87	99.00		
Rad	99.77	87.80	93.40	98.45		
Straight	100	100	100	100		
Thumbs	86.51	100	92.76	98.05		
Average	97.38	97.10	97.11	99.28		

Table 26 Xception test results for *patience* = 7.

4.6 Discussion

The best performing parameter values were selected for each model based on overall accuracy and training time. Table 27 gives the average precision, recall, and F1-score, overall accuracy, and training time for the four models using their best parameter values with augmentation. The SVM model best performance was with $C = 10$. The RF model best performance was with *n_estimators* = 200. The CNN model best performance was with *patience* = 5. The Xception model best performance was with *patience* = 5.

Model	Best Parameter	Average Precision (%)	Average Recall (%)	Average F1-Score (%)	Overall Accuracy (%)	Training Time (min)
SVM	$C = 10$	91.10	90.82	90.75	90.83	20.35
RF	*n_estimators* = 200	85.33	83.67	83.69	83.68	0.46
CNN	*patience* = 5	99.20	99.20	99.19	99.20	5.89
Xception	*patience* = 5	99.12	99.07	99.08	99.08	11.02

Table 27 Model performance with the best parameter values with augmentation.

The CNN model obtained the highest precision of 99.20%, indicating reliable predictions for all gesture classes. The Xception model obtained approximately the same precision of 99.12%, and the SVM model obtained a precision of 91.10%. The RF model obtained the lowest precision value of 85.33%.

The CNN model obtained the highest recall of 99.20%, indicating that it correctly identified almost all gesture classes. The Xception model obtained approximately the same recall of 99.07%. The SVM model obtained a recall of 90.82%, which was lower than the DL models. The RF model had the lowest recall of 83.67%.

The CNN model obtained the highest F1-score of 99.19%, indicating overall balance between precision and recall. The Xception model obtained approximately the same F1-score of 99.08%. The SVM model obtained an F1-score of 90.75%, indicating reduced balance compared to the DL models. The RF model had the lowest F1-score of 83.69%.

The CNN model obtained the highest overall accuracy of 99.20%. The Xception model obtained approximately the same overall accuracy of 99.08%. The SVM model obtained an overall accuracy of 90.83%, which was lower than the DL models. The RF model had the lowest overall accuracy of 83.68%.

The RF model required the shortest training time of 0.46 min, making it the fastest model. The CNN model had a training time of 5.89 min, which was the next shortest. The Xception model required 11.02 min, while the SVM model had the longest training time of 20.35 min.

Gesture-level performance shows that the CNN and Xception models recognised all gestures with high precision, recall, and F1-score values above 97%. The CNN model obtained 100% precision, recall, and F1-score for the five and okay, and the Xception model obtained 100% for the straight. In contrast, lower performance was observed for certain gestures using the SVM and RF models. For the SVM model, the five obtained 73.20% recall, the peace obtained 82.86% precision, and the thumbs obtained 84.42% precision. The okay also showed a reduced value with 87.70% precision. For the RF model, the thumbs obtained 67.44% precision and 73.85% F1-score, while the straight and peace obtained 74.00% and 76.41% precision, respectively. These gestures involve fewer extended fingers and more similar hand shapes, which reduces the difference between gestures and increases the chance of misclassification. Variations in thumb angle and finger bending in the dataset images also contributed to lower recognition performance. In comparison, gestures with more distinct finger separation such as five, okay, and straight were recognised more accurately, particularly by the DL models.

Table 28 gives the average precision, recall, and F1-score, overall accuracy, and training time for the four models using their best parameter values without augmentation. When compared with Table 27, all four models show reduced values without augmentation. The CNN model precision was reduced by 1.36%, and the Xception model precision was reduced by 1.49%. The SVM model precision showed a reduction of 6.16%, and the RF model precision was reduced by 3.70%. The CNN model recall was reduced by 1.40%, and the Xception model recall was reduced by 1.52%. The SVM model recall showed a reduction of 6.20%, and the RF model recall was reduced by 3.07%. The CNN model F1-score was reduced by 1.40%, and the Xception model F1-score was reduced by 1.55%. The SVM model F1-score showed a reduction of 6.29%, and the RF model F1-score was reduced by 3.23%. The CNN model overall accuracy was reduced by 1.40%, and the

Xception model overall accuracy was reduced by 1.53%. The SVM model overall accuracy showed a reduction of 6.20%, and the RF model overall accuracy was reduced by 3.07%.

Model	Best Parameter	Average Precision (%)	Average Recall (%)	Average F1-Score (%)	Overall Accuracy (%)	Training Time (min)
SVM	$C = 10$	84.94	84.62	84.46	84.62	7.12
RF	*n_estimators* = 200	81.63	80.60	80.46	80.60	0.17
CNN	*patience* = 5	97.84	97.80	97.79	97.80	4.36
Xception	*patience* = 5	97.63	97.55	97.53	97.55	8.21

Table 28 Model performance with the best parameter values without augmentation.

The SVM model training time reduced from 20.35 min to 7.12 min. The RF model training time reduced from 0.46 min to 0.17 min. The CNN model training time reduced from 5.89 min to 4.36 min. The Xception model training time reduced from 11.02 min to 8.21 min.

These results show that DL models (CNN and Xception) performed better than traditional ML models (SVM and RF) in terms of overall accuracy, precision, recall, and F1-score. Further, data augmentation improved generalization across models while increasing their training time.

Chapter 5 Conclusion and Future Work

This study compared the performance of the four supervised ML models for SLR. The dataset had 20,000 preprocessed grayscale images across eight gesture classes fist, five, okay, peace, rad, straight, thumbs, and none. All images were resized to 71×71 pixels and normalized to (0, 1). Data augmentation was applied to the training set using six transformations: rotation, width shift, height shift, shear, zoom, and horizontal flipping. For the SVM and RF models, each image was flattened into a one-dimensional vector containing 5,041 features. For the CNN and Xception models, the images were directly processed without flattening. The SVM model was trained using an RBF kernel with C = 2, 4, 6, 8, 10, 12, 14, and the best overall accuracy was 90.83% with C = 10 and a training time of 20.35 min. The RF model was evaluated using *n_estimators* = 200, 300, 400, and 500 with *max_depth* = 30, and the best overall accuracy was 83.68% with *n_estimators* = 200 and a training time of 0.46 min. The CNN model used three convolutional layers with 32, 64, 128 filters, batch normalization, dropout (0.2, 0.3, 0.4), global average pooling, and a dense layer with 256 neurons. It was trained using the Adam optimizer with a learning rate of 0.0008. The best overall accuracy was 99.20% with *patience* = 5 and a training time of 5.89 min. The Xception model used depthwise separable convolutions, global average pooling, dropout (0.5), and a dense layer with 256 neurons. It was trained using the Adam optimizer with a learning rate of 0.0008 and the best overall accuracy was 99.08% with *patience* = 5 and a training time of 11.02 min.

CNN had the highest overall accuracy and a good balance between training time and performance, followed by Xception. The traditional models SVM and RF had lower overall accuracy. Further, data augmentation improved generalization across models.

Future work can consider SLR for continuous sign recognition where full sentences and transitions between gestures are recognized instead of individual signs in isolation. Facial expressions, head movement, and body posture can be included as they are essential components of SL. Larger and more diverse datasets are needed to handle variations in signers, lighting, background, and different sign languages. Another important direction is developing lightweight models that can run in real time on mobile or edge devices. In addition, future systems should be designed with input from the deaf community to ensure accuracy, fairness, and real-world usability.

Bibliography

[1] World Health Organization (WHO), Deafness. Accessed: October 18, 2021. [Online]. Available: https://www.who.int/news-room/facts-in-pictures/detail/deafness

[2] T. Patel, "The role of American Sign Language in enhancing awareness of deaf culture," Social Science Research Network, April 30, 2025.

[3] World Federation of the Deaf, International Week of Deaf People and International Day of Sign Languages. Accessed: June 16, 2025. [Online]. Available: https://wfdeaf.org/international-week-of-deaf-people-2025/

[4] T. Poibeau, Machine Translation, Cambridge, MA, USA, MIT Press, 2017.

[5] J. M. Power, G. W. Grimm, and J.-M. List, "Evolutionary dynamics in the dispersal of sign languages," Royal Society Open Science, vol. 7, no. 1, art. 191100, January 2020.

[6] B. A. Al Abdullah, G. A. Amoudi, and H. S. Alghamdi, "Advancements in sign language recognition: A comprehensive review and future prospects," IEEE Access, vol. 12, pp. 128871–128895, September 2024.

[7] B. S. Parton, "Sign language recognition and translation: A multidisciplined approach from the field of artificial intelligence," Journal of Deaf Studies and Deaf Education, vol. 11, no. 1, pp. 94–101, October 2005.

[8] U. Farooq, M. S. M. Rahim, N. Sabir, A. Hussain, and A. Abid, "Advances in machine translation for sign language: Approaches, limitations, and challenges," Neural Computing and Applications, vol. 33, no. 21, pp. 14357–14399, November 2021.

[9] World Health Organization, Deafness and Hearing Loss, 2018. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss

[10] P. Das, T. Ahmed, and M. F. Ali, "Static hand gesture recognition for American Sign Language using deep convolutional neural network," Proc. IEEE Region 10 Symposium, pp. 1762–1765, Dhaka, Bangladesh, June 2020.

[11] Y. S. N. Rao, Y. T. Chong, R. U. Khan, C. S. Teh, M. H. Barawi, and M. S. Sunar, "Dynamic sign language recognition and translation through deep learning: A systematic literature review," Journal of Theoretical and Applied Information Technology, vol. 102, no. 21, pp. 132-145, November 2024.

[12] P. P. Urmee, M. A. A. Mashud, J. Akter, A. S. M. M. Jameel, and S. Islam, "Real-time Bangla Sign Language detection using Xception model with augmented dataset," Proc. IEEE International WIE Conference on Electrical and Computer Engineering, pp. 1450–1467, Bangalore, India, November 2019.

[13] R. K. Singh and A. K. Mishra, "Modified CNN model for hand gesture recognition using sign language," Proc. International Conference on Trends in Electronics and Informatics, pp. 604–609, Muradabad, India, June 2018.

[14] K. Myagila and H. Kilavo, "A comparative study on performance of SVM and CNN in Tanzania Sign Language translation using image recognition," Proc. International Conference on Artificial Intelligence and Image Processing, pp. 845–859, Dodoma, Tanzania, November 2020.

[15] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436–444, May 2015.

[16] A. Rahman, A. A. Khan, and T. M. Shoumik, "A novel lightweight CNN approach for Bangladeshi sign language gesture recognition," Bachelor's thesis, Dept. of Computer Science and Engineering, Brac University, Dhaka, Bangladesh, May 2022.

[17] S. Shalev-Shwartz and S. Ben-David, Understanding Machine Learning: From Theory to Algorithms, Cambridge University Press, Cambridge, UK, July 2014.

[18] T. K. Ho, "Random decision forests," Proc. International Conference on Document Analysis and Recognition, pp. 278–282, Montreal, QC, Canada, August 1995.

[19] T. M. Mitchell, Machine Learning, McGraw-Hill, New York, NY, USA, March 1997.

[20] A. Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd ed., O'Reilly Media, Santa Rosa, CA, USA, September 2019.

[21] C. Cortes and V. Vapnik, “Support-vector networks,” Machine Learning, vol. 20, no. 3, pp. 273–297, September 1995.

[22] N. Cristianini and J. Shawe-Taylor, An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods, Cambridge University Press, Cambridge, UK, March 2000.

[23] W. Luo, Y. Li, R. Urtasun, and R. Zemel, “Understanding the effective receptive field in deep convolutional neural networks,” Proc. Conference on Neural Information Processing Systems, 2016, pp. 4898–4906, Barcelona, Spain, December 2016.

[24] W. Rawat and Z. Wang, “Deep convolutional neural networks for image classification: A comprehensive review,” Neural Computing and Applications, vol. 29, no. 11, pp. 495–505, August 2017.

[25] F. Chollet, “Xception: Deep learning with depthwise separable convolutions,” Proc. IEEE Conference on Computer Vision and Pattern Recognition, pp. 1251–1258, Honolulu, HI, USA, July 2017.

[26] L. Breiman, “Random forests,” Machine Learning, vol. 45, no. 1, pp. 5–32, January 2001.

[27] L. Xu, S. Yoon, A. Fuentes, D. S. Park, “A comprehensive survey of image augmentation techniques for deep learning,” Journal of Big Data, vol. 9, art. 76, May 2022.

[28] L. Perez and J. Wang, “The effectiveness of data augmentation in image classification using deep learning,” Machine Learning, vol. 107, no. 1, pp. 523–555, December 2017.

[29] K. Dimililer and D. Kayali, “Mask detection and categorization during the COVID-19 pandemic using deep convolutional neural network,” Ingeniería e Investigación, vol. 43, no. 3, pp. 145–154, September 2023.

[30] K. Pawlasová, I. Karafiátová, and J. Dvořák, “Neural networks with functional inputs for multi-class supervised classification of replicated point patterns,” Advances in Data Analysis and Classification, vol. 18, pp. 705–721, September 2024.