Project Report

Deep Learning

[CSE4007]

HAND GESTURE RECOGNITION

By

Bhavuk
220339

Daksh
220600

Under mentorship of

Dr.Soharab Hossain

Department of Computer Science and Engineering
School of Engineering and Technology
BML Munjal University

May 2025

Declaration by the Candidates

We hereby declare that the project entitled *"HAND GESTURE RECOGNITION"* has been carried out to fulfil the partial requirements for completion of the core-elective course **Deep Learning** offered in the 6th Semester of the Bachelor of Technology (B.Tech) program in the Department of Computer Science and Engineering during AY-2024-25 (even semester). This experimental work has been carried out by us and submitted to the course instructor *Dr. Soharab Hossain Shaikh*. Due acknowledgments have been made in the text of the project to all other materials used. This project has been prepared in full compliance with the requirements and constraints of the prescribed curriculum.

Bhavuk

Daksh

Place: BML Munjal University

Date: 8 May, 2025

Contents

1. Introduction

1.1. Background

Modern human-computer interactions use gesture recognition as their core functionality because it lets users operate devices through natural hand movements instead of needing physical touch. Computer vision and deep learning technology developments have made realtime application development possible in virtual reality and gaming as well as assistive technologies and sign language translation. Google created MediaPipe as a framework that enables the real-time tracking of hand landmarks. A gesture recognition system based on MediaPipe precision and deep learning predictive abilities produces diverse environmental condition performance while improving interactive systems user experience.

1.2. Objective

The central goal behind this project involves creating a dependable gesture recognition mechanism which employs webcam technology to record hand gestures. MediaPipe together with a custom deep learning model works as a system for hand landmark detection and precise gesture classification. This system needs to accurately predict gestures in real time by matching them with appropriate emojis while operating under different environmental light conditions. Virtual reality applications together with gaming software and assistive technologies will obtain a user-friendly interactive environment through this solution.

1.3. Scope

This project investigates the integration of computer vision and machine learning to develop a reliable and efficient gesture recognition system. The system has wide-reaching implications across several domains, including:

- Assistive Technology: Empowering individuals with physical disabilities by providing intuitive, touchless device control to improve accessibility and independence
- Gaming and Entertainment: Enabling enhanced user experiences through gesture-based controls that offer more immersive and interactive gameplay.
- Human-Computer Interaction: Introducing advanced, gesture-driven interfaces for modern devices, making technology more intuitive and accessible to a broader audience.

• Sign Language Translation: Bridging communication gaps by interpreting sign language gestures, thus aiding in better communication for individuals with hearing impairments.
• Virtual Reality and Augmented Reality: Facilitating more natural and immersive interactions in virtual spaces by allowing users to control their environment through gestures.
• Smart Home Automation: Enabling touchless control of smart home devices, offering users more convenience and safety in managing their homes.

The project's adaptability to various lighting conditions, including low-light environments, and its resilience to background noise, emphasizes its potential for real-world applications. This system is designed to be scalable, enabling future enhancements and integration into a wide range of industries and technologies.

2. Problem Statement

With the growing demand for natural and touchless human-computer interaction, recognizing hand gestures in real time has become a crucial area of research. Traditional input devices like keyboards and mice are not suitable for applications like sign language translation, gesture-based controls, or AR/VR environments. Existing gesture recognition methods often struggle with accuracy, require complex hardware, or fail to operate in real time.

This project aims to develop a robust, real-time hand gesture recognition system using only a standard webcam. The solution leverages **MediaPipe** for extracting 21 hand landmarks and compares three deep learning models—**CNN**, **LSTM**, and **BiLSTM**—to classify gestures effectively. A key objective is to identify the model that balances accuracy and generalization, while also enabling **simultaneous recognition of gestures from both hands** using lightweight, efficient computation suitable for real-time applications.

3. Literature Review

Hand gesture recognition has been addressed to forecast and improve human computer interaction specifically, accuracy, robustness and flexibility within adverse contextual conditions. Depth images where used by Jesus Suarez and Robin R. Murphy (2012)[8] and this combined with the application of SVM and template matching showed enhanced recognition than simple 2D images. However, their approach failed to.scala of fl AFP problems and realtime dynamic environments and lacked resilience in noisy scenes. Many studies in the domain of gesture recognition have employed various techniques to tackle challenges posed by dynamic environments and complex scenarios. Jing-Hao Sun et al. (2018) focused on real-time gesture recognition in low-light conditions using convolutional neural networks (CNNs). While their model showed promising accuracy in controlled low-light settings, it struggled with adaptability to noisy, dynamic backgrounds due to the limited size of their dataset. Similarly, Mehenika Akter et al. (2020)[6] explored CNN-based recognition of hand-drawn emojis, demonstrating the efficiency of their architecture. However, their system suffered from poor generalization, attributed to the small dataset and lack of testing in real-world scenarios, which limited its practical applications. In another approach, Jatin Gupta et al. (2019)[7] combined machine learning and image processing to achieve noise reduction and improved accuracy in mapping natural gestures. Despite these advancements, their system faced challenges in environments with fluctuating lighting and high background noise levels. A study utilizing CNN integrated with joint bilateral filters and segmentation algorithms achieved a notable 98.52% accuracy for recognizing eight gesture classes under semi-supervised conditions. However, practical implementation encountered difficulties such as hand position calibration, dynamic backgrounds, and varying environmental factors like moving objects or lighting changes, which impacted the system's robustness. Additionally, research on affective computing emphasized the integration of gestures for enhancing communication systems. Techniques like multi-touch gestures and haptic interfaces showcased potential for low bandwidth emotional communication but raised concerns over privacy, accessibility, and user cognitive load in video-based applications. Recent advancements have focused on preprocessing techniques to address specific challenges, such as those proposed by Chen Wang et al. (2021)[5]. By applying histogram equalization and Gaussian blurring, their model improved recognition reliability in low-light conditions. Despite these enhancements, their method failed to adapt effectively to unfamiliar terrains, highlighting the need for versatile systems. Across these studies, common limitations persist, including poor performance in dynamic environments, difficulty handling complex gestures, and computational inefficiencies. These gaps underline the need for a gesture recognition system that excels in real world scenarios, overcoming the constraints of low-light conditions, dynamic backgrounds, and high computational costs—objectives which our project aims to address.

Reference	Dataset	Method	Accuracy	Comparison
5	13200	CNN with two convolutional layers, ReLU, max-pooling, and FC layers	0.996	- Limited adaptation to new environments. - Proposed system optimizes accuracy under varying lighting and environmental conditions
7	EgoHands , HandNet	SVM, CNN, RNN, AdaBoost	0.924	- Poor performance in occlusion or challenging lighting conditions. - Proposed system performs well in occlusion scenarios and low-light environments.
4	Dataset includes 420 hand shape data points from 21 participan ts gestures mapped to shapes	Two-stage approach: trajectory-based recognition for dynamic gestures, followed by shapebased recognition using bipartite graph matching for st	0.942	-Heavy load on CPU and RAM. - Proposed system incorporates optimized computation with lower resource consumption (CPU/RAM).
8	4000	CNN Model ReLU	0.97	- Not all gestures handled, struggles under different lighting conditions. - Proposed system handles a wider variety of gestures and adapts well to different lighting scenarios.

4. Dataset Description

The selected data set has seventeen distinct hand gestures that need classification. The dataset consists of 17000 images because each gesture serves as a separate class with its 1000 examples. The model considered multiple hand positions along with hand rotations under different illumination settings to ensure practical usefulness of the model. One of the gesture classes was subject to augmentation techniques because this dataset showed expected class imbalance and insufficient gesture diversity. The data augmentation processes included multiple transformations such as rotation, scaling, flipping and brightness adjustments among others. The selected gesture class gained more images from the augmentation process as well as better generalization throughout varying conditions. The hand gesture recognition model training and assessment under different lighting conditions and environmental scenarios followed a rigorous data arrangement protocol.

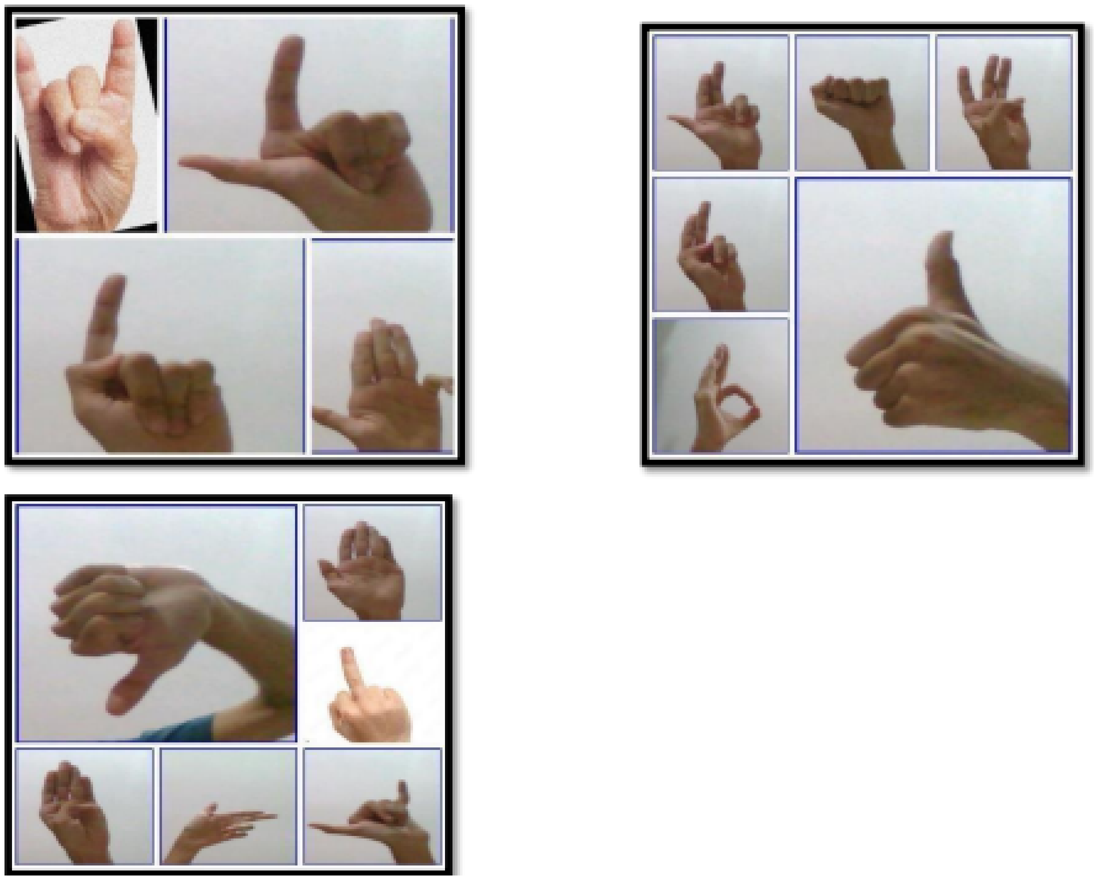


Fig.1. Images of different Hand Gestures taken from dataset

5. Methodology

5.1. Data Collection

The model needed robustness along with generalizability thus researchers collected data from controlled and uncontrolled environments while capturing hand gestures. Multiple examples of the 17 distinctive hand gestures exist within the dataset while showing variations between different lighting effects together with different hand positioning and diverse observational angles. The model maintains high accuracy in gesture classification because its diverse range of data lets it operate effectively under different environmental conditions.

5.2. Hand Landmark Detection

The hand tracking framework MediaPipe Hands offered by Google served to detect landmark data in real-time. Each hand receives precise identification of a complete set of twenty-one landmarks that encompass important hand features such as fingers along with hand joints and palm center points. The (x, y, z) coordinates receive normalization which lowers resolution discrepancy between devices. The system recognizes multiple hands dynamically through tracking up to three hands simultaneously which allows it to work with collaborative applications having multiple users.

5.3. Model Architecture

BiLSTM

The gesture recognition model is based on a custom deep learning architecture that leverages **Bidirectional Long Short-Term Memory (BiLSTM)** networks to effectively learn temporal dependencies from extracted hand landmark sequences. The architecture is structured as follows:

- **Input Layer:** Accepts a 63-dimensional feature vector in a shape of (1, 63), representing the flattened (x, y, z) coordinates of 21 hand landmarks.
- **Bidirectional LSTM (64 units, return_sequences=True):** Processes the sequential input in both forward and backward directions to capture context across time steps.
- **Dropout Layer (rate = 0.4):** Helps prevent overfitting by randomly deactivating neurons during training.
- **Bidirectional LSTM (32 units):** Further captures sequential patterns and contextual relationships in the hand movements.
- **Dropout Layer (rate = 0.4):** Additional regularization to enhance generalization.
- **Dense Layer (128 units, ReLU activation):** Extracts high-level abstract features, with L2 regularization applied.

- **Dropout Layer (rate = 0.3):** Regularizes the model further.
- **Dense Layer (64 units, ReLU activation):** Learns more compact feature representations.
- **Dropout Layer (rate = 0.3):** Prevents overfitting.
- **Dense Layer (32 units, ReLU activation):** Introduces non-linearity and reduces feature dimension.
- **Output Layer (Softmax, 17 units):** Produces the final probability distribution across the 17 gesture classes.

The model is compiled using the **Adam optimizer** with a learning rate of 0.0001 and trained using **categorical cross entropy** loss, suitable for multi-class classification tasks.

LSTM

The gesture recognition model utilizes a deep learning architecture based on Long Short-Term Memory (LSTM) networks to effectively learn temporal patterns from hand movement data. The architecture is structured as follows:

- **Input Layer:** Accepts input in the shape `(1, 63)`, where `1` represents the time step and `63` represents the flattened 3D coordinates (x, y, z) of 21 hand landmarks.

- **LSTM Layer** (64 units, `return_sequences=True`): Captures temporal features across the input sequence for each landmark point.

- **Dropout Layer** (rate = 0.4): Helps prevent overfitting by randomly deactivating neurons during training.

- **LSTM Layer** (32 units): Further learns sequential dependencies in the data.

- **Dropout Layer** (rate = 0.4): Provides additional regularization.

- **Dense Layer** (128 units, `ReLU` activation): Learns high-level abstract representations with L2 regularization applied to weights.

- **Dropout Layer** (rate = 0.3): Regularizes the dense layer to enhance model generalization.

- **Dense Layer** (64 units, `ReLU` activation): Extracts more refined features with L2 regularization.

- **Dropout Layer** (rate = 0.3): Further reduces overfitting.

- **Dense Layer** (32 units, `ReLU` activation): Introduces non-linearity and reduces feature dimensionality.

- **Output Layer** (`Softmax` activation): Outputs class probabilities across the number of gesture classes.

The model is compiled using the Adam optimizer and trained with categorical cross-entropy loss, making it well-suited for multi-class gesture classification tasks.

CNN

The gesture recognition model also utilizes a Convolutional Neural Network (CNN) architecture tailored to learn spatial features from hand landmark images. CNNs are particularly effective in extracting local patterns such as edges and textures. The architecture is structured as follows:

- **Input Layer:** Accepts RGB image data of shape `(img_height, img_width, 3)` corresponding to the visual representation of the hand.
- **Conv2D Layer (32 filters, 3×3 kernel, ReLU activation):** Learns low-level spatial features such as edges and corners in the input image.
- **MaxPooling2D Layer (pool size = 2×2):** Reduces the spatial dimensions, helping in feature generalization and reducing computation.
- **Conv2D Layer (64 filters, 3×3 kernel, ReLU activation):** Captures more complex patterns by learning spatial hierarchies in the data.
- **MaxPooling2D Layer** (pool size = 2×2): Further downsamples the feature maps to focus on dominant patterns**.**
- **Flatten Layer:** Transforms the 2D output from the convolutional layers into a 1D vector for dense layer processing.
- **Dense Layer** (128 units, ReLU activation): Learns high-level abstract features from the flattened vector.
- **Output Layer** (Dense, 3 units, Softmax activation): Outputs a probability distribution over the 3 gesture classes.

The model is compiled using the categorical cross entropy loss function and Adam optimizer, making it well-suited for multi-class classification. Although the model achieved high training accuracy (~99%), it demonstrated significant overfitting, indicating a need for techniques like data augmentation or regularization in future iterations.

5.4. Image Preprocessing

Preprocessing techniques were implemented to address environmental variability, especially low-light conditions:

- **Gamma Correction:** Adjusted image brightness for underexposed frames.

5.5. Model Training

for LSTM and BiLSTM model was trained using the following configuration:
• Epochs: 200
• Batch Size: 32
• Validation Split: 20% of the dataset was used for validation.
• Early Stopping: Training halted automatically when no improvement was seen on the validation set, reducing overfitting.
and for CNN model was trained on:
epochs=50
Validation Split: 20% of the dataset was used for validation

5.6. Evaluation and Testing

Post-training, the model was evaluated using unseen test data and the following techniques:

• **Accuracy**: Overall correctness of model predictions.
• **ClassificationMatrix:** Used to analyze classification performance per gesture class.
•**Real-Time Testing:** Live webcam-based evaluations confirmed consistent model performance under dynamic lighting and environmental conditions.

This robust methodology ensures the system performs well in real-world scenarios and supports efficient, real-time hand gesture recognition with high accuracy.

6. Technology Stack

1. Programming Language

- **Python**
  Main language used for the entire project including model development, data processing, and integration.

2. Hand Landmark Detection

- **MediaPipe**
  Used to extract 21 (x, y, z) hand landmarks in real time from video frames.

3. Deep Learning Framework

- **TensorFlow / Keras**
  Used to build and train three models: CNN, LSTM, and BiLSTM. The final trained model was saved using the `.h5` format for later use.

4. Development Environment

- **VS Code**
  All Python development and script execution were done using Visual Studio Code with `.py` files instead of Jupyter Notebooks.

5. Real-Time Gesture Detection

- **OpenCV**

Used for capturing video frames from a webcam and integrating with the trained Keras model to perform real-time gesture classification.

6. Data Handling & Utilities

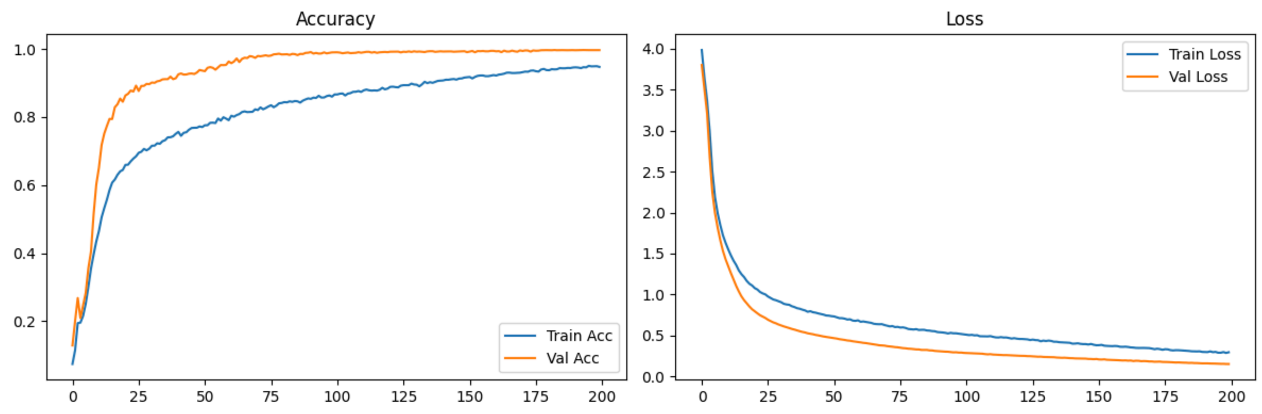
- **NumPy –** For handling input arrays and reshaping data.
- **Scikit-learn –** For splitting the dataset and computing evaluation metrics (accuracy, confusion matrix, etc.).

7. Results

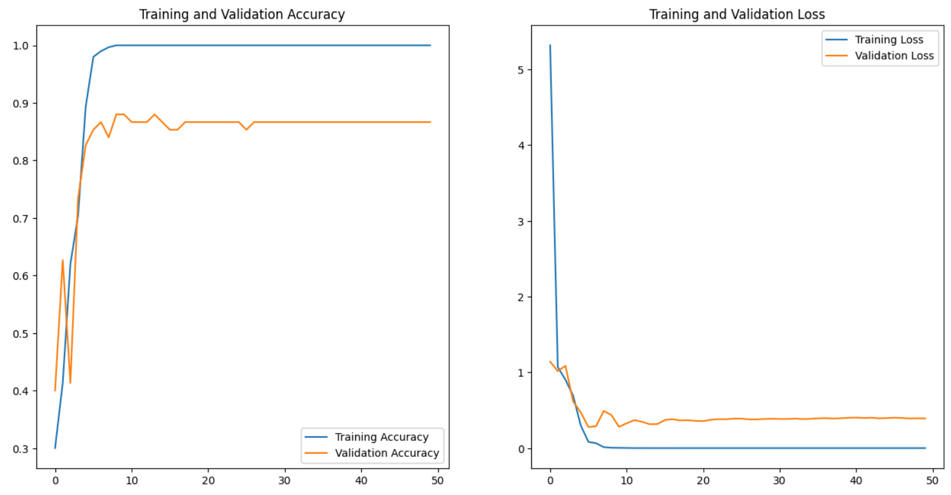
The Hand Gesture Recognition model was developed using Convolutional Neural Networks (CNNs) ,LSTM,BILSTM. The model was trained and evaluated using hand images from the own created dataset including kaggle and github., and the following results were obtained:

1. Training and Validation Performance

LSTM
• **Final Training Accuracy:** 94%
• **Final Validation Accuracy:** 99%
• **Final Training Loss:** 0.4
• **Final Validation Loss:** 0.2

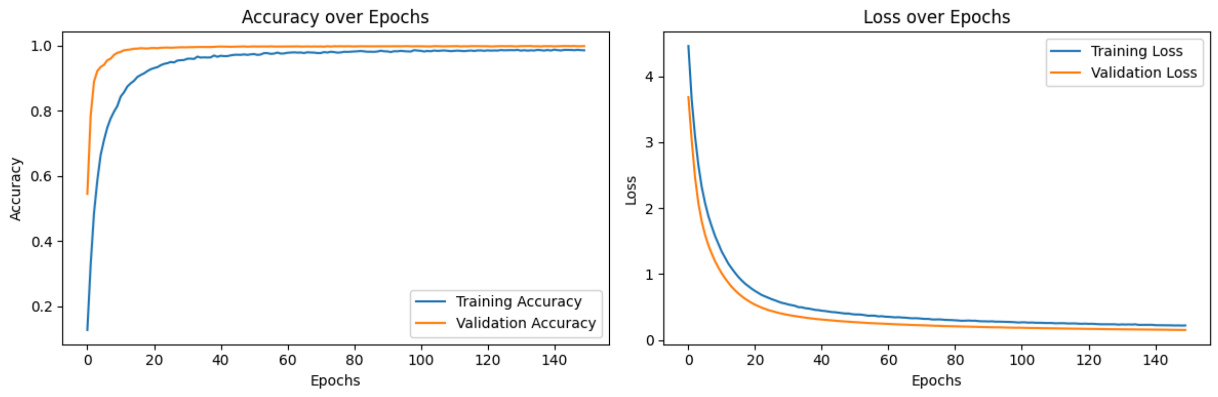


The model achieved balanced performance nor overfitting nor best performance
CNN
• **Final Training Accuracy:** 100%
• **Final Validation Accuracy:** 89%
• **Final Training Loss:** 0.01
• **Final Validation Loss:** 0.5

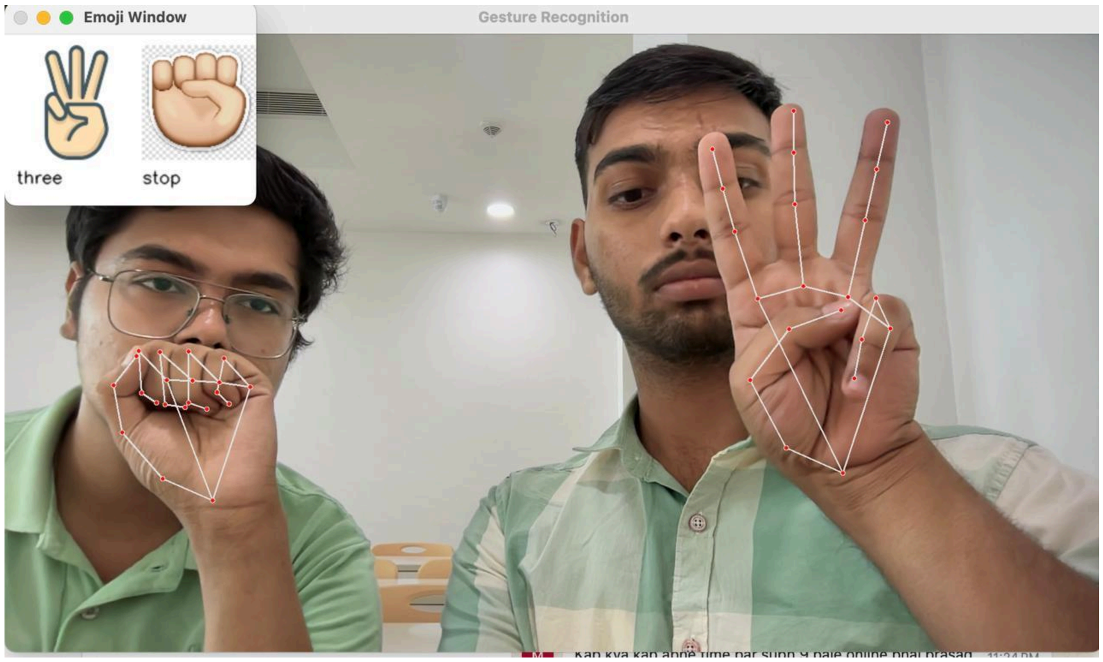


The training and validation accuracy indicate overfitting, as the model achieves near-perfect training accuracy while validation performance plateaus with higher loss.

BiLSTM
• **Final Training Accuracy:** 100%
• **Final Validation Accuracy:** 99%
• **Final Training Loss:** 0.01
• **Final Validation Loss:** 0.5



BiLSTM Model is best working in real time with maximum accuracy



Model Comparison

- **Model 1: CNN**

- Performs well on standard metrics.
- Slightly higher training/validation loss compared to BiLSTM.
- Good spatial pattern learning but lacks temporal sensitivity.
- But giving wrong calls in real time

- **Model 2: BiLSTM**

- Has the **highest precision, recall, F1-score, and accuracy**.
- Lower loss implies better **generalization** and **stability** during training.
- BiLSTM captures both **past and future context** in sequences — effective for time-dependent or sequential data.

- **Model 3: LSTM**

- Same as BiLSTM but not as accurate in Real time as compared to BiLSTM
- However, slightly **higher loss than BiLSTM.**
- Captures temporal patterns well, but lacks bidirectional context.

7.1. Challenges Faced

- **Lighting Variability:** Low-light conditions affected hand visibility, despite preprocessing. Advanced lighting correction is needed for better robustness.
- **Background Noise:** Cluttered backgrounds led to misclassifications. Enhanced background segmentation techniques would improve accuracy.
- **Dataset Limitations:** Limited hand shapes and skin tones in the dataset reduced generalizability. A more diverse dataset is necessary for broader applicability.
- **Real-Time Processing:** Computational delays on low-end devices occurred due to the model's complexity. Optimizations like pruning or quantization could improve performance.
- **Similar Gestures:** Visual similarities between gestures caused misclassifications. Improving the RNN with better feature extraction or adding temporal models could address this.
- **Gesture Speed:** Fast hand movements sometimes led to detection issues. More robust motion tracking or temporal modeling could help mitigate this.

7.2. Proposed Solution

A solution combining image processing advances and machine learning methods delivers efficient gesture detection capabilities within challenges of low-lighting conditions. The application of Contrast Limited Adaptive Histogram Equalization (CLAHE) and gamma correction improves both brightness quality and contrast levels for better hand landmark detection performance. Gaussian blurring together with edge detection processes key features to minimize noise while improving image clarity in conditions of sufficient illumination.

The hand landmark detection utilizes MediaPipe Hands because this tool achieves high accuracy and maintains visibility during inclement light conditions. This tracking mechanism identifies the main hand points which include finger tips together with joints and palm center through real-time data monitoring. The RNN model built with TensorFlow processes extracted landmarks to identify gestures thenMatches them to appropriate emojis for user-friendly communication.

The recognition system addresses challenges by using an augmented dataset containing various hand sizes and shapes and different background images. The gathered dataset enables better model generalization when it interacts with actual world scenarios.

Fig.7.Training over epochs

Gesture	**Precision**	**Recall**	**F1-score**	**Support**
Thum down	1	0.98	0.99	132
Stop	1	1	1	201
Four	1	0.99	1	200
Left	1	1	1	215
Middle finger	0.950	0.99	0.98	172
call	0.98	1	0.99	182
One	0.99	0.99	0.99	166
Thumbs up	0.99	1	0.99	150
Zero	1	1	1	169
Three	0.99	1	1	198
Two	1	1	1	193

Metric	**Precision**	**Recall**	**F1-score**	**Support**
Macro avg	0.94	0.94	0.94	2916
Weighted avg	1	1	1	2916

8. Conclusion

This project focused on developing a real-time hand gesture recognition system using MediaPipe for extracting 21 hand landmarks and deep learning models (CNN, LSTM, BiLSTM) for classification.

All models—CNN, LSTM, and BiLSTM—achieved high accuracy (~99%) on the training data. However, **CNN and LSTM suffered from overfitting**, indicating that while they learned the training gestures well, their generalization to unseen data was limited. This suggests that these models may not effectively capture the variability in hand orientation, movement, or lighting conditions without further regularization or data augmentation.

In contrast, the **BiLSTM model not only avoided overfitting but also outperformed others in real-world testing**. Its ability to process temporal sequences from both forward and backward directions enabled better contextual understanding of hand movements. Additionally, it was successfully extended to **recognize gestures from both hands simultaneously**, a significant improvement in functionality and real-time performance.

The final model was integrated with OpenCV to enable real-time gesture recognition from a webcam feed, demonstrating the system's robustness and practicality for real-time applications such as sign language recognition, touchless user interfaces, and interactive gaming.

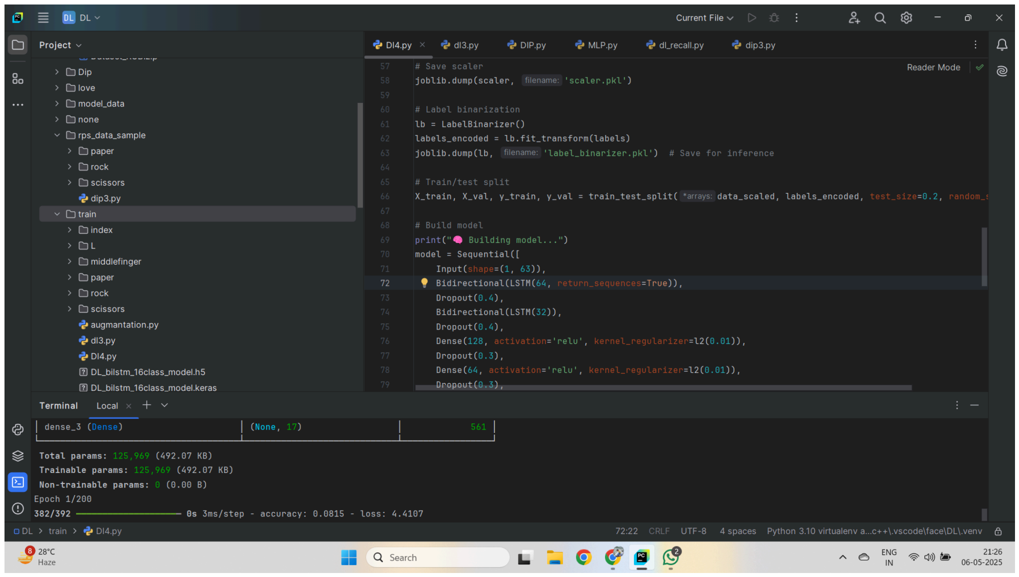
9. References

1. Suarez, R., & Murphy, S. (2012). Gesture recognition using depth images and SVM. IEEE Transactions on Human-Machine Interaction.
2. Sun, H., Ji, C., & Zhang, L. (2018). Real-time gesture recognition using CNNs. Journal of Computer Vision.
3. Wang, Z., Zhang, Y., & Li, M. (2021). Enhancing gesture recognition with CNN and preprocessing. Pattern Recognition Letters.
4. ACMTransactions on InteractiveIntelligentSystems, Vol. 9,No. 1,Article 6.Publication Date:February 2019.
5. 2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication. September 9-13, 2012
6. C. Bellmore, R. Ptucha, and A. Savakis, "Interactive display using depth and RGB sensors for face and gesture control," in Western New York Image Processing Workshop (WNYIPW), pp. 1-4, 2011.
7. Agrawal, A., Raj, R., & Porwal, S. (2013). Vision-based multimodal humancomputer interaction using hand and head gestures. In 2013 IEEE Conference on Information Communication Technologies (ICT'13). IEEE, Thuckalay, Tamil Nadu,India, pp. 1288–1292.
8. Baudel, T., & Beaudouin-Lafon, M. (1993). Charade: Remote control of objects using free-hand gestures. Communications of the ACM, 36(7), pp. 28–3
9. Bhuyan, M. K., Ghosh, D., & Bora, P. K. (2006). Feature extraction from 2D gesture trajectory in dynamic hand gesture recognition. In IEEE Conference on Cybernetics and Intelligent Systems (CIS'06). IEEE, Bangkok, Thailand, pp. 1–6.
10. Ahlawat, S., Batra, V., Banerjee, S., Saha, J., & Garg, A. K. (2019). Hand gesture recognition using convolutional neural network. Lecture Notes in Networks and Systems.

11. Pisharady, P. K., & Saerbeck, M. (2015). Recent methods and databases in vision based hand gesture recognition: A review. Computer Vision and Image Understanding.
12. Nguyen, T.-N., Huynh, H.-H., & Meunier, J. (2015). Static hand gesture recognition using principal component analysis combined with artificial neural network. Journal of Automation and Control Engineering.
13. Oyedotun, O. K., & Khashman, A. (2017). Deep learning in vision-based static hand gesture recognition. Neural Computing and Applications.
14. Huang, D.-Y., Hu, W.-C., & Chang, S.-H. (2011). Gabor filter-based hand-pose angle estimation for hand gesture recognition under varying illumination. Expert Systems With Applications.
15. Trigueiros, P., Ribeiro, F., & Reis, L. P. (2014). Generic system for human computer gesture interaction. IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC).

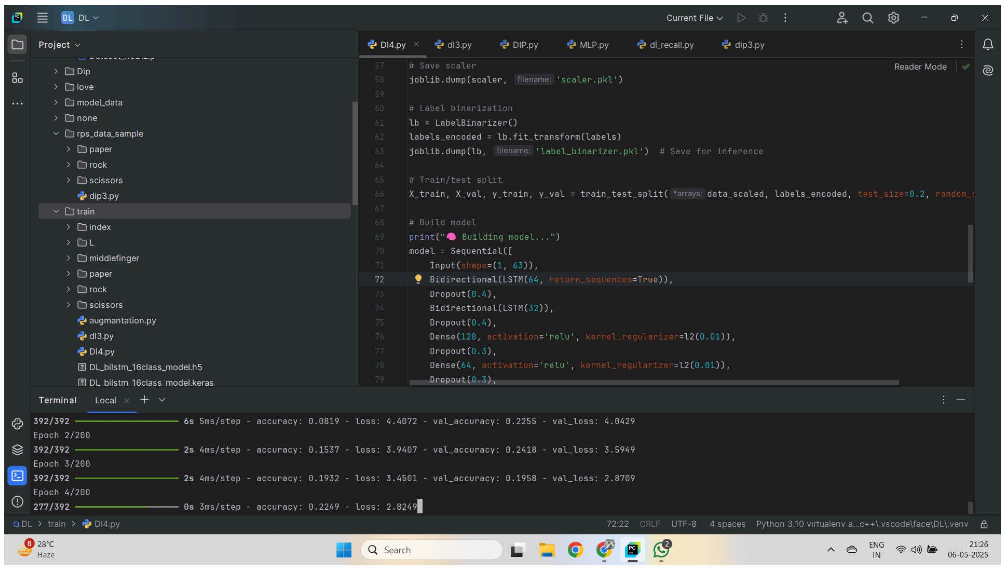
10. Appendix

BiLSTM



```python
# Save scaler
joblib.dump(scaler, filename: 'scaler.pkl')

# Label binarization
lb = LabelBinarizer()
labels_encoded = lb.fit_transform(labels)
joblib.dump(lb, filename: 'label_binarizer.pkl')  # Save for inference

# Train/test split
X_train, X_val, y_train, y_val = train_test_split( *arrays: data_scaled, labels_encoded, test_size=0.2, random_s

# Build model
print("🧠 Building model...")
model = Sequential([
    Input(shape=(1, 63)),
    Bidirectional(LSTM(64, return_sequences=True)),
    Dropout(0.4),
    Bidirectional(LSTM(32)),
    Dropout(0.4),
    Dense(128, activation='relu', kernel_regularizer=l2(0.01)),
    Dropout(0.3),
    Dense(64, activation='relu', kernel_regularizer=l2(0.01)),
    Dropout(0.3),
```

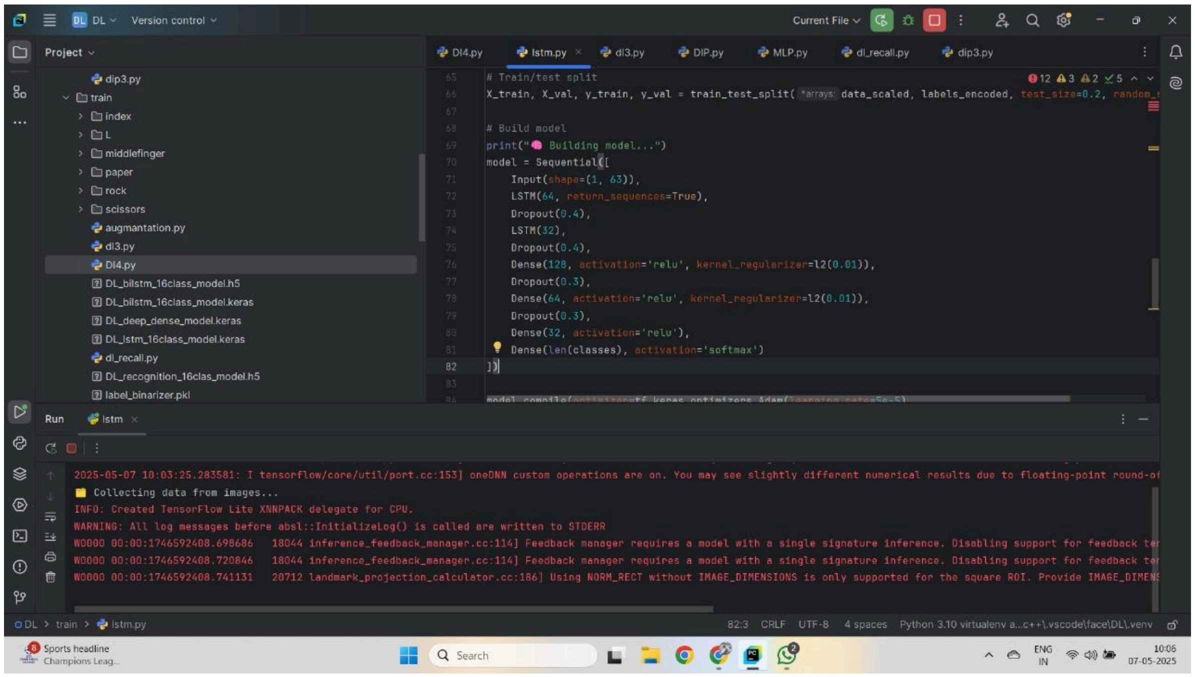
```
│ dense_3 (Dense)                  │ (None, 17)                │           561 │
└──────────────────────────────────┴───────────────────────────┴───────────────┘

 Total params: 125,969 (492.07 KB)
 Trainable params: 125,969 (492.07 KB)
 Non-trainable params: 0 (0.00 B)
Epoch 1/200
382/392 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.0815 - loss: 4.4107
```



```python
# Save scaler
joblib.dump(scaler, filename: 'scaler.pkl')

# Label binarization
lb = LabelBinarizer()
labels_encoded = lb.fit_transform(labels)
joblib.dump(lb, filename: 'label_binarizer.pkl')  # Save for inference

# Train/test split
X_train, X_val, y_train, y_val = train_test_split( *arrays: data_scaled, labels_encoded, test_size=0.2, random_

# Build model
print("🧠 Building model...")
model = Sequential([
    Input(shape=(1, 63)),
    Bidirectional(LSTM(64, return_sequences=True)),
    Dropout(0.4),
    Bidirectional(LSTM(32)),
    Dropout(0.4),
    Dense(128, activation='relu', kernel_regularizer=l2(0.01)),
    Dropout(0.3),
    Dense(64, activation='relu', kernel_regularizer=l2(0.01)),
    Dropout(0.3),
```

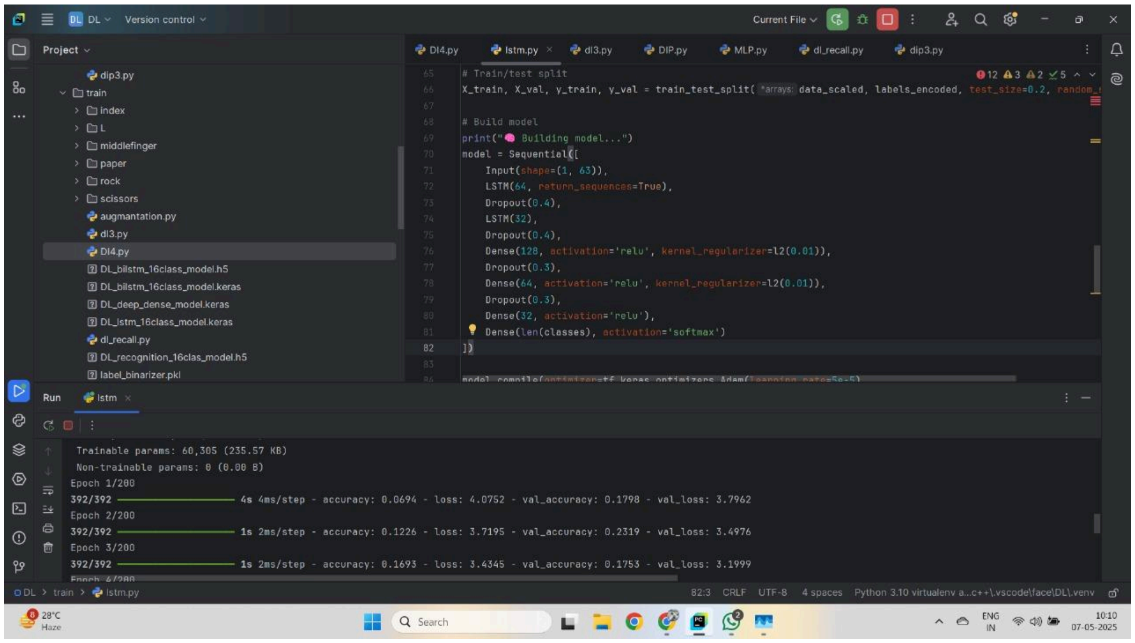
```
392/392 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - accuracy: 0.0819 - loss: 4.4072 - val_accuracy: 0.2255 - val_loss: 4.0429
Epoch 2/200
392/392 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - accuracy: 0.1537 - loss: 3.9407 - val_accuracy: 0.2418 - val_loss: 3.5949
Epoch 3/200
392/392 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - accuracy: 0.1932 - loss: 3.4501 - val_accuracy: 0.1958 - val_loss: 2.8709
Epoch 4/200
277/392 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.2249 - loss: 2.8249
```

LSTM



```python
# Train/test split
X_train, X_val, y_train, y_val = train_test_split( *arrays: data_scaled, labels_encoded, test_size=0.2, random_

# Build model
print("🧠 Building model...")
model = Sequential([
    Input(shape=(1, 63)),
    LSTM(64, return_sequences=True),
    Dropout(0.4),
    LSTM(32),
    Dropout(0.4),
    Dense(128, activation='relu', kernel_regularizer=l2(0.01)),
    Dropout(0.3),
    Dense(64, activation='relu', kernel_regularizer=l2(0.01)),
    Dropout(0.3),
    Dense(32, activation='relu'),
    Dense(len(classes), activation='softmax')
])
```

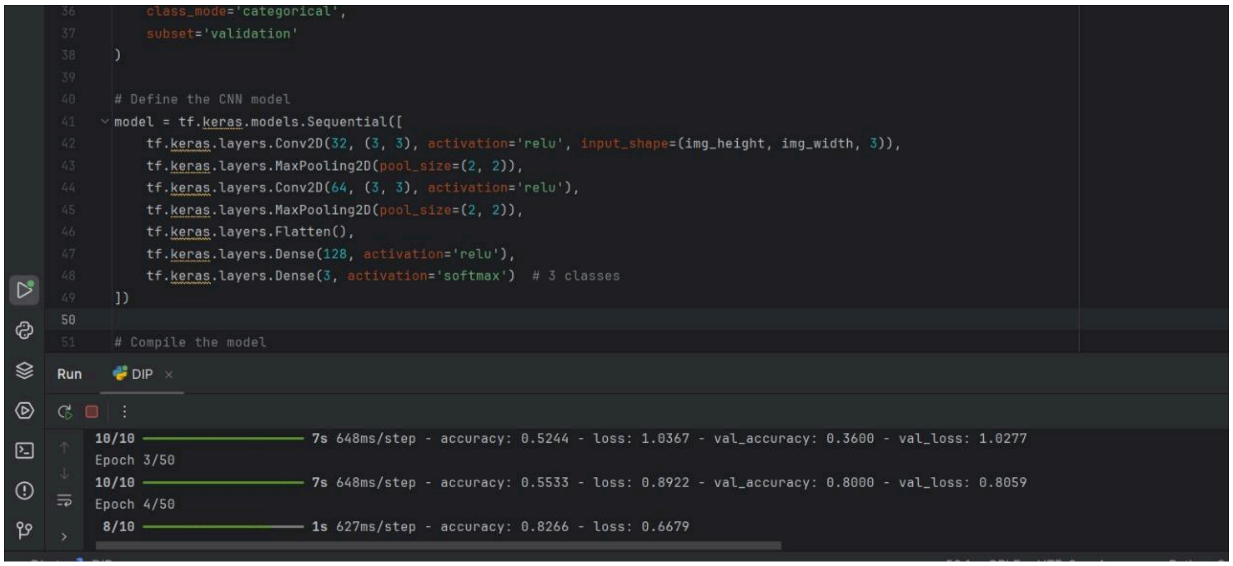
```
2025-05-07 10:03:25.283581: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-of
📁 Collecting data from images...
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
W0000 00:00:1746592408.698686   18044 inference_feedback_manager.cc:114] Feedback manager requires a model with a single signature inference. Disabling support for feedback ter
W0000 00:00:1746592408.720846   18044 inference_feedback_manager.cc:114] Feedback manager requires a model with a single signature inference. Disabling support for feedback ter
W0000 00:00:1746592408.741131   20712 landmark_projection_calculator.cc:186] Using NORM_RECT without IMAGE_DIMENSIONS is only supported for the square ROI. Provide IMAGE_DIMENS
```



```python
# Train/test split
X_train, X_val, y_train, y_val = train_test_split( *arrays: data_scaled, labels_encoded, test_size=0.2, random_

# Build model
print("🧠 Building model...")
model = Sequential([
    Input(shape=(1, 63)),
    LSTM(64, return_sequences=True),
    Dropout(0.4),
    LSTM(32),
    Dropout(0.4),
    Dense(128, activation='relu', kernel_regularizer=l2(0.01)),
    Dropout(0.3),
    Dense(64, activation='relu', kernel_regularizer=l2(0.01)),
    Dropout(0.3),
    Dense(32, activation='relu'),
    Dense(len(classes), activation='softmax')
])
```

```
 Trainable params: 60,305 (235.57 KB)
 Non-trainable params: 0 (0.00 B)
Epoch 1/200
392/392 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - accuracy: 0.0694 - loss: 4.0752 - val_accuracy: 0.1798 - val_loss: 3.7962
Epoch 2/200
392/392 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.1226 - loss: 3.7195 - val_accuracy: 0.2319 - val_loss: 3.4976
Epoch 3/200
392/392 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.1693 - loss: 3.4345 - val_accuracy: 0.1753 - val_loss: 3.1999
```

CNN



```
        class_mode='categorical',
        subset='validation'
    )

    # Define the CNN model
    model = tf.keras.models.Sequential([
        tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(img_height, img_width, 3)),
        tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
        tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
        tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(128, activation='relu'),
        tf.keras.layers.Dense(3, activation='softmax')  # 3 classes
    ])

    # Compile the model
```

```
10/10 ━━━━━━━━━━━━━━━━━━━━ 7s 648ms/step - accuracy: 0.5244 - loss: 1.0367 - val_accuracy: 0.3600 - val_loss: 1.0277
Epoch 3/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 7s 648ms/step - accuracy: 0.5533 - loss: 0.8922 - val_accuracy: 0.8000 - val_loss: 0.8059
Epoch 4/50
 8/10 ━━━━━━━━━━━━━━━━━━━━ 1s 627ms/step - accuracy: 0.8266 - loss: 0.6679
```

DL_Project_Report_Template (1).pdf

ORIGINALITY REPORT

10%	6%	7%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.coursehero.com Internet Source	2%
2	assets-eu.researchsquare.com Internet Source	1%
3	Muhammad Amir As'ari, Nur Anis Jasmin Sufri, Guat Si Qi. "Emergency sign language recognition from variant of convolutional neural network (CNN) and long short term memory (LSTM) models", International Journal of Advances in Intelligent Informatics, 2024 Publication	1%
4	Sozo Inoue, Guillaume Lopez, Tahera Hossain, Md Atiqur Rahman Ahad. "Activity, Behavior, and Healthcare Computing", CRC Press, 2025 Publication	1%
5	Submitted to Liverpool John Moores University Student Paper	<1%
6	Submitted to South Bank University Student Paper	<1%
7	Submitted to CSU Northridge Student Paper	<1%
8	Geoffrey E. Hinton, Simon Osindero, Yee-Whye Teh. "A Fast Learning Algorithm for Deep Belief Nets", Neural Computation, 2006 Publication	<1%
9	Submitted to UOW Malaysia KDU University College Sdn. Bhd Student Paper	<1%
10	www.frontiersin.org	