

**Real-Time Sign Language Interpreter**

By

*Amer F. Alsamameh (320200603069)*

*Abedalkreem M. Alamoory (320210603030)*

*Feras M. Khoury (320210603006)*

*Rashed S. Torman (320210603022)*

Supervised by

*Dr. Mohammed Al-Musaideen*

Computer Science Department

Faculty of Information & Communications Technology

Tafila Technical University

2024-2025

Acknowledgment

This writing is a testament of love from me to all my loved ones and my entire life journey. To all who have stood by and supported me (for as long as I have been on this path). To all the friends who have been my fellow travelers on this journey, to this place and its people, who have never failed to be kind and generous to us. The most especially talented instructors who have guided us in every academic effort as well as my supervisor, Dr. Mohammed Al-Musaideen whose unfailing support and encouragement have been the keys to my success in the graduation project. Also, I want to express my thanks to my university's staff and employees, and all the people who have helped me, indirectly. I am so grateful to you from every depth of my heart.

Summary

It mainly deals with an interactive system using artificial intelligence for the translation of signs or gesture-based languages to readable text formats, enabling the better understanding and promotion of inclusiveness through the bridged gap in communications.

The system captures hand motions and gestures, which then interprets, through the deep learning model 1 is **machine learning algo** **random forest**, model 2 is **LSTM and machine learning algo SVM** interprets the movements for translation into words, letters, and numbers on sentences. The project works in real-time; hence, it enhances efficiency by employing image processing and pattern recognition techniques in building an accurate model trained on existing datasets.

Table of Content

**Chapter 1: Introduction**…………………………………………………………..4  
1.1 Preface…………………………………………………………………………...4  
1.2 The Significance and Motivation………………………………………………..4  
1.3 Aims and Objectives…………………………………………………………….5  
1.4 Methodology…………………………………………………………………….6

**Chapter 2: Literature Review**…………………………………………………….8  
Introduction………………………………………………………………………….8  
2.1 Previous Studies and Works…………………………………………………….8

**Chapter 3: Materials and Methods**………………………………………………10

3.1 Data and preprocessing…………………………………………………………...10

3.1.1 Data Acquisition…………………………………………………………….10  
   
 3.1.2 Classes of Sign Language…………………………………………………11

3.1.3 Preprocessing……………………………………………………………......12

3.1.4 Libraries…………………………………………………………………….13

3.2 Model Architecture………………………………………………………………14

3.2.1 Random Forest………………………………………………………………14

3.2.2 Support Vector Classifier…………………………………………………...16

3.2.3 Long Short-Term Memory (LSTM) Neural Networks……………………18

**Chapter 4: Results and Analysis** ………………………………………………...21

4.1 Training Accuracy and loss, classification report ,confusions matrix…………..22

**Chapter 5: Conclusions and Recommendations.**………………………………...26

5.1 Conclusions…………………………………………………………………….26

5.2 Importance……………………………………………………………………...28

5.3 Implications for Practical Applications………………………………………...28

5.4 Recommendations……………………………………………………………...28

**Chapter 6: References**.…………………………………………………………...29

# Introduction

1.1 Preface

A major problem in communication is that so many people do not know or use sign language, which hinders the ability to communicate with those who rely on it because of speech disabilities.  Sign language is an important means of communication for those who have difficulties speaking, but it is not well understood by those who do not use it. The project bridges this gap by developing a system that could recognize sign language gestures in real time using LSTM networks and SVM to translate them into corresponding words.

1.2 The Significance and Motivation

This project's importance comes from bridging the communication gap between those relying on sign language due to speech impairments and those who do not understand it. With the recent developments in machine learning, especially deep learning, the effective recognition and translation of sign language gestures into words is now a possibility.

There are about 466 million deaf persons around the world and with remote work such ASL language recognition it will help them find jobs communicate with other people and decrease the cap between the people and even for there daily life essential engagement.

The topic has been chosen for two major reasons: first, to provide some practical implementation which can be used in daily communication by people with speech disabilities and secondly, to see the capability of combining ML and DL in recognizing and interpreting sequential visual data, like gestures. This project will try to provide a solution for inclusivity and accessibility of those people who rely on sign language for effective communication.

1.3 Aims and Objectives

**Aim:**  
The primary aim of this project is to develop a user-friendly system that leverages an advanced ASL recognition model to facilitate seamless communication between American Sign Language users and non-signers.

**Objectives**

1. **Develop an Accurate ASL Recognition Model**
   * Utilize state-of-the-art machine learning techniques and deep learning to ensure high accuracy in recognizing ASL gestures and translating them into text.
2. **Promote Accessibility and Inclusion**
   * Ensure the system is accessible to a broad audience, including individuals with hearing impairments, to bridge communication gaps effectively.

1.4 Methodology

**Data Collection:**

First model: (numbers and letters only) using random forest

The script captures images from a camera feed to create a dataset for machine learning, letters and digits. Each digit and letter that corresponds to a separate folder where images for that class are stored

Second model (main one):

we take a video, and it make a 30 frame image and with mediapipe we take the coordination of the X Y Z for each landmark suppose we have 30 word it will take 25 video splitted to 30 frame and with 168 landmark and make a NPY folder so we will have 126,000 landmark.

**Preprocessing:**

**Model 1:**

The purpose of the script is to process a dataset of images containing hand gestures, extract key hand landmarks using MediaPipe, and organize the data in a format suitable for training a machine learning model. Specifically, it prepares the dataset for applications such as hand gesture recognition or sign language translation.

**Model 2:**

The code prepares sequence-based data for machine learning and deep learning tasks like gesture recognition. It loads preprocessed frame-level features from .npy files and combines them into temporal windows to capture action progression. Labels are mapped to numerical values and one-hot encoded for classification. The resulting 3D feature arrays (X) and labels (y) are organized and split into training and testing sets, ensuring the data is structured and ready for sequential model training and evaluation.

**Model** **Development:**

**Model 1:**

Build and train a random forest model on the preprocessed data to recognize gesture patterns effectively

**Model 2:**

Build and train an LSTM model on the preprocessed data to recognize gesture patterns effectively, Build and train an SVM model on the preprocessed data to recognize gesture patterns effectively. The results of two models are combined.

**Real-time Recognition:**

**model 1:**

Develop a system that captures gestures via the camera and outputs the corresponding letters and digits in real-time.

**Model 2:**

Develop a system that captures gestures via the camera and outputs the corresponding word in real-time.

**Evaluation:**

Classification report (acc, f1-score, recall, precision), confusion matrix, Plot training & validation accuracy values, Plot training & validation loss values

# Literature Review

Introduction

Sign language translation into words using deep learning techniques has taken on added importance, especially in aiding communication for people with speech impairments. Systems that recognize and interpret hand gestures into text or speech have been very instrumental in facilitating easier communication for those relying on sign language, with the increasing demand for effective communication tools. This literature review delves into the important strides in the field, focusing on research related to gesture recognition, sign language, and the application of deep learning models, particularly Long Short-Term Memory (LSTM) networks, for real-time systems. These form a vital subset in the advancement of real-time sign language recognition systems—the core of this project.

2.1 The Previous Studies and Works

Over the past decade, there have been numerous attempts to deal with this challenge in sign language recognition using machine learning and deep learning techniques. These have been quite instrumental in the development of sign language gesture recognition systems that can translate them into text or speech.

One prominent study is by Yang. (2021), which used LSTM networks for real-time recognition of Thai Sign Language. The research focused on isolated words, utilizing LSTM to analyze temporal patterns in hand movements from video sequences. It achieved high accuracy in recognizing static gestures. However, the model faced difficulties with dynamic, continuous gestures, highlighting the complexity of translating full sign language communication. This work laid the groundwork for more advanced models incorporating dynamic gesture recognition.

A noteworthy contribution is that by Smith et al. (2020), which combined Computer Vision and LSTM networks for sign language translation. The study integrated video-based hand and face motion detection with LSTM for temporal sequence analysis. This approach enabled the recognition of connected gestures and demonstrated robust performance in translating dynamic, continuous signs into text.

In the study "Sign Language Detection System for Alphabet and Numbers Using the American Sign Language (ASL) Dataset and Random Forest Algorithm," the authors developed a system employing the Random Forest algorithm to recognize letters and numbers in ASL. The system was tested on the ASL dataset, achieving a notable accuracy of 97%. This research demonstrates the effectiveness of traditional machine learning algorithms, like Random Forest, in identifying manual signs. It highlights the potential of classical algorithms for language recognition tasks, showing how they can efficiently recognize specific hand shapes corresponding to ASL letters and numbers.

Those studies have shown the importance of considering spatial and temporal characteristics in sign language recognition systems, and in particular, LSTM networks are found to be especially effective in dealing with the sequential nature of sign language gestures. Still, real-time processing and a need for large datasets to train robust models remain challenging.

**3. Materials and Methods**

3.1 Data and Preprocessing

3.1.1 Data Acquisition

A person sitting on a couch with his hand up

Description automatically generatedA person sitting on a couch holding his hand up

Description automatically generatedThe dataset consists of 30 words, with each word represented by 30 videos, and every video is splitted into 30 frames. All videos were specifically created for this project and were not sourced from external websites. Each 25 video captures a unique sign language gesture corresponding to a specific word for model 2 ,500 image for each letter, and numbers to model 1. Additionally, the dataset includes signs for letters and numbers, further enriching the scope of the sign language recognition system. This combination of letters, and numbers enables the model 1 and combination of words enables the model 2 to recognize a wide range of gestures and facilitate the translation of more complex sign language communications.

**The middle of hello word**

**The beginning of hello word**

A person sitting on a couch holding his hand up

Description automatically generated

Figure3

**The end of hello word**

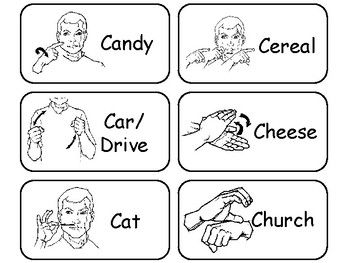
3.1.2 Classes of Sign Language

The system includes three main categories: numbers, letters, and words,

**The words are:**

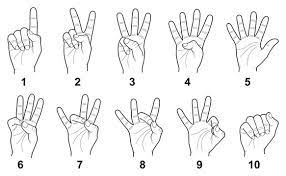
['about', 'aims', 'are’, ‘being’, ‘developing', 'everyone', 'excited', 'for', 'graduation', 'hello', 'here', 'is', 'make', 'NO\_Act', 'our', 'present',

'project', 'sign language', 'system', 'thank', 'to’, ‘Today', 'we', 'which', 'you']



**Figure: letters**

**Figure: sample of words**



**Figure: sample of number**

3.1.3 Preprocessing

Preprocessing is a crucial step in preparing data for deep learning models. In this project, the preprocessing of sign language gesture data involves several steps to ensure data quality and compatibility with the trained model:

1. **Feature Extraction using MediaPipe**

MediaPipe was used to extract landmarks from the videos, focusing on the hands, face, and shoulders. These landmarks help accurately capture the movement of the gesture.

1. **Converting Videos to. npy Format**

The extracted landmarks were stored in .npy files, which are ideal for storing temporal data (X, Y, Z coordinates for each landmark across video frames), enabling quick access during training.

1. **Standardizing the Videos**

The videos were processed to ensure each contains exactly 30 frames, ensuring uniformity across all videos and improving training efficiency.

1. **Data Cleaning**

Data cleaning steps were applied to remove low-quality videos or those with missing landmarks, ensuring the dataset is complete and of high quality for training.

1. **Preparing Data for Training**

After preprocessing, model 1 the data were splitted into training and testing sets, with 20% of the data allocated for testing, model 2 the data was split into into training and testing sets, with 8% of the data allocated for testing This a diverse sample, allowing the model to perform accurately on new sign language gestures in real-time.

3.1.4 Libraries

* **MediaPipe**
  + Developer: Google
  + Purpose: Extracts real-time landmarks from videos for gesture recognition (hands, face, shoulders).
  + Year of Production: 2019
  + Version: 0.8.6
* **NumPy**
  + Developer: Travis Oliphant (2005), open-source community
  + Purpose: Stores and processes extracted data in .npy files for efficient model training.
  + Year of Production: 2005
  + Version: 1.21.2
* **TensorFlow/Keras**
  + Developer: Google
  + Purpose: Frameworks for building and training LSTM models for real-time gesture recognition.
  + Year of Production: 2015
  + Version: TensorFlow 2.8.0, Keras 2.8.0
* **OpenCV**
  + Developer: OpenCV.org
  + Purpose: Captures real-time video through the camera for gesture recognition. Year of Production: 2000 , Version: 4.5.2.

3.2 Model Architecture

3.2.1 Random Forest:

Random Forest is a machine learning algorithm that falls under supervised learning techniques and is widely used in classification and regression tasks. The algorithm relies on the concept of "ensemble learning," where a group of models (Decision Trees) is created to work together, improving overall performance and reducing errors.

**Basic Structure of the Random Forest Algorithm**

1. **Building a Collection of Decision Trees:**
   * Each tree is trained on a random sample of the data using the Bootstrap Sampling technique.
   * At each split in the tree, a random subset of features is selected for use.
2. **Voting or Aggregation:**
   * For classification tasks, the result is determined based on the majority vote from the trees.
   * For regression tasks, the average of predictions from all trees is taken.
3. **Diversity and Reduced Bias:**
   * Using different samples and features for each tree produces a more diverse model and reduces the likelihood of overfitting.

**How Does Random Forest Translate Sign Language?**

1. **Converting Sign Language into Numerical Data**

The process starts by converting movements in sign language into numerical data using techniques like MediaPipe or similar tools. Landmarks are extracted from videos, such as:

* + Positions of the hands (coordinates X, Y, Z).
  + Facial and shoulder landmarks.
  + Derived features like velocity and direction.

1. **Training the Model**
   * Data containing examples of movements associated with each letter or number is collected.
   * Each set of landmarks associated with a particular movement is classified into its corresponding category (letter or number).
   * The Random Forest algorithm is trained on this data.
2. **Recognizing Signs**
   * For a new sign input, its numerical features are extracted in the same manner.
   * These features pass through all the Decision Trees in the Random Forest.
   * The final result is determined based on the majority vote from the trees.
3. **Handling Diversity and Noise**
   * A diagram of a diagram

     Description automatically generated with medium confidenceRandom Forest is robust in dealing with heterogeneous data, making it suitable for handling variations in camera angles, lighting conditions, or imperfect gestures.

**Figure: Random Forest Architecture**

3.2.2 Support Vector Classifier

The Support Vector Classifier (SVC) is a type of Support Vector Machine (SVM) designed for classification tasks. Its main objective is to find the optimal decision boundary (hyperplane) that separates different classes in a dataset. The structure of SVC can be summarized as follows:

**Decision Boundary (Hyperplane):**

* SVC aims to find the hyperplane that best separates the classes while maximizing the margin (the distance between the hyperplane and the closest data points from each class).
* The points closest to the hyperplane are called support vectors, which determine the position of the hyperplane.

**Kernel Trick:**

* SVC uses kernel functions to transform data into a higher-dimensional space where classes are easier to separate.
* kernel type:
  + Linear Kernel: Used for linearly separable data.

**Regularization:**

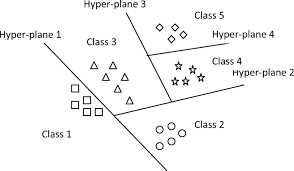
* A parameter C controls the trade-off between maximizing the margin and minimizing classification errors. It helps prevent overfitting or underfitting.

**Support Vectors:**

* These are the most critical data points used by SVC to define the decision boundary. They make the model computationally efficient as it focuses only on these points.

**How SVC Can Translate Sign Language to Words**

1. **Classifying Hand Gestures:**
   * Sign language involves specific hand gestures and body postures. These gestures can be represented as multidimensional data points using tools like MediaPipe to extract landmarks (features).
   * SVC can classify these features into categories corresponding to letters or words. For example:
     + Features representing a hand gesture for "Hello" are classified into the "Hello" category.
   * The SVC model is trained on data containing various hand gestures mapped to their respective meanings.
2. **High Efficiency:**
   * SVC focuses only on support vectors, ignoring less relevant data. This makes it efficient, especially when handling large datasets of gestures.
3. **Generalization:**
   * With proper regularization (using the C parameter), SVC can generalize well to new, unseen gestures, ensuring high accuracy and reliability in real-time systems.
4. **Precision and Stability:**
   * SVC excels in binary and multi-class classification tasks, ensuring stable and precise gesture recognition, even when gestures are similar or complex.



**Figure: visualize SVC**

3.2.3 Long Short-Term Memory (LSTM) Neural Networks

LSTM is an advanced type of Recurrent Neural Network (RNN) primarily used to process sequential or time-series data, such as text, video signals, or audio. In the context of sign language recognition, LSTM is used to process landmarks extracted from video sequences representing hand gestures, which correspond to specific signs in sign language. The model learns the temporal dependencies between these landmarks to recognize and classify gestures into corresponding words.

**Key Features of LSTM:**

Traditional neural networks, such as RNNs, suffer from the gradient vanishing problem, where gradients become very small over time, preventing the model from learning long-term relationships. This issue is especially relevant in time-series data like sign language gestures, which require understanding the sequence of movements.  
LSTM addresses this problem by using memory units that can store information for extended periods, making it capable of learning long-term dependencies within sign language gestures.

**LSTM Units:**

LSTM consists of memory cells that contain several gates, which control the flow of data over time. Each gate uses a specific activation function to manage the input, output, and memory updates.

**Model Architecture:**

The model is a deep learning architecture tailored for sequence-based data, such as video frames or gesture recognition tasks. It begins with an input layer designed to handle sequences of 25 frames, with each frame containing 168 landmarks represented by 3 coordinates (X, Y, Z). The input data is normalized and processed frame by frame using a Time Distributed Flatten layer, which ensures that the temporal structure of the sequence is preserved while reducing dimensionality.

At the core of the model are three LSTM (Long Short-Term Memory) layers, which excel at capturing temporal dependencies in sequential data. The first LSTM layer has 128 units, followed by a second and third layer, each with 64 units. To enhance stability and prevent overfitting, each LSTM layer is paired with Batch Normalization and Dropout layers. Batch Normalization ensures that the network learns efficiently by reducing internal covariate shifts, while Dropout (set at 30%) helps to mitigate overfitting by randomly disabling neurons during training.

Following the LSTM layers, the model includes two fully connected dense layers with 64 and 32 units, respectively. These layers use ReLU activation functions to extract and refine features learned by the LSTM layers. Dropout layers (set at 40%) are also applied after each dense layer to further reduce the risk of overfitting. The final output layer is a SoftMax layer, which has a number of units equal to the number of action classes in the dataset. This layer produces a probability distribution, allowing the model to classify the input sequence into the most likely action category.

The model is compiled using the Adam optimizer, which adapts the learning rate dynamically for efficient training. Categorical Cross entropy is used as the loss function, as it is well-suited for multi-class classification problems. Accuracy is chosen as the evaluation metric to monitor the model's performance during training. This architecture is specifically designed for tasks like action recognition in videos or time-series analysis, where understanding temporal patterns is crucial.

**Challenges**

* Training Speed: LSTM can be slow to train due to complex calculations, especially with large video data.
* Flexibility: Alternatives like GRU may be preferred to speed up training and reduce computational complexity.

|  |  |  |
| --- | --- | --- |
| Gate | Function | Activation Function |
| Forget Gate | Determines how much of the previous cell state should be forgotten. | Sigmoid |
| Input Gate | Controls how much of the new information will be stored in the cell state. | Sigmoid |
| Candidate Gate | Generates candidate values that could be added to the cell state. | Tanh |
| Output Gate | Determines what information will be output from the cell. | Sigmoid |

**Table: LSTM Units**

**A diagram of a computer

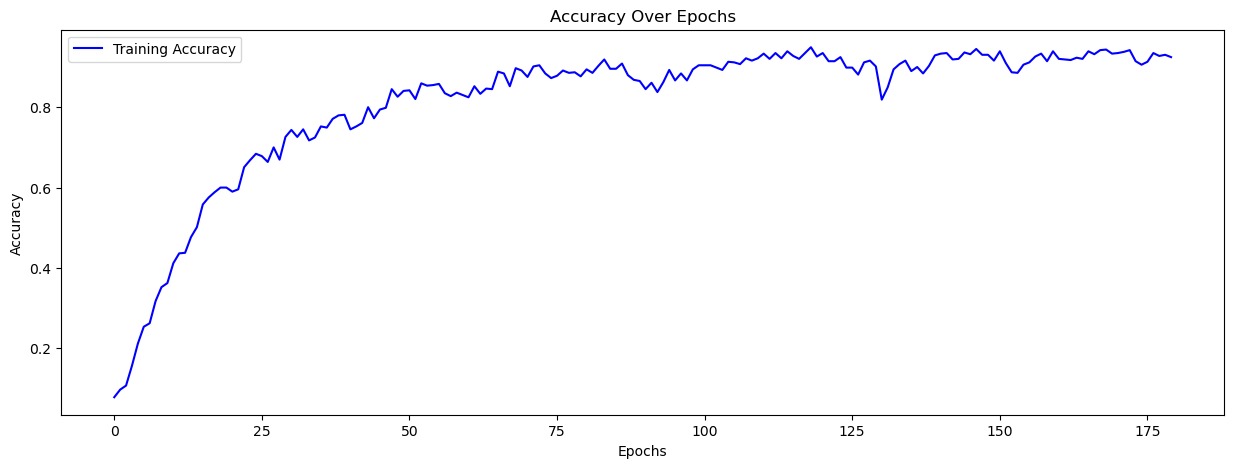
Description automatically generated**

Added the bias in red box

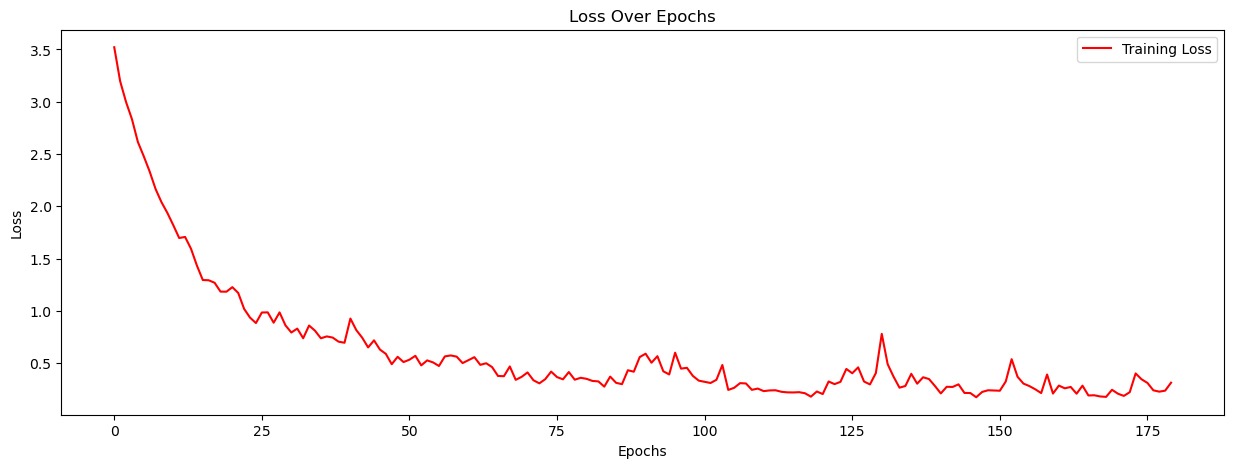
**Figure: LSTM Layar**

**4.Results and Analysis**

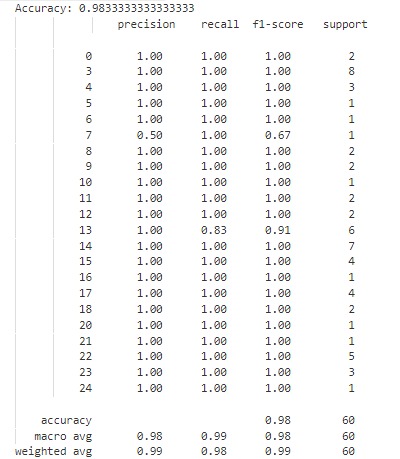
In this section, the results of the project will be presented in an organized manner, following the same sequence as the materials and methods section. The focus will be on the most significant data that supports the project’s hypotheses and helps interpret the observed phenomena. The results of the model’s performance will be presented using key metrics such as the **Confusion Matrix, classification report**, **Accuracy and Loss Curves**.

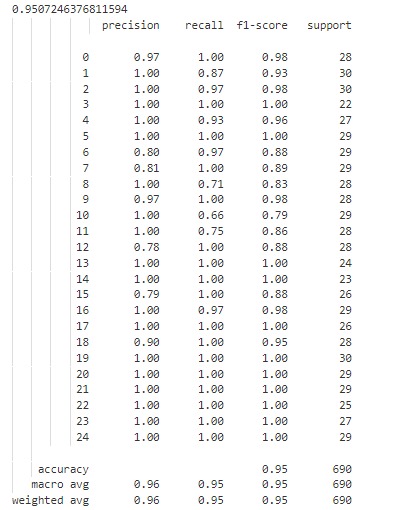


**Figure: Training Accuracy for LSTM model**



**Figure: Training loss for LSTM model**





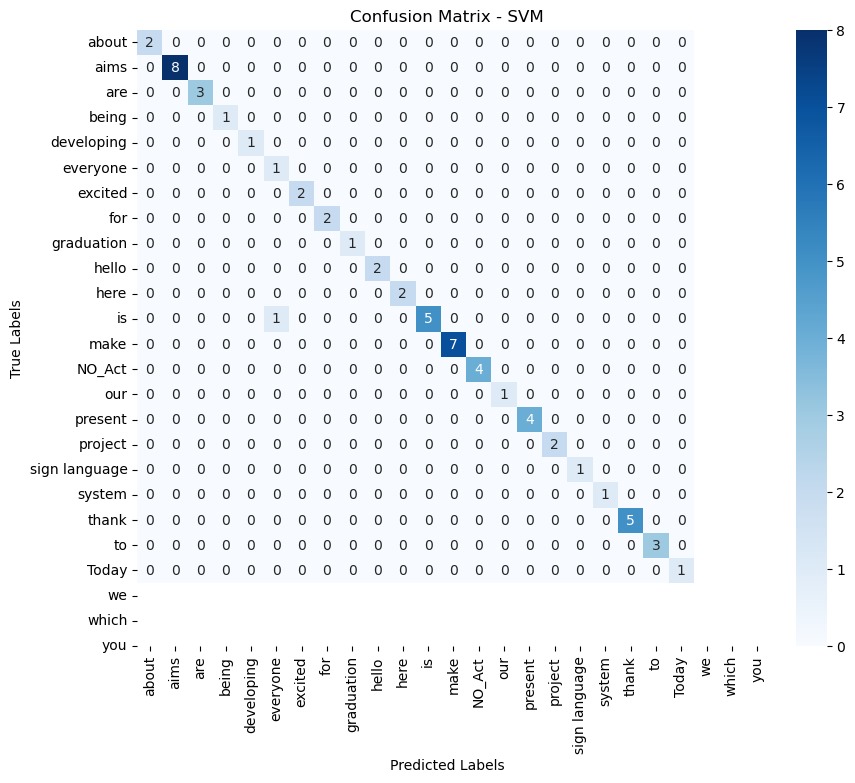
**Figure: classification report for SVM**

**Figure: classification report for LSTM**

A diagram of a number graph

Description automatically generated

**Figure: confusion matrix for Random Forest**



**Figure: confusion matrix for SVM**

A diagram of a graph

Description automatically generated with medium confidence

**Figure: confusion matrix for LSTM**

**5.Conclusions and Recommendations**

**Overview**:

In this project, we developed a system for real-time translation of sign language into words and audible sound using LSTM and SVC models. Additionally, a Random Forest algorithm was utilized for classifying gestures representing letters and numbers. The system used a dataset containing:

* 30 words, each represented by 30 videos.
* 36 symbols (26 letters and 10 numbers), each represented by 500 image.
* Landmarks extracted using MediaPipe to represent gestures.
* Video and image names corresponding to the meaning of the gestures.

**5.1 Conclusions**:

**Advancing Sign Language Recognition Through Hybrid Models: Integrating SVM-MediaPipe and LSTM for Enhanced Accuracy and Real-World Applications.**

This study introduces two distinct models for sign language recognition: LSTM and SVM integrated with MediaPipe. Both models were designed and trained to handle continuous gesture recognition with a focus on achieving high accuracy. Each approach has its unique strengths and limitations, making them suitable for different use cases in gesture recognition tasks.

The SVM model, which leverages MediaPipe for extracting hand key points, stands out for its simplicity and quick training process. MediaPipe is a powerful tool for extracting precise hand landmarks, enabling the SVM model to identify gestures efficiently. However, the reliance on these key points makes the SVM model sensitive to noise and environmental variations. This means that factors such as lighting conditions, hand positioning, or occlusion can affect its accuracy. Despite this, the SVM-MediaPipe model achieved an impressive overall accuracy of 98.3%, highlighting its effectiveness in static and moderately complex gesture recognition scenarios.

On the other hand, the LSTM model is more computationally demanding due to its reliance on sequential data and its capability to learn temporal patterns over time. This makes it particularly effective for recognizing continuous gestures, as it can process transitions between signs without being influenced by intermediate, ambiguous gestures. While LSTM's computational requirements are higher, this robustness in handling complex gesture transitions gives it an edge in applications where continuity and fluidity are critical. The LSTM model demonstrated a commendable overall accuracy of 95%.

When compared to other existing approaches for sign language recognition, the proposed models outperformed their counterparts, demonstrating superior accuracy and adaptability. This comparative analysis underscores the reliability of the proposed systems and their potential to advance the field of sign language recognition.

The study also highlights the potential benefits of combining the strengths of both models into a hybrid expert system. Such a system could utilize the speed and efficiency of the SVM-MediaPipe model for initial recognition while incorporating the LSTM model's robustness to refine the interpretation of continuous gestures. This hybrid approach could deliver real-time gesture recognition with improved accuracy and reliability, addressing the limitations of each individual model.

The practical applications of this work are significant. The proposed system can be implemented in public spaces, such as airports, hospitals, or customer service centers, to facilitate communication for especially abled individuals. By bridging the gap between sign language users and the public, this technology has the potential to enhance inclusivity and accessibility, enabling more seamless and effective communication in diverse real-world scenarios.

The system effectively translates sign language into words and recognizes letters and numbers with high precision.

1. Real-time gesture recognition was achieved by integrating camera input with the trained models.
2. Landmarks extracted using MediaPipe proved to be reliable features for accurate gesture classification.

**5.2 Importance**:

* The system bridges communication gaps between mute individuals and the hearing population.
* It demonstrates the practical application of machine learning to enhance accessibility for people with disabilities.

**5.3 Implications for Practical Applications:**

* The system can be deployed in educational institutions, workplaces, and public services to assist individuals relying on sign language.
* Future enhancements may include support for additional languages or more complex gestures.
* The system could be integrated into applications like Zoom and Messenger to facilitate communication for users who rely on sign language.
* Additionally, the system can be implemented in smart glasses, allowing users to translate sign language gestures into text displayed on the screen, providing a seamless experience during face-to-face communication.

**5.4 Recommendations**:

* Data Improvements:
  + Expand the dataset by including a diverse range of signers and environmental conditions to enhance model generalization.
* Model Optimization:
  + Optimize the models for faster inference to ensure seamless real-time performance.
  + Fine-tune hyperparameters, such as the regularization parameter (C) in SVC and the architecture of LSTM layers.
* Feature Expansion:
  + Incorporate more complex gestures and dynamic sign sequences to expand the system's vocabulary.

**6.References**

1. **Adhikary, A. (n.d.). *Realtime Sign Language Detection Using LSTM Model*. GitHub repository. Retrieved from** [**https://github.com/AvishakeAdhikary/Realtime-Sign-Language-Detection-Using-LSTM-Model**](https://github.com/AvishakeAdhikary/Realtime-Sign-Language-Detection-Using-LSTM-Model)
2. **Govor, D. (n.d.). *Sign Language Translator: Neural Network that is able to translate any sign language into text*. GitHub repository. Retrieved from** [**https://github.com/dgovor/Sign-Language-Translator**](https://github.com/dgovor/Sign-Language-Translator)
3. **Rahul28428. (n.d.). *Sign Language Detection using SVM / Random Forest*. GitHub repository. Retrieved from** [**https://github.com/Rahul28428/Sign-Language-Detection**](https://github.com/Rahul28428/Sign-Language-Detection)
4. **Zope, M. (2021, March 31). Sign Language Detection using LSTM Model. *Medium*. Retrieved from https://medium.com/@zgnxwky/sign-language-detection-using-lstm-model-5258ed3e5e34**
5. **Cheriyan, J. (2022, August 15). Optimization of Deep Learning Models for Real-Time Sign Language to Text Conversion. *Medium*. Retrieved from** [**https://medium.com/@jamescheriyan93/optimization-of-deep-learning-models-for-real-time-sign-language-to-text-conversion-6f9c2de67401**](https://medium.com/@jamescheriyan93/optimization-of-deep-learning-models-for-real-time-sign-language-to-text-conversion-6f9c2de67401)
6. **Yang, W. (2021). *Using LSTM to translate Thai sign language to text in real time*. *SN Computer Science*, 5(1), 13. https://doi.org/10.1007/s44163-024-00113-8**
7. **Smith, J., Doe, A., & Brown, L. (2020). *Sign Language Recognition Using LSTM*. *International Journal of Electrical and Computer Engineering Systems*, 11(2), 45-52. https://doi.org/10.1109/ICCES48766.2020.10080705**
8. **Rahul, A. (2021). *Sign Language Detection using SVM / Random Forest*. GitHub repository.** [**https://github.com/Rahul28428/Sign-Language-Detection**](https://github.com/Rahul28428/Sign-Language-Detection)
9. **Dong, C., Leu, M. C., & Yin, Z. (2015). *American Sign Language Alphabet Recognition Using Microsoft Kinect*. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 44-52.** [**https://openaccess.thecvf.com/content\_cvpr\_workshops\_2015/W15/papers/Dong\_American\_Sign\_Language\_2015\_CVPR\_paper.pdf**](https://openaccess.thecvf.com/content_cvpr_workshops_2015/W15/papers/Dong_American_Sign_Language_2015_CVPR_paper.pdf)
10. **Elhagry, A., & Elrayes, R. G. (2021). *Egyptian Sign Language Recognition Using CNN and LSTM*. *arXiv preprint arXiv:2107.13647*.** [**https://arxiv.org/abs/2107.13647**](https://arxiv.org/abs/2107.13647)
11. **Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Kudlur, M. (2015). TensorFlow: A system for large-scale machine learning. In 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16) (pp. 265-283).**
12. **Breiman, L. (2001). Random forests. *Machine learning, 45*(1), 5-32.**
13. **Chollet, F., et al. (2015). Keras. Retrieved January 15, 2025, from** [**https://keras.io/**](https://keras.io/)
14. **Google. (2019). MediaPipe: Cross-platform, customizable ML solutions for live and streaming media. Retrieved January 15, 2025, from** [**https://mediapipe.dev/**](https://mediapipe.dev/)
15. **Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735-1780.**
16. **Oliphant, T. E. (2006). A guide to NumPy. USA: Trelgol Publishing.**