



**UNIVERSITY OF TURKISH AERONAUTICAL
ASSOCIATION**

ENGINEERING FACULTY

DEPARTMENT OF COMPUTER ENGINEERING

CENG 496 Senior Design Project II

Project Members:

Asiye İlayda AŞAR

190444023 Şevval ÖZCANOĞLU

190444022

Project Advisor:

ASSIST. PROF. DR. SHADI AL SHEHABI

Table of Contents

1.Introduction.	4
1.1 Purpose of the system.	4
1.2 Scope of the system.	4
1.3 Objectives and success criteria of the project.	5
1.4 Definitions, acronyms, and abbreviations.	5
1.5 References.	6
1.6 Overview.....	8
2. Current system	8
3.Proposed system	9
3.1 Functional requirements.	9
3.2 Nonfunctional requirements	9
3.2.1. Usability	...10
3.2.2. Reliability	10
3.2.3. Performance.	10
3.2.4. Supportability	11
3.2.5. Implementation.	11
3.2.6. Interface.	12
3.2.7. Packaging.	12
3.2.8. Legal.	12
3.3 System models	13
3.3.1 Scenarios.	13
3.3.2 Use case model.	17
3.3.3 Analysis object model.	18
3.3.4 Dynamic model.	19

4. Glossary	24
5. Application	25

1. Introduction

1.1. Purpose of The System

According to the World Health Organization (WHO), there are approximately 466 million deaf individuals worldwide, and this number is expected to reach 700 million by the year 2050, constituting approximately 10% of the world's population. A study conducted in Turkey revealed that 4.1% of the population uses hearing aids, while 0.1% has severe hearing loss that cannot be aided by such devices.

Sign languages are visual languages used by deaf or hard of hearing individuals to communicate through hand movements, facial expressions, and body postures. However, the majority of hearing individuals do not understand sign language, creating a significant communication barrier for deaf individuals. To address this gap, there is a need for computer vision-based methods that can automatically recognize and interpret sign language.

In this context, the aim of this study is to develop a system capable of recognizing and translating Turkish Sign Language in real-time. Various methods and techniques have been employed to develop a model for recognizing Turkish Sign Language. This study aims to make significant contributions to the field of sign language recognition both technically and practically

1.2. Scope of The System

Automatic sign recognition is a field aimed at facilitating the lives of hearing-impaired individuals by automatically recognizing sign language from video data. The scope of a model that recognizes and interprets Turkish sign language is quite broad, primarily aiming to enhance communication accessibility for the deaf community in Turkey. This model provides a tool for effectively communicating Turkish sign language gestures to those who do not know sign language. In this context, various techniques and methods have been employed to develop and optimize the model.

Additionally, a web-based platform has been integrated into the application. This platform includes features such as a Turkish sign language learning module, word translation, and real-time translation. These features aim to empower users in learning and using sign language, thereby encouraging more effective communication with hearing-impaired individuals.

1.3. Objectives and Success Criteria of the Project

The success criteria of the project involve achieving target values in the training, testing, and validation processes. Various training strategies and techniques are employed to increase the model's accuracy during the training process. For example, different deep learning algorithms and architectures can be experimented with, or hyperparameters such as the learning rate can be adjusted. One of the success criteria is to reach a certain level of accuracy on the training data.

Comparing different models and methods is also crucial for the success of the project. This aims to identify the model that provides the best solution to a specific problem and improve performance. These comparisons are made to understand the impact of different algorithms, hyperparameters, and features.

In addition, having features such as real-time translation, translation from text to video, and a learning module for users to learn is crucial in the web-based application. The real-time translation feature enables users to communicate instantly, increasing communication accessibility. The text-to-video translation feature allows deaf individuals to communicate with those who do not know sign language. The learning module helps users learn Turkish Sign Language and practice, enabling them to use sign language more effectively. These features enhance the functionality and user experience of the application, supporting the success of the project.

1.4. Definitions, Acronyms, and Abbreviation

- TSLRM: Turkish Sign Language Recognition Model
- TSLT: Turkish Sign Language Translator
- SLR (Sign Language Recognition): Process of analyzing video/images, recognizing gestures, and interpreting them as TSL signs.
- ML (Machine Learning): Enabling computers to learn without explicit programming.
- DL (Deep Learning): Subset of ML utilizing artificial neural networks for large data processing.
- UI (User Interface): Graphical elements users interact with in the app.
- Backend: Server-side data processing and logic handling for the app.
- Cloud Computing: On-demand computing services (servers, storage, etc.) delivered via the internet.
- CNN (Convolutional Neural Network): Deep learning architecture for image recognition and classification, ideal for TSL gesture recognition.
- RNN (Recurrent Neural Network): Processes sequential data like TSL gestures.
- LSTM (Long Short-Term Memory): A type of RNN designed for long-term dependency learning in TSL sequences.
- Inception I3D is a deep learning model with a 3D architecture. This model is specifically designed for the classification of video data. By expanding the filters in

2D convolution layers and pooling layers to three dimensions (time, height, and width), 3D can better represent the spatial and temporal features of videos.

- Pose Estimation: Identifying the position and orientation of body parts .
- Gesture Segmentation: Identifying the start and end points of individual signs
- OpenCV: Popular library for image processing and feature extraction, useful for video pre-processing in SLR.
- MediaPipe is an open-source framework used for image and video processing. It is specifically designed for creating real-time computer vision and interactive experiences.
- PyTorch: is an open-source machine learning library used to develop deep learning and artificial intelligence applications. PyTorch stands out for its flexibility, speed, and ease of use, making it widely preferred among researchers and industrial users.
- HTML5: The latest version of the HyperText Markup Language, used for structuring and presenting content on the web.
- CSS (Cascading Style Sheets): A style sheet language used for describing the presentation of a document written in HTML or XML.
- Flask: A lightweight WSGI web application framework in Python, known for its simplicity and flexibility.
- AUTSL: The Ankara University Turkish Sign Language dataset, consisting of video sequences representing different Turkish Sign Language words.
- API (Application Programming Interface): Set of methods for building software applications.
- SDK (Software Development Kit): Tools and libraries provided by a platform to facilitate development.
- UI/UX (User Interface/User Experience): Design and usability aspects of the app.
- TTS (Text-to-Sign): Converting written text into sign language .
- STT (Sign-to-Text): Converting sign language into written text (potentially for sign language recognition input).

1.5. References

- <https://www.freepik.com/>
<https://mockflow.com/blog/M2SfIXX5Kh-How-to-Create-a-Wireframe-A-detailed-Step-by-Step-Guide>
<https://www.justinmind.com/blog/best-tool-to-develop-screen-mockups/>
<https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/>
arXiv:2004.01283 [cs.CV]
- Sincan, O. M., Keles, H. Y. “AUTSL: A Large Scale Multi-modal Turkish Sign Language Dataset and Baseline Methods”. IEEE Access, vol. 8, pp. 181340-181355, 2020.
- Viola P. and Jones M., Rapid object detection using a boosted cascade of simple features, in Computer Vision and Pattern Recognition, CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on, vol. 1. IEEE, 2001.
- Hochreiter S. and Schmidhuber J., Long short-term memory, Neural computation, 1997, 9(8): 735–1780.
- R. DAŞ, B. Polat, ve G. Tuna, “Derin Öğrenme ile Resim ve Videolarda Nesnelerin Tanınması ve Takibi”, Fırat Üniversitesi Mühendislik Bilimleri Dergisi, c. 31, sy. 2, ss. 571–581, 2019, doi: 10.35234/fumbd.608778.
- Wang, H. G., Sarawate, N. N., & Leu, M. C. (2004, July). Recognition of American sign language gestures with a sensory glove. In Japan USA Symposium on Flexible Automation, Denver, CO (pp. 102-10
- <https://www.w3schools.com/html/>
<https://spreadthesign.com/tr.tr/search/>
- <https://medium.com/nerd-for-tech/review-quo-vadis-action-recognition-a-new-model-and-the-kinetics-dataset-video-classification-a7535aa8bf48>
https://v-iashin.github.io/video_features/models/i3d/
- L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4305–4314, 2015
<https://doi.org/10.48550/arXiv.2012.06567>
- @INPROCEEDINGS{albanie20_bsl1k, title = {{BSL-1K}: Scaling up co-articulated sign language recognition using mouthing cues}, author = {Albanie, Samuel and Varol, Gökhan and Momeni, Liliane and Afouras, Triantafyllos and Chung, Joon Son and Fox, Neil and Zisserman, Andrew}, booktitle = {ECCV}, year = {2020}}
- <https://www.geeksforgeeks.org/long-short-term-memory-lstm-rnn-in-tensorflow/>
<https://github.com/google/mediapipe/blob/master/docs/solutions/holistic.md>

1.6. Overview

This document is divided into four sections.

The first chapter is a brief introduction to our project and is divided into six subsections (purpose of the system, scope of the system, Objectives of the project and success criteria, Definitions, Resources, Overview).

The second part is the current system, this part will explain the Turkish sign recognition model system currently in use. It will explain how this program works and TSRM related issues and also explain how to fix it.

This section will contain all the information regarding the Turkish sign recognition model (TSRM). This section has four subsections, the first is an overview of the system's functions, the second is the functional requirements that describe the high-level functionality of the system, and the third is the user-level requirements and non-functional requirements that discuss its division. divided into eight subsections, respectively (Usability, Reliability, Performance, Supportability, Implementation, Interface, Packaging, Law), the fourth is system models, which includes five subsections, respectively (scenarios, use case model, object model, dynamic model, user interface).

The fourth part is in the application section, we examine the operational aspects of our system by providing a comprehensive guide on its functionality and use.

2. Current System

There have been some previous studies and models related to this topic, and some of them have been utilized in this project. The Ankara University Turkish Sign Language (AUTSL) dataset, created by Ankara University and recorded using the Kinect device, consists of RGB videos. This dataset is a fundamental data source for developing the Turkish sign language recognition model.

Additionally, some models have been developed related to this topic. The InceptionI3d model is a deep learning architecture specifically designed for classifying videos. This model uses convolutional layers to consider the 3D structure of videos (time, height, and width) and learns the motion and temporal relationships in the videos. However, a detailed Turkish sign language translation application has not been developed in Turkey. Existing applications provide one-way translation. This cannot be beneficial for all users; hence, in this project, we aim to provide bi-directional translation. Additionally, people can use the learning module to learn and recognize Turkish sign language.

3. Proposed System

3.1.Functional Requirements

- Hearing-Impaired Individual, Non-Sign Language User and Content Creators are required to login to the system.
- The application should provide a user registration form with fields for essential information, such as username, email, and password.
- The system should be able to recognize Turkish Sign Language movements accurately and it must support a wide range of TSL gestures.
- The user performs TSL gestures in front of the camera, and the application translates the gestures into written text in real-time.
- A user navigates through the application, and initiates a translation effortlessly.
- The hearing individual, unfamiliar with TSL, accesses the application's learning module to better understand common TSL gestures.
- There should be a designated input field where Non-Sign Language Users can enter text related to Turkish Sign Language (TSL) expressions.
- Upon entering text, the application should display the corresponding video demonstrating the TSL expression associated with the entered text.
- When the users click on the "Logout" button, the application should display a confirmation prompt to confirm the intention to log out.

3.2.Nonfunctional Requirements

3.2.1 Usability

Usability is a critical non-functional requirement for the application translating Turkish Sign Language (TSL) movements into text. It focuses on ensuring that the application is user-friendly, efficient, and provides a positive experience for its intended users. Here are key aspects of usability for this application:

User Interface (UI) Design: The user interface should be visually appealing, intuitive, and culturally sensitive to the needs of the Turkish Sign Language user community.

User Interaction: Gestures and interactions for initiating translations and accessing features should be natural and easy to perform for users proficient in Turkish Sign Language.

Learnability: The application should be easy to learn, even for users with varying levels of technological proficiency or familiarity with similar applications.

Efficiency: The translation process should be efficient, with minimal delays between sign language gestures and the generation of translated text.

Accessibility: Adhere to accessibility standards to ensure that the application is usable by individuals with different abilities, including those with motor skills or vision impairments.

3.2.2. Reliability

Reliability is a critical non-functional requirement for an application translating Turkish Sign Language (TSL) movements into text, ensuring consistent and dependable performance. The application must exhibit high recognition accuracy, reliably translating a diverse range of TSL gestures to text with minimal errors. System uptime is crucial, demanding a robust server infrastructure and effective monitoring to minimize downtime and maintain continuous accessibility. Consistent translations across similar gestures build user trust, necessitating a robust translation algorithm and regular system updates. Finally, a dependable and trustworthy system is established.

3.2.3. Performance

The performance, encompassing several key aspects to ensure an efficient and responsive user experience. Real-time gesture recognition is paramount, demanding low latency to maintain a seamless communication flow. Translation speed is a crucial factor, requiring optimization of algorithms and backend processes for swift and accurate results. Overall response time, including user interface interactions, should be minimized to enhance user satisfaction. In addition, dataset augmentation improves the performance of the system and helps to achieve better results in different sign language recognition scenarios.

3.2.4. Supportability

The system should be easy to update and maintain. New sign categories should be added without major changes to the system architecture. Clear and informative error messages or warnings will be provided to guide users when something goes wrong and suggest possible solutions. Collectively, these supportability features contribute to flexible, adaptable and user-friendly implementation.

3.2.5. Implementation

Gesture Recognition:

- Frameworks: OpenCV or PyTorch offer powerful image processing and machine learning capabilities for accurate gesture analysis.
- Scripting: Python's versatility in machine learning and backend development makes it an ideal choice.
- Machine Learning and Computer Vision: Convolutional Neural Networks (CNNs) or other deep learning models can effectively recognize and interpret sign language gestures.

Application Development:

- **Frontend:** HTML5 provides a structured framework for web content, ideally suited for displaying sign language videos.

CSS is a style language used to determine the appearance (style, color, size, etc.) of web pages. It takes the structural content of HTML and applies a visual style to this content.

- **Backend:** Python excels in backend development, handling logic and data management efficiently.
Flask used for backend development, Flask is a lightweight and modular web framework. It provides the necessary tools and building blocks for creating and managing web applications.

Real-Time Gesture Recognition:

- Computer Vision and Deep Learning Libraries: TensorFlow, PyTorch, OpenCV, and MediaPipe provide pre-trained models and tools for real-time hand gesture, body language, and facial expression recognition.
- TSL Gesture Recognition Model: Leverage existing pre-trained models or train custom ones for specific TSL gesture recognition needs.

Inception I3D is a deep learning model with a 3D architecture. This model is specifically designed for the classification of video data. By expanding the filters in 2D convolution layers and pooling layers to three dimensions (time, height, and width), I3D can better represent the spatial and temporal features of videos.

The LSTM model is a type of recurrent neural network (RNN) architecture designed to overcome the vanishing gradient problem in traditional RNNs

and capture long-range dependencies in sequential data

Text Generation and Translation:

- **Text Input Processing:** The user-provided text is tokenized and processed to prepare it for gesture recognition and video matching.
- **Video Database:** A structured collection of sign language videos indexed by their corresponding words and phrases

Web App Development:

- **Web App Development Frameworks:** Flask: Flask is a lightweight and flexible web framework based on Python. It is known for its simple and modular structure, starting as a minimal framework and expandable as needed.

3.2.6. Interface

HTML is a markup language used to structure the content of web pages. CSS is used to style HTML content and make it visually appealing. Flask is a Python-based web framework that facilitates the development of web applications. When HTML, CSS, and Flask come together, they can create interactive and visually pleasing web interfaces for users.

3.2.7. Packaging

When we will building the app,we will compile the source code into binary files specific to each platform.We will create signing certificates for ensuring authenticity and security of our app.We will configure manifest files for specifying app permissions, icons, and other metadata.

3.2.8 Legal

We work with AUTSL dataset and make publications which is provided by Ankara University Computer Vision and Machine Learning Laboratory.Our project is intended for academic purposes so it is allowed to use it.

3.3.System Models

3.3.1.Scenarios

- ❖ Login
- ❖ Register
- ❖ Access to the learning module
- ❖ Translate TSL to Text
- ❖ Enter the text and view the corresponding video
- ❖ Logout

Name of event	Login	
Summary	Registered user enters the application to access translation features.	
Preconditions	The user has a registered account.	
Postconditions	User is successfully logged into the application and gains access to translation features	
Primary Actor(s)	Hearing-Impaired Individual, Non-Sign Language User	
Secondary Actor(s)	None.	
Flow of events	Step	Action
	1	User opens the TSL translation application on their computer.
	2	The application presents a login screen with fields for entering the user's credentials (username or email and password).
	3	User enters their registered username or email and password into the corresponding fields.
	4	User clicks the "Login" button.
	5	The application validates the entered credentials, checking for the correctness of the username or email and password.
	6	Upon successful authentication, the application creates a secure session for the user, allowing access to translation features.

Name of event	Register	
Summary	New user creates an account to access translation features.	
Preconditions	None	
Postconditions	User is successfully registered and gains access to the application's features.	
Primary Actor(s)	Hearing-Impaired Individual, Non-Sign Language User	
Secondary Actor(s)	None	
Flow of events	Step	Action
	1	User opens the TSL translation application on their computer.
	2	The application presents a registration screen with fields for entering necessary information, such as username, email, and password etc.
	3	User enters the required registration details into the corresponding fields.

	4	User clicks the "Register" or "Sign Up" button.
	5	The application validates the entered information, checking for the correctness of the data and ensuring that the username or email is not already in use.
	6	If the entered information is valid, the application proceeds to create a new user account.
	7	Users log in using their registered credentials.

Name of event	Access to learning module	
Summary	User who doesn't know TSL accesses the learning module to learn and enhance their proficiency in Turkish Sign Language.	
Preconditions	User is logged into the TSL translation application.	
Postconditions	User gains access to the learning module and its educational content to learn TSL.	
Primary Actor(s)	Non-Sign Language User	
Secondary Actor(s)	None	
Flow of events	Step	Action
	1	User is logged into the TSL translation application.
	2	User navigates to the main dashboard or menu of the application.
	3	The application displays options for various features, including the learning module.
	4	User selects the "Learning Module" option.

	5	The application loads the learning module interface, presenting educational videos and quizzes.
--	---	---

Name of event	Translate TSL to Text	
Summary	The user uses the TSL translation application to Decode TSL gestures into written text, improve communication and understanding between sign language users.	
Preconditions	User is logged into the TSL translation application.	
Postconditions	User receives accurate written text translations of the input TSL gestures.	
Primary Actor(s)	Non-Sign Language User, Hearing-Impaired Individual	
Secondary Actor(s)	None	
Flow of events	Step	Action
	1	User is logged into the TSL translation application.
	2	The user navigates to the translation feature within the application.
	3	The application presents the translation interface, ready to receive TSL input.
	4	The user performs a TSL gesture in front of the device's camera or input sensor.
	5	The application processes the TSL gesture using computer vision or sensor technology to recognize the sign.
	6	The application translates the recognized TSL gesture into written text.
	7	The user receives a written text translation of the TSL movement.

Name of event	Enter Text and View Corresponding Video	
Summary	The user uses the TSL translation application to enter written text and receive a corresponding video representation in Turkish sign language (TSL).	
Preconditions	User is logged into the TSL translation application.	
Postconditions	User views a video demonstration corresponding to the entered text.	
Primary Actor(s)	Non-Sign Language User	
Secondary Actor(s)	None	
Flow of events	Step	Action
	1	User is logged into the TSL translation application.
	2	The user navigates to the feature that allows text entry and corresponding video viewing.
	3	The application presents a text entry interface, ready to receive user input
	4	The user enters written text into the provided interface, specifying the content or phrase they want to see demonstrated in TSL.
	5	The application processes the entered text, preparing to fetch the corresponding video content.

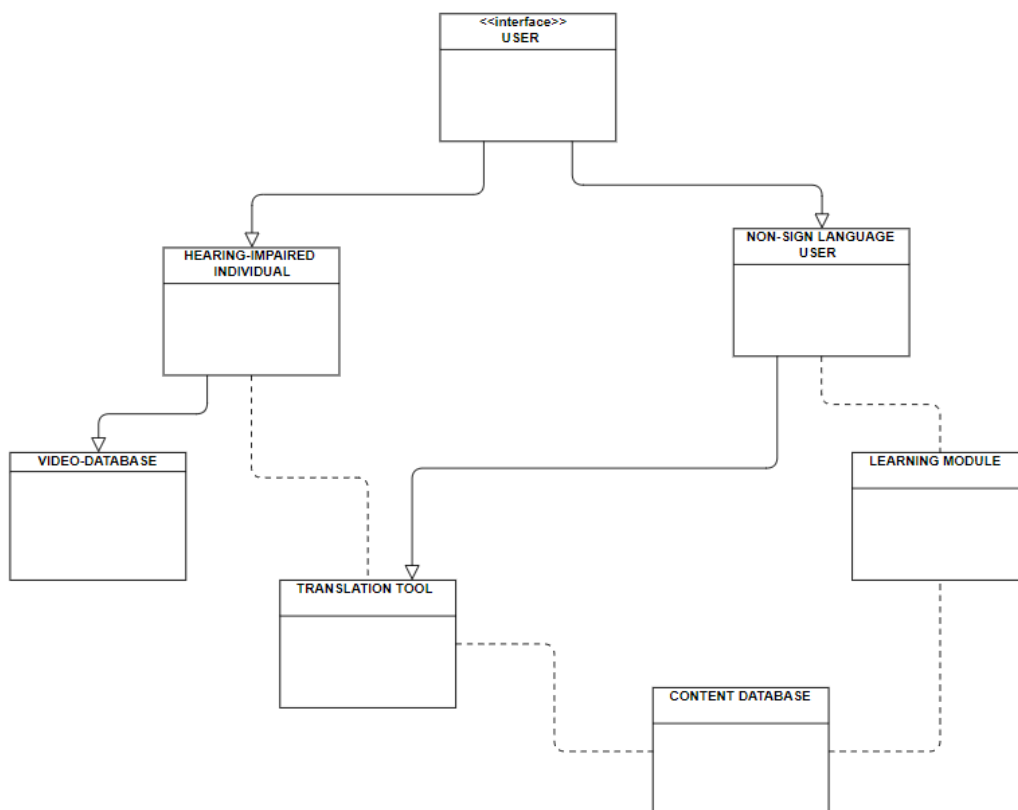
	6	The application retrieves the video content corresponding to the entered text and displays it for the user.
	7	The user views the video demonstration, providing a visual representation of the signed expression for the entered text.
	8	The user may have the option to repeat the video, explore additional related videos, or proceed with further learning.

Name of event	Logout	
Summary	The user concludes their session in the TSL translation application by logging out, ensuring the security of their account and data.	
Preconditions	User is currently logged into the TSL translation application.	
Postconditions	User is successfully logged out, and the application returns to the login screen.	
Primary Actor(s)	Hearing-Impaired Individual, Non-Sign Language User	
Secondary Actor(s)	None	
Flow of events	Step	Action
	1	The user, while in the application, navigates to the logout option.
	2	The application display a confirmation prompt to ensure the user intends to log out.
	3	The user, upon viewing the confirmation prompt, accepts the intention to log out.
	4	The application logs out the user, terminating the current session.
	5	The application returns the user to the login screen, ready for the next login session.

3.3.2. Use Case Model



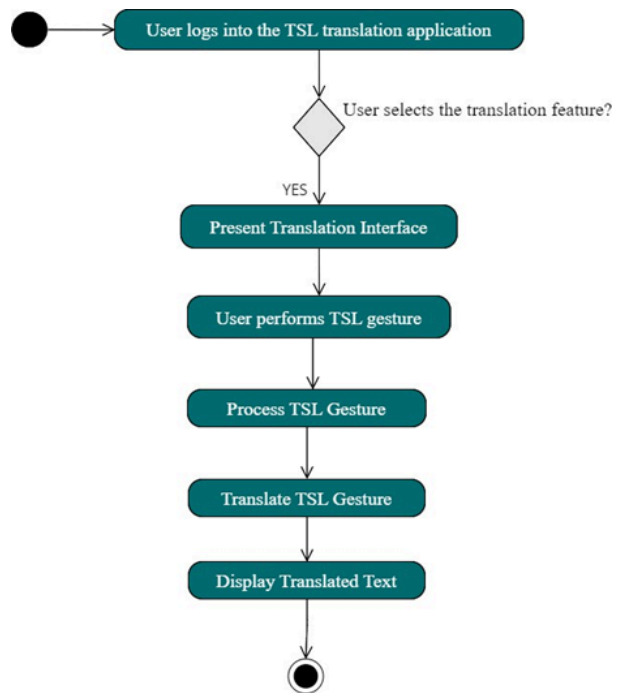
3.3.3. Analysis Object Model



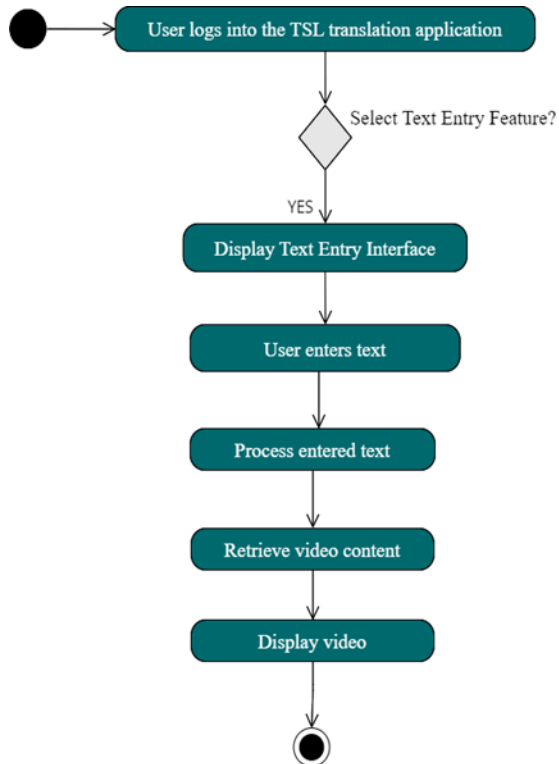
3.3.4. Dynamic Model

3.3.4.1. Activity Diagrams

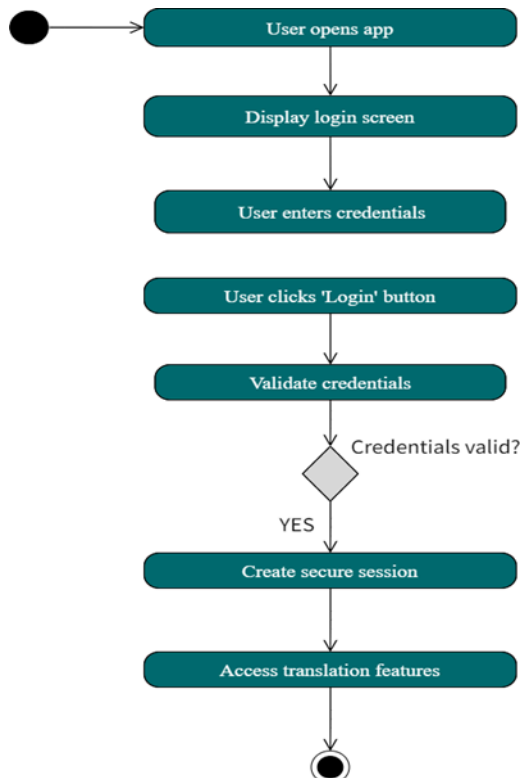
Translate TSL to Text



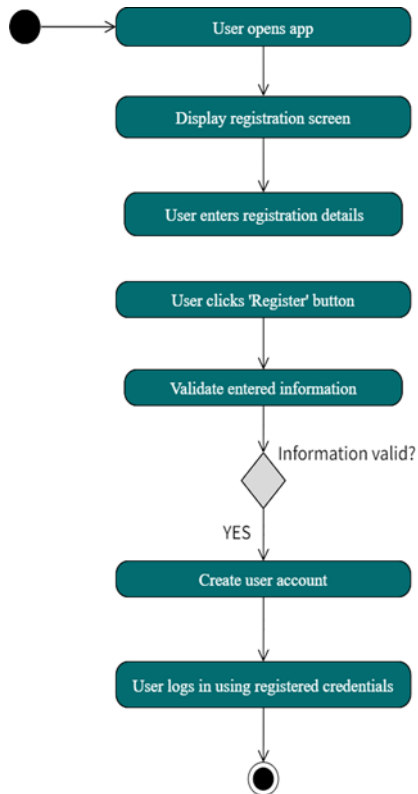
Enter Text and View Corresponding Video



Login

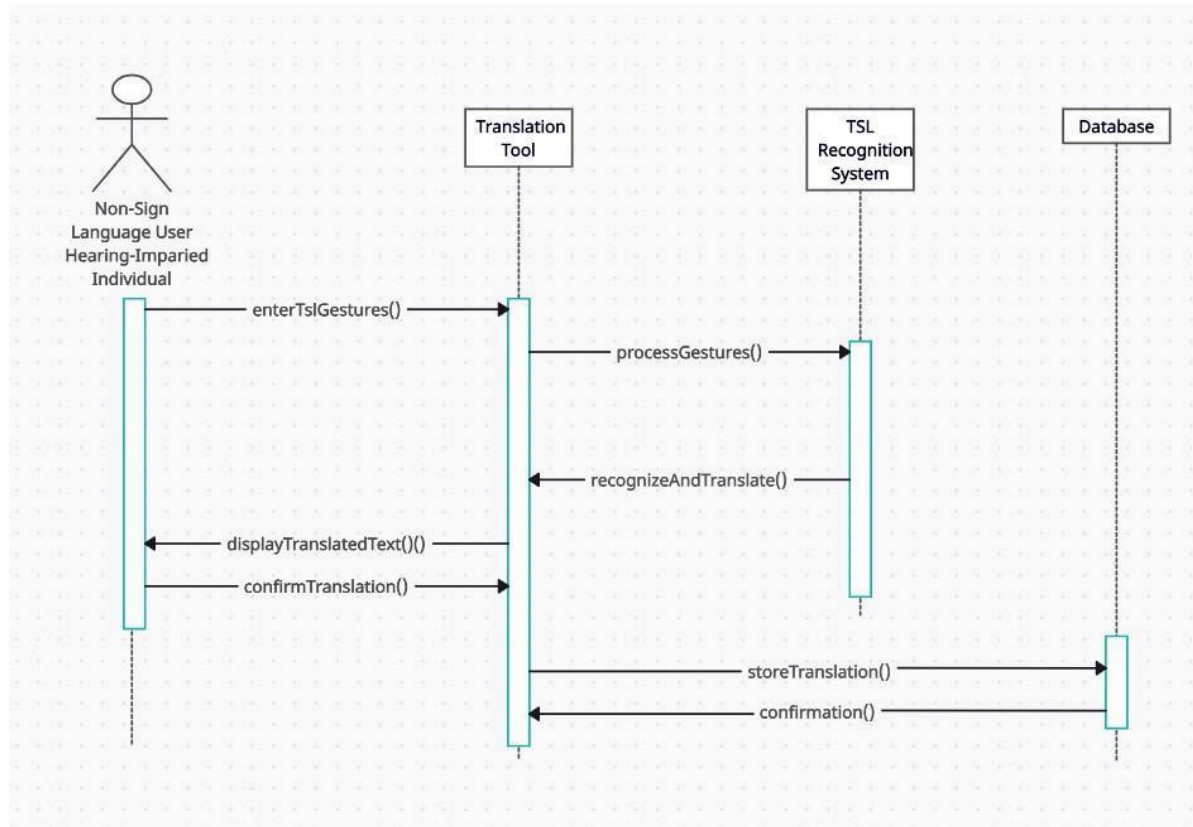


Register

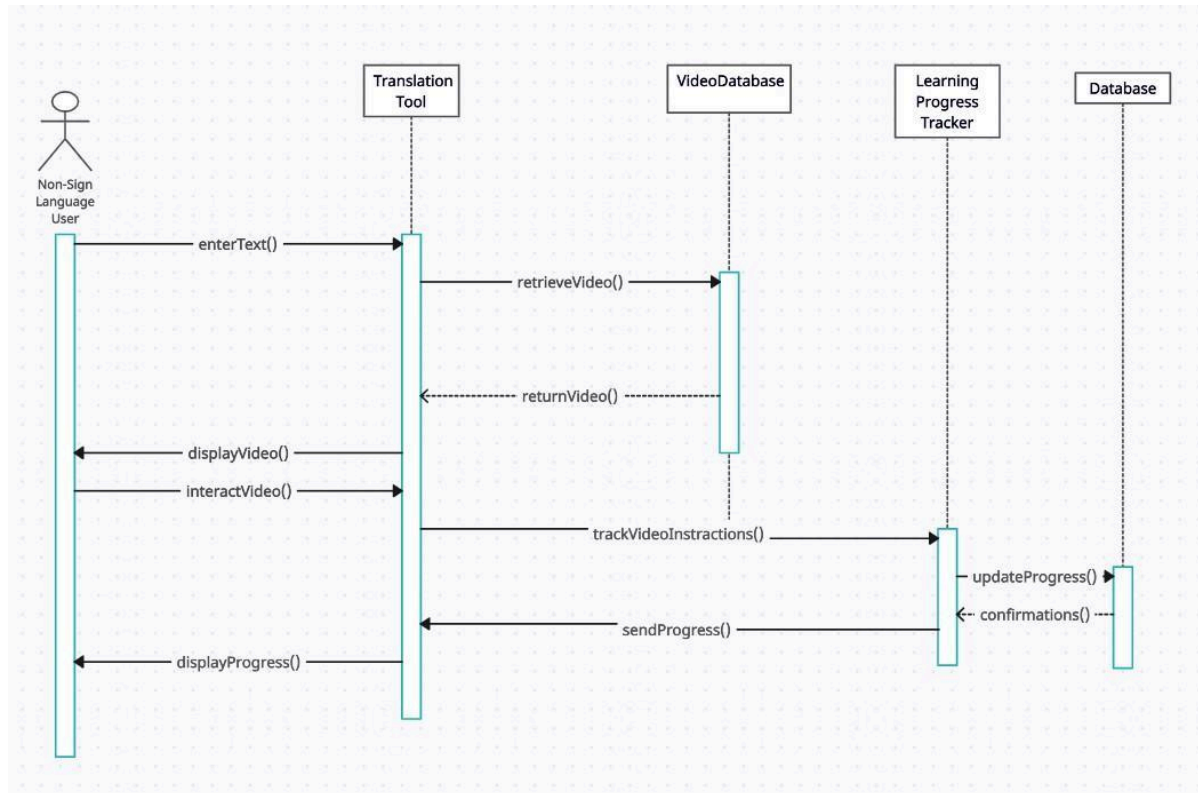


3.3.4.2. Sequence Diagrams

Translate TSL to Text



Enter Text and View Corresponding Video



4.Glossary

- TSL:Turkish Sign Language
- TSRM:Turkish Sign Language Model
- TSLT:Turkish Sign Language Translator
- SLR :Sign Language Recognition
- ML: Machine Learning
- DL: Deep Learning
- UI:User Interface
- CNN: Convolutional Neural Network
- RNN: Recurrent Neural Network)
- LSTM: Long Short-Term Memory
- AUTSL: Ankara University Turkish Sign Language Dataset

5. Application

First, we began researching and experimenting with models and deep learning techniques that could achieve high accuracy rates in recognizing various signs. The Inception I3D model is suitable for video-based applications due to its effective processing of spatial and temporal features. This model demonstrates strong performance when working with video data, as it can capture features across both time and space dimensions. Given the complexity and variability of sign language gestures, selecting the appropriate model and applying suitable training methods is essential for obtaining reliable performance. Therefore, the correct configuration and training of deep learning models are critical for accurately recognizing sign language.

Training Inception I3D Model for 3 Signers on 10 Word AUTSL Dataset

In this project, the AUTSL dataset was used, consisting of video sequences representing different Turkish Sign Language (TSL) words, each performed by various signers. Initially, the dataset included 10 words, each signed by three individuals. Each signer expressed the words at different speeds and styles, adding diversity to the dataset. This dataset was used to evaluate the model's baseline performance and to establish a starting point.

Training and evaluating deep learning models generally require high computational power, especially when dealing with large datasets and complex models. Working with a CPU for such tasks can be slow, resulting in very long training times. Therefore, in this project, GPU was used to accelerate the training and evaluation processes. CUDA is a platform and programming model that enables parallel computing using NVIDIA GPUs. The use of CUDA allows the model to be trained faster and more efficiently.

The use of GPU significantly shortens the training time due to its capacity to perform large matrix operations and deep learning computations in parallel. The Inception I3D model's complex computations were performed much faster, allowing training to be conducted over more epochs with larger datasets.

In the initial phase of the project, the Inception I3D model was trained using a small subset of the AUTSL dataset, which consisted of 10 Turkish Sign Language (TSL) words, each performed by three different individuals.

This section explains how to process video data using the VideoDataset class and convert it into a format that can be used for model training. The VideoDataset class loads, transforms, and prepares the videos for training by taking a specified list of videos and a root directory. Designed to be compatible with the PyTorch data loader, the VideoDataset class performs the necessary preprocessing while loading the videos. For this class to work, the following Python libraries need to be installed:

- cv2 (OpenCV)
- os
- numpy
- torch
- torchvision

The training process was carried out using the `train` function. This function processes the training data through the model for each epoch, calculates the loss function, and updates the model weights via backpropagation.

- For each batch, input data and labels are passed through the model.
- The loss is calculated using the `CrossEntropyLoss` function and the model is updated through backpropagation.
- At the end of each epoch, the average loss and accuracy are calculated and written to the log file.

Libraries and Settings Used

`numpy`, `torch`, `torch.optim`, `torch.nn.functional`, `matplotlib.pyplot`

`VideoDataset`: A custom dataset class for loading and processing videos

`DataLoader`: A PyTorch class used to load training and evaluation data

Model and Training Parameters

`BATCH_SIZE`: 4

`INPUT_SIZE`: 224

`worker_n`: 8 (number of workers used for the `DataLoader`)

`EPOCHS`: 30

`LR` (Learning Rate): 0.01

`model_num`: 2

`version`: 2

`MODE`: 'train' or 'test' (to set the mode)

`test_epoch`: 15

The model was trained for 30 epochs, and the results are as follows:

Training Results (Epoch 30):

- Average Loss: 0.077555
- Accuracy: 99.320% (146/147)

Validation Results (Epoch 30):

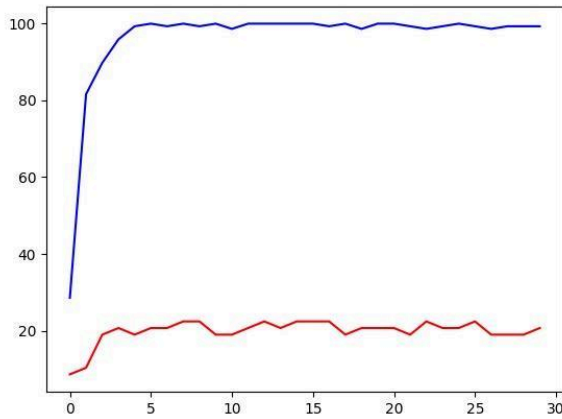
- Average Loss: 3.839085
- Accuracy: 20.690% (12/58)

Best Validation Loss: 2.597122 (achieved at epoch 1)

Analysis and Evaluation:

The training results indicate that the model performed quite well on the training set. With an average loss value of 0.077555 and an accuracy rate of 99.320%, it demonstrates that the model learned the training data very effectively. This high accuracy rate suggests that the model could classify almost all training examples correctly, which appears promising initially.

In contrast, the validation results were quite poor. The average loss for the validation set was 3.839085, and the accuracy rate was only 20.690%. This significant difference between the training and validation performance is a strong indicator of overfitting. Overfitting occurs when the model learns the training data too well but fails to generalize to new, unseen data.



Training Inception I3D Model for 10 Signers on 10 Word AUTSL Dataset

To prevent overfitting, data augmentation techniques were employed. Among these techniques, generating new data samples by randomly cropping and horizontally flipping each image was implemented. This method involves randomly cropping each image and then horizontally flipping it. These operations diversify each sample into different images, enhancing the diversity of the dataset.

Additionally, to increase the dataset, the number of signers was increased to 10. This enhances the diversity of the dataset by simulating that each sign was performed by different individuals, thus strengthening the model's generalization capability.

Lastly, the processed images were converted to tensor format and normalized. This brings the image data into a suitable format for the model to process and is crucial for stabilizing the training process.

The model was trained for 30 epochs, and the results are as follows:

Training Results (Epoch 30):

- Average Loss: 0.039978
- Accuracy: 99.328% (591/595)

Validation Results (Epoch 30):

- Average Loss: 10.772554
- Accuracy: 13.793% (8/58)

Best Validation Loss: 2.597122 (achieved at epoch 1)

Looking at the data, we observe that by the end of the training process, the model has fit the training data with high accuracy. However, the performance on the validation set is notably poor. The high average loss value and low accuracy rate indicate that the model has overfit to the training data, resulting in weak generalization ability. Additionally, considering that the best validation loss was achieved in the 1st epoch, it can be inferred that the model experienced overfitting during the training process. This situation limits the model's ability to generalize to new data. Therefore, strategies such as regularization techniques or utilizing more data may need to be considered to prevent overfitting and improve the model's generalization capability.

Training Inception I3D Model for 10 Signers on 25 Word AUTSL Dataset

Due to the unsatisfactory performance of the results, we considered trying a different approach to enhance the model's ability to distinguish between classes. For this purpose, we decided to increase the number of classes and expand the vocabulary to 25 words.

Increasing the number of classes can improve the model's generalization ability by enabling it to learn from more diverse datasets. This can allow the model to better distinguish between extremely subtle differences in various signs.

Similarly, increasing the vocabulary size can also enhance the diversity of different signs and gestures that the model can learn. This enables the model to recognize a wider range of movements.

The model was trained for 30 epochs, and the results are as follows:

Training Results (Epoch 30):

- Average Loss: 0.039978
- Accuracy: 99.328% (591/595)

Validation Results (Epoch 30):

- Average Loss: 0.623953
- Accuracy: 81.757% (121.0/148.0)

Test Results:

- Average Loss: 2.034901
- Accuracy: 62.500% (45.0/72.0)

The high accuracy on the training dataset indicates that the model has effectively learned from the training data. The increased accuracy and decreased loss values on the validation and test datasets demonstrate an improvement in the model's generalization ability. Higher accuracy rates imply that the model is better adapting to new data. These results indicate an enhancement in the model's performance and its ability to make more reliable predictions.

By increasing the word count to 25 and using techniques such as Batch Normalization, L2 Regularization, Dropout, and Early Stopping, we aimed to enhance the model's performance:

1. **Batch Normalization:** By normalizing the distribution of each mini-batch during training, Batch Normalization stabilizes and accelerates the training process. This helps prevent the model from becoming overly sensitive to the initial weights and reduces overfitting.
2. **L2 Regularization:** By penalizing large weights in the model, L2 Regularization prevents overfitting. It can enhance the model's generalization ability by keeping the weights small.
3. **Dropout:** Dropout randomly disables neurons during training, making the model more robust and preventing overreliance on specific neurons. This technique can significantly reduce overfitting and improve generalization ability.
4. **Early Stopping:** By monitoring the validation loss during training and stopping the training process when improvement ceases, Early Stopping prevents overfitting. This technique stops the training before the model memorizes the training data excessively.

By employing these techniques, we aimed to improve the model's performance and prevent overfitting, thereby enhancing its ability to generalize to unseen data.

The final results after these techniques:

Training Results (Epoch 30):

- Average Loss: 0.007427
- Accuracy: 99.933% (1492.0/1493.0)

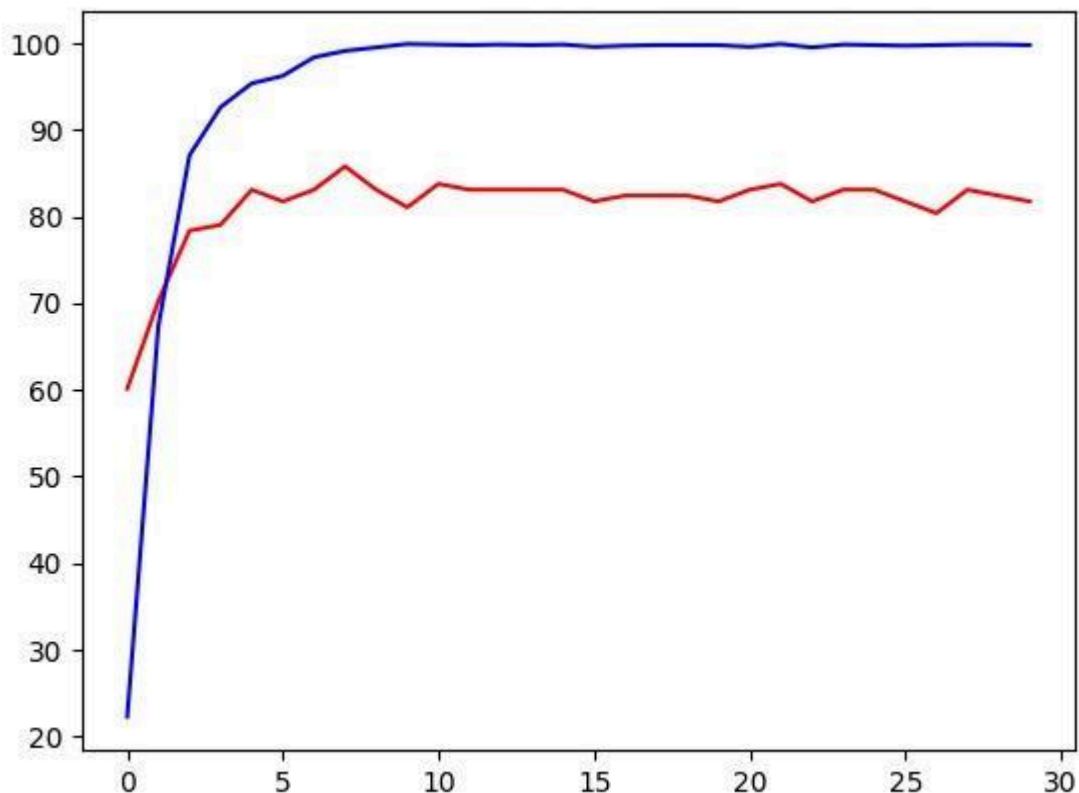
Validation Results (Epoch 30):

- Average Loss: 0.617918
- Accuracy: 83.108% (123.0/148.0)
- Best Validation Loss: 0.597155 (achieved at epoch 16)

Test Results:

- Average Loss: 2.604720
- Accuracy: 63.514% (47.0/74.0)

These results indicate that the training process of the model has been successful. High accuracy and low loss values in the training dataset demonstrate that the model has effectively learned from the training data. The increased accuracy rates in the validation and test datasets indicate an improvement in the model's generalization ability. This suggests that the model adapts better to new data and makes more reliable predictions. There has been an enhancement in the model's performance with the application of data augmentation and other techniques.



Training with LSTM Model for 24 Words On Our Dataset

After the Inception I3d model was tried, we decided to try another model in order to compare the results and find the best one. We tried the LSTM Model.

Layer (type)	Output Shape	Parameter
lstm_4 (LSTM)	(None, 30, 64)	442112
batch_normalization_5 (Batch Normalization)	(None, 30, 64)	256
dropout_5 (Dropout)	(None, 30, 64)	0
lstm_5 (LSTM)	(None, 30, 128)	98816
batch_normalization_6 (Batch Normalization)	(None, 30, 128)	512
dropout_6 (Dropout)	(None, 30, 128)	0
lstm_6 (LSTM)	(None, 64)	49408
batch_normalization_7 (Batch Normalization)	(None, 64)	256
dropout_7 (Dropout)	(None, 64)	0
dense_3 (Dense)	(None, 64)	4160
batch_normalization_8 (Batch Normalization)	(None, 64)	256
dropout_8 (Dropout)	(None, 64)	0
dense_4 (Dense)	(None, 32)	2080
batch_normalization_9 (Batch Normalization)	(None, 32)	128
dropout_9 (Dropout)	(None, 32)	0
dense_5 (Dense)	(None, 24)	792

Total params: 598776 (2.28 MB)

Trainable params: 598072 (2.28 MB)

Non-trainable params: 704 (2.75 KB)

This output shows the summary of the LSTM model we provided. Here's a breakdown of what each part means:

Layer (type): This column specifies the type of layer used in the model. Here, you see "LSTM" for Long Short-Term Memory layers, "BatchNormalization" for batch normalization layers, "Dropout" for dropout layers, and "Dense" for fully-connected dense layers.

Output Shape: This column indicates the size and dimensionality of the output produced by each layer. For example, "lstm_4 (LSTM)" has an output shape of "(None, 30, 64)". This means:

- "(None)": The batch size can vary (it's not explicitly defined here).
- "30": This represents the number of time steps in the sequence data the model is processing.
- "64": This is the number of hidden units (features) learned by this LSTM layer.

Param #: This column shows the number of parameters (weights and biases) associated with each layer. The total number of parameters is crucial for understanding model complexity and potential for overfitting.

Layer Breakdown:

1. **lstm_4 (LSTM):** This first LSTM layer processes the input sequence data (likely with a shape of "(batch_size, 30, feature_count)"), which isn't shown here. It has 64 hidden units and outputs a sequence with 30 time steps and 64 features for each step.
 2. **batch_normalization_5 (BatchNormalization):** This layer normalizes the activations of the previous LSTM layer, improving training stability.
 3. **dropout_5 (Dropout):** This layer randomly drops 30% of the activations from the previous layer during training, preventing overfitting.
 4. **lstm_5 (LSTM):** The second LSTM layer takes the output of the first LSTM and further processes it. It has 128 hidden units.
 5. **batch_normalization_6 & dropout_6:** Similar to layers 2 & 3, these perform normalization and dropout on the output of the second LSTM.
 6. **lstm_6 (LSTM):** The third LSTM layer takes the output of the second LSTM and has 64 hidden units. However, its output shape is "(None, 64)" because `return_sequences=False` is likely used, indicating it only returns the final output vector for the entire sequence.
 7. **batch_normalization_7 & dropout_7:** Similar to layers 2 & 3, these layers normalize and apply dropout on the final output vector of the third LSTM.
- Dense layers and BatchNormalization/Dropout:** Three dense layers follow the LSTMs. These are fully-connected layers that transform the high-dimensional features from the LSTMs into a lower-dimensional representation suitable for classification. Each dense layer has its number of units (64, 32, and finally the number of classes, likely 24 in this case) and uses batch normalization and dropout for regularization.

Total Parameters:

- The model has a total of 598,776 parameters, which is a relatively moderate size for an LSTM model.

- Trainable parameters (598,072) are the ones the model can adjust during training to learn the optimal representation for the task.
- Non-trainable parameters (704) are typically fixed values like biases in batch normalization layers.

Understanding the Model:

This summary shows a well-structured LSTM model with three LSTM layers for capturing temporal dependencies in your data. Batch normalization and dropout are effectively used to prevent overfitting. The final dense layers with appropriate activation functions (likely sigmoid for hidden layers and softmax for the output layer) aim to classify the data into 24 categories.

Precision: 0.9076947358197357

Recall: 0.8916666666666667

F1 Score: 0.8887718150741437

Test accuracy: 0.8916666507720947

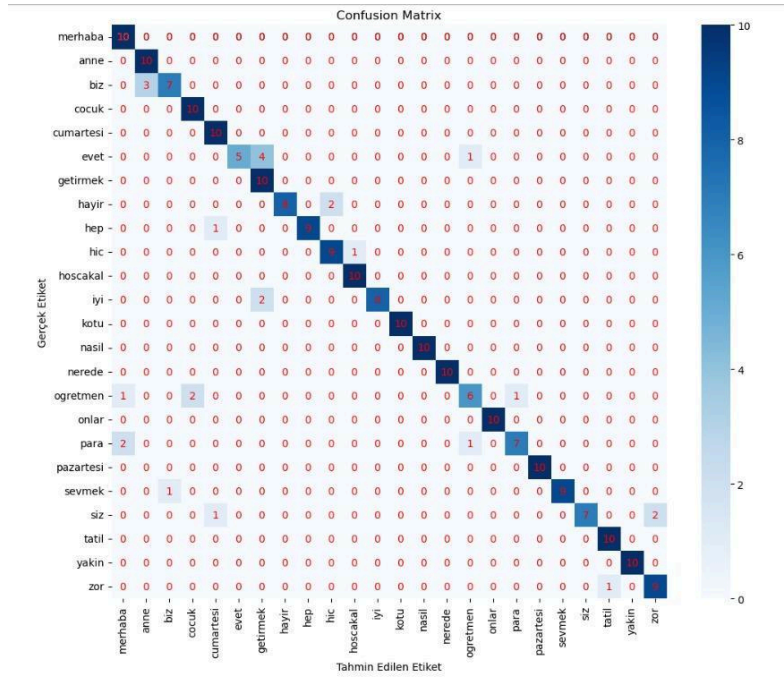
The model has achieved a high test accuracy of 0.8916666507720947. This shows that the model is quite successful in making accurate predictions in the test data.

Precision: The model correctly classified 90.77% of the instances it classified as positive. This indicates that the model has high precision and is less likely to generate false alarms.

Recall: The model correctly identified 89.17% of all actual positive examples. This suggests that the model is comprehensive and less likely to miss true positives.

F1 Score: The F1 score is the average of precision and recall, summarizing the model's overall performance in both making correct predictions and identifying all positive examples. For this model, the F1 score is 0.8887, indicating a good balance between precision and recall and overall good performance.

Below is the confusion matrix of this model we are using. The confusion matrix is a table that shows the relationship between the actual classes predicted by a classification model. Decryption matrix is a table that shows the relationship between the actual classes. This table is a useful way to visualize how correctly or incorrectly the model predicts which classes.



By making improvements to the number of LSTM layers and the dropout ratio numbers, we obtained the following results and observed an increase in test accuracy.

Layer (type)	Output Shape	Param #
lstm_20 (LSTM)	(None, 30, 64)	442112
batch_normalization_10 (Batch Normalization)	(None, 30, 64)	256
tchNormalization)		
dropout_20 (Dropout)	(None, 30, 64)	0
lstm_21 (LSTM)	(None, 30, 128)	98816
batch_normalization_11 (Batch Normalization)	(None, 30, 128)	512
tchNormalization)		
dropout_21 (Dropout)	(None, 30, 128)	0
lstm_22 (LSTM)	(None, 64)	49408
batch_normalization_12 (Batch Normalization)	(None, 64)	256
tchNormalization)		
dropout_22 (Dropout)	(None, 64)	0
dense_18 (Dense)	(None, 64)	4160
batch_normalization_13 (Batch Normalization)	(None, 64)	256

tchNormalization)

dropout_23 (Dropout)	(None, 64)	0
dense_19 (Dense)	(None, 32)	2080
batch_normalization_14 (Batch Normalization)	(None, 32)	128

tchNormalization)

dropout_24 (Dropout)	(None, 32)	0
dense_20 (Dense)	(None, 24)	792

The network consists of twelve sequentially arranged layers:

- **Two Long Short-Term Memory (LSTM) layers:** lstm_20 and lstm_21
- **A batch normalization layer** following each LSTM layer: batch_normalization_10 and batch_normalization_11
- **Three dropout layers:** dropout_20, dropout_21, and dropout_23
- **Three dense layers:** dense_18, dense_19, and dense_20
- **A batch normalization layer** following each dense layer: batch_normalization_13 and batch_normalization_14
- **A final dropout layer:** dropout_24

Data Flow

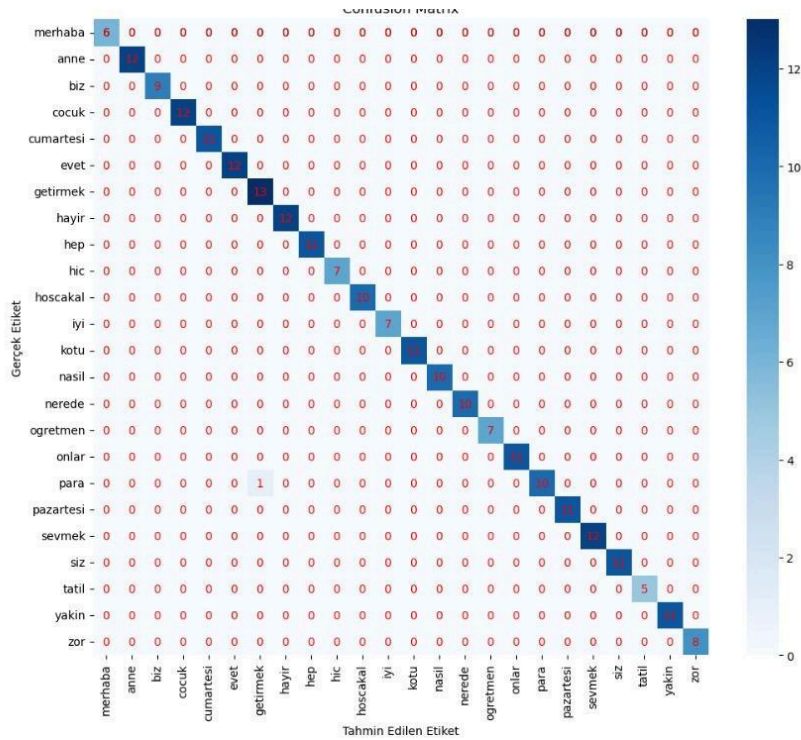
The network's structure suggests that it can process time-series data. The first two LSTM layers are used to extract information in the time dimension (30 time steps). The batch normalization layer following each LSTM layer forces the network to converge faster during training. Dropout layers are used to prevent overfitting.

The subsequent dense layers, using the features extracted by the LSTM layers, are likely used for a classification or prediction task. Batch normalization and dropout layers again improve the network's training at this stage. The last layer (dense_20) produces a 24-dimensional output. This suggests that the network can either predict 24 different classes or produce a 24-dimensional vector output.

Parameters

- **Total parameter count:** 598,776 (2.28 MB)
- **Trainable parameter count:** 598,072 (2.28 MB)
- **Untrainable parameter count:** 704 (2.75 KB)

Test accuracy: 0.9958333373069763



Model Performance:

- Achieved 99.58% test accuracy.

Model Compilation:

- **Optimizer:**
 - Adam optimizer automatically adjusts learning rate.
 - Learning rate set to 0.001.
- **Loss Function:**
 - Categorical Crossentropy loss function for multi-class classification.
 - Measures the difference between predicted and true classes.
- **Evaluation Metrics:**
 - Model evaluated using accuracy metric.
 - Accuracy indicates the correctness of model predictions.

Early Stopping and Learning Rate Reduction:

- **Early Stopping:**
 - Early Stopping callback halts training if validation loss doesn't improve for 10 epochs.
 - Prevents overfitting.
- **ReduceLROnPlateau:**
 - ReduceLROnPlateau callback reduces learning rate by 10% if validation loss doesn't improve for 5 epochs.
 - Helps find optimal model weights.

Conclusion:

- This LSTM model achieved a remarkable test accuracy of 99.58%.
- The model architecture, activation functions, Dropout, and Batch Normalization effectively facilitated learning.

Training with LSTM Model for 10 Words ON Our Dataset

Finally, we made the number of classes 10 and tried it, and in it we got the following results.

Layer (type)	Output Shape	Param #
lstm_27 (LSTM)	(None, 30, 64)	442,112
batch_normalization_45 (BatchNormalization)	(None, 30, 64)	256
dropout_45 (Dropout)	(None, 30, 64)	0
lstm_28 (LSTM)	(None, 30, 128)	98,816
batch_normalization_46 (BatchNormalization)	(None, 30, 128)	512
dropout_46 (Dropout)	(None, 30, 128)	0
lstm_29 (LSTM)	(None, 64)	49,408
batch_normalization_47 (BatchNormalization)	(None, 64)	256
dropout_47 (Dropout)	(None, 64)	0
dense_27 (Dense)	(None, 64)	4,160

batch_normalization_48 (BatchNormalization)	(None, 64)	256	
dropout_48 (Dropout)	(None, 64)	0	
dense_28 (Dense)	(None, 32)	2,080	
batch_normalization_49 (BatchNormalization)	(None, 32)	128	
dropout_49 (Dropout)	(None, 32)	0	
dense_29 (Dense)	(None, 10)	330	

Total params: 1,793,536 (6.84 MB)

Trainable params: 597,610 (2.28 MB)

Non-trainable params: 704 (2.75 KB)

Optimizer params: 1,195,222 (4.56 MB)

Model Architecture Breakdown

This section details the architecture of a neural network, likely used for classification. Here's a breakdown of each layer and its function:

- **lstm_27 (LSTM):** This is the first Long Short-Term Memory layer. It processes sequential data (30 time steps) with 64 hidden units.
- **batch_normalization_45 (BatchNormalization):** This layer normalizes the output of the LSTM layer, improving training stability.
- **dropout_45 (Dropout):** This layer randomly drops a certain percentage of activations during training, preventing overfitting.
- **lstm_28 (LSTM):** Similar to lstm_27, this processes the normalized output from the previous layer with 128 hidden units.
- **batch_normalization_46 & dropout_46:** These layers perform the same functions as their counterparts after lstm_27.
- **lstm_29 (LSTM):** The final LSTM layer takes the normalized output from lstm_28 and compresses it into a 64-dimensional vector.
- **batch_normalization_47 & dropout_47:** Similar functionality as before.
- **dense_27 (Dense):** This dense layer transforms the 64-dimensional vector into a vector with 64 units.
- **batch_normalization_48 & dropout_48:** Normalization and dropout for the dense layer's output.

- **dense_28 (Dense):** Another dense layer, reducing the dimensionality to 32 units.
- **batch_normalization_49 & dropout_49:** Normalization and dropout for the second dense layer.
- **dense_29 (Dense):** The final output layer with 10 units. This suggests the model classifies data into 10 categories.

Model Parameters

- **Total params:** 1,793,536. This represents the total number of numerical values the model uses to make predictions.
- **Trainable params:** 597,610. These parameters are adjusted during training to minimize errors on the training data.
- **Non-trainable params:** 704. These are likely pre-defined values (e.g., scaling factors in Batch Normalization) that are not modified during training.
- **Optimizer params:** 1,195,222. These parameters are specific to the optimization algorithm used to train the model (not shown here).

Test accuracy: 0.9833333492279053

This indicates that the model correctly classified 98.33% of the examples in the test dataset. This is a very high accuracy, suggesting the model generalizes well to unseen data.

Confusion Matrix:

```
[[ 6 0 0 0 0 0 0 0 0 0]
 [ 0 5 0 0 0 0 0 0 0 0]
 [ 0 0 7 0 0 0 0 0 0 0]
 [ 0 0 0 9 0 0 0 0 0 0]
 [ 0 0 0 0 2 0 0 0 0 0]
 [ 0 0 0 0 0 5 0 0 0 0]
 [ 0 0 0 0 0 0 4 0 0 1]
 [ 0 0 0 0 0 0 0 12 0 0]
 [ 0 0 0 0 0 0 0 0 3 0]
 [ 0 0 0 0 0 0 0 0 0 6]]
```

The model performs well on most classes, with high diagonal values (e.g., 12 in row 7, column 7).

There are some instances of misclassification between classes 1 and 0, 2 and 0, and 6 and 7 (indicated by non-zero off-diagonal values). Overall, the model achieves high accuracy with relatively few misclassifications. This model seems to be a reliable classifier for the task at hand.

In the other part of the project, a sophisticated Flask application is outlined, developed for sign

language recognition and education. This application seamlessly integrates diverse technologies such as computer vision, machine learning, and web development to create an interactive and user-friendly platform.

Model Loading and Prediction

At the core of this application lies a custom-trained sign language classification model (designated as `my_model4.h5`). This model processes sequences of keypoints—specific anatomical landmarks on the human body—to predict corresponding sign language words. The model is instantiated during application startup to ensure its availability for real-time predictions.

Real-Time Sign Language Detection

Leveraging MediaPipe, the application executes real-time detection of facial, hand, and body pose landmarks from a webcam feed. The `mediapipe_detection` function processes each frame, while the `draw_styled_landmarks` function superimposes detected landmarks onto the frame for visualization purposes.

Keypoints extracted from these landmarks are subsequently fed into the model, facilitating the recognition of sign language gestures.

The predicted word, along with its confidence score, is displayed on the video feed, providing instantaneous feedback.

User Management and Authentication

User authentication is meticulously managed via a robust system that supports registration and login through a SQLite database. Session management ensures users remain authenticated across different pages, thereby creating a seamless user experience.

Dynamic Routes and Templates

The application is architected with multiple routes, each serving a specific functionality:

- Login and Registration: Facilitates user authentication.

Giriş Yap

Kullanıcı Adı:

Şifre:

[Giriş Yap](#)

Hesabınız yok mu? [Kaydol](#)



Kayıt Ol

Ad Soyad:

E-posta:

Kullanıcı Adı:

Şifre:

Doğum Tarihi:

Cinsiyet:

[Kayıt Ol](#)

Zaten bir hesabınız var mı? [Giriş Yap](#)



- Home Page: Welcomes users and provides navigational links.

Hoşgeldiniz, ılayda

Gerçek Zamanlı Çeviri

İşaretlerinizi metne aktarın ve sürekli iletişimin tadını çıkarın!



Gerçek Zamanlı Çeviriye Başla

Metinden İşaret Diline Çeviri

Metninizi işaretlere dönüştürün ve kesintisiz iletişimin tadını çıkarın!



Metinden İşaret Diline Çevireye Başla

İşaret Dili Öğrenme Modülü

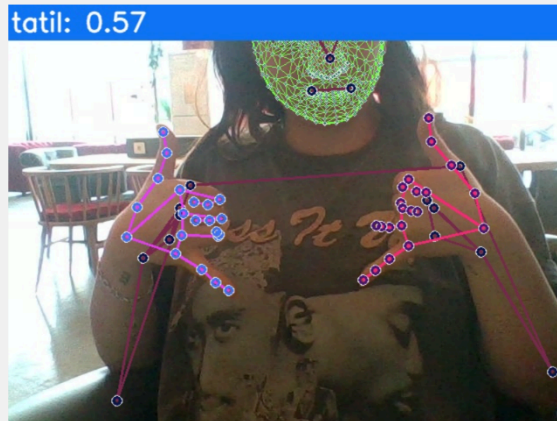
İnteraktif dersler ve sınavlarla işaret dilini öğrenin!



İşaret Dili Öğrenmeye Başla

- Real-Time Detection: Displays the webcam feed and predicts sign language gestures.

Real-Time Translation



- Text-to-Video Conversion: Allows users to input text and receive corresponding sign language videos.


Metinden İşaret Diline Çeviri

Çevirmek istediğiniz metni girin:

Çevir


- Learning Modules: Offers educational content through videos, images, and interactive elements.

Ders 1: Alfabe




[Öğren](#) [Quiz](#)

Ders 2: Meslekler




[Öğren](#) [Quiz](#)

Ders 3: Aile



[Öğren](#) [Quiz](#)

Ders 4: Selamlaşma



[Öğren](#) [Quiz](#)

Meslekler

Aşçı
Bakka
Eczacı
Fırıncı
Kuafor
Mimar
Mühendis
Pilot
Saatçi
Veteriner

aşçı



- Quizzes: Tests user knowledge across various sign language categories, including alphabets, professions, greetings, and family members.

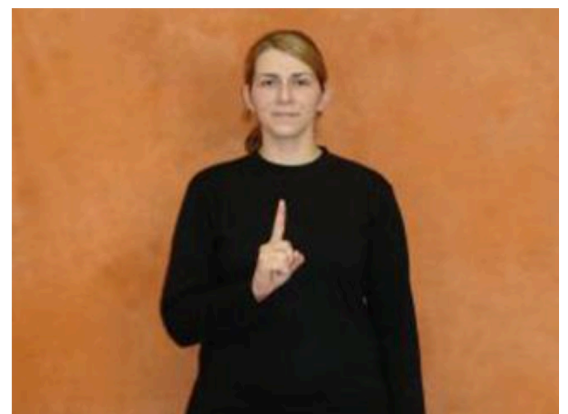
Meslek Quiz



Bu meslek nedir?

Gönder

Alfabe Quiz



Bu harf nedir?

Gönder

Yanlış! Doğru cevap: Bakka

Quiz Sonucu

Dynamic HTML templates render these pages, adapting based on user interactions and data retrieved from the database.

Stylish User Interface

The application employs CSS to enhance the visual appeal and usability of the HTML templates. The design is responsive and intuitive, ensuring a pleasant user experience across different devices. Key features include:

- **Header and Navigation:** A consistent and visually appealing header with centered navigation links.
- **Main Content:** Centrally aligned sections with interactive elements, hover effects, and smooth transitions.
- **Forms and Buttons:** User-friendly forms with clear instructions, styled buttons, and responsive feedback.

Interactive Quizzes

The quiz feature enhances the learning experience through interactive challenges. Each quiz randomly selects a word or video and presents it to the user with multiple-choice options. The application tracks user progress and scores, providing feedback upon quiz completion. Quizzes cover various categories, such as alphabets, professions, greetings, and family members.

Enhancements and Security

While the application is functional and comprehensive, several areas warrant further enhancement:

1. **Session Timeout:** Implementing session timeouts to log users out after inactivity, thereby enhancing security.
2. **Robust Error Handling:** Enhancing error handling mechanisms for potential issues such as camera access failures or model loading errors.
3. **Advanced Security Measures:** Incorporating password hashing, secure communication (HTTPS), and protection against common web vulnerabilities.
4. **Scalability:** Addressing aspects such as load balancing, database optimization, and efficient resource management to accommodate future growth.

Conclusion

This project represents a sophisticated and user-friendly application for sign language recognition and education. By leveraging modern web technologies, machine learning, and computer vision, it provides an interactive platform for users to learn and practice sign language effectively. With further enhancements and optimizations, this application has the potential to serve as a valuable educational tool for the hearing-impaired community and those interested in learning sign language.