ASL Recognition

Final project for the 2025 Postgraduate Course on Artificial Intelligence with Deep Learning

UPC School – Universitat Politècnica de Catalunya (BarcelonaTech)

Authors: Enrique Bermejo, Enric Dorca, Daniel Muelle, and Carles Vila

Supervisor: Amanda Duarte

Repository: https://github.com/cvilafer/AIDL-Project

Abstract

We propose a novel approach for American Sign Language (ASL) recognition that combines Google MediaPipe's real-time hand and body landmark tracking with a lightweight deep learning model trained on the MS-ASL dataset. We called this model “Key Frame MLP”, and it extracts the key frames features from the sequence of hand and pose landmarks, enabling efficient recognition without requiring RGB input. Evaluated on the MS-ASL 1000 dataset, our approach achieves **61% top-1 average class accuracy**, demonstrating strong performance relative to its simplicity. The entire system is optimized for real-time operation and low computational cost, making it suitable for deployment on edge devices. These results highlight the effectiveness of combining modular MLPs with fast landmark-based inputs for scalable sign language recognition.

Table of Contents

INTRODUCTION

Over 460 million people around the world are deaf or hard-of-hearing, representing a significant portion of the global population. This diverse community includes individuals with varying degrees of hearing loss, ranging from mild to profound. Many of them face unique challenges in communication, education, employment, and access to public services.

Of this population, more than 70 million people are deaf and use sign language as their primary means of communication. These individuals rely on visual language systems that are rich, complex, and fully capable of expressing abstract ideas, emotions, and technical concepts. There are over 300 different sign languages used worldwide, each with its own grammar, vocabulary, and cultural significance. These languages are not universal; they evolve naturally within communities and are deeply rooted in local culture and history.

Despite the prevalence and importance of sign languages, many deaf individuals still face barriers to communication and inclusion. Limited access to sign language interpreters, a lack of awareness in the hearing population, and inadequate support in education and healthcare systems contribute to social and economic inequalities.

Recognizing and supporting sign language users is essential for building a more inclusive world where deaf and hard-of-hearing individuals can fully participate in society, access information, and express themselves without barriers.

MOTIVATIONS

Inclusion

The primary motivation is to bridge the communication gap between the deaf and hearing communities.

Impact

Breaking down communication barriers, **SLR AI** helps Deaf and hearing communities understand each other better and interact more easily.

Application

The primary drive behind this Sign Language Recognition (SLR) AI project is the urgent need to translate advanced technological capabilities into tangible, user-friendly tools that directly address real-world communication barriers for the Deaf and hard-of-hearing community.

GOALS/MILESTONES

The main goal of this project is to develop a robust deep learning–based system for isolated American Sign Language (ASL) recognition. The work focuses on leveraging multimodal data and evaluating model performance across datasets of increasing complexity. The specific objectives are:

- Create a deep learning–based model for isolated American Sign Language recognition.
- Effectively process and extract features from multimodal input such as hand and body landmarks.
- Train models using MS-ASL 100 to evaluate performance on a small and manageable vocabulary set.
- Train models using MS-ASL 200 to assess scalability as the number of classes increases.
- Train models using MS-ASL 1000 to test generalization and robustness on a large and diverse dataset.

DATASET

The data we used to train and validate our model has been the MS-ASL (Microsoft AmericanSign Language.

The **MS-ASL** dataset is a large-scale, real-life dataset designed for advancing research in **sign language recognition**. It contains over **25,000 annotated videos** of American Sign Language (ASL) signs, making it one of the most comprehensive datasets in this domain. The dataset is particularly valuable for training machine learning models due to its diversity and scale.

Key Features

- **Large Vocabulary**: The dataset includes **1,000 unique signs**, covering a wide range of vocabulary.
- **Diverse Signers**: It features recordings from over **200 signers**, enabling robust generalization to unseen individuals.
- **Realistic Conditions**: Videos are captured in unconstrained, real-life settings, making the dataset suitable for practical applications.

At crunch time, the number of downloadable videos turned out to be lower. This is the summary of usable clips:

Training: 12808 (1000 signs. The word with more signs is “eat“ with 43 clips. Average is 12 clips)

Validation: 3258 (920 signs. The word with more signs is “happy” with 17 clips. The average is 4 clips)
Test: 2832 (932 signs. The word with more signs is “hello” with 17 clips. Average is 3 clips)

We didn’t filter videos with bad quality mediapipe coordinates.

The official sizes of the MS-ASL dataset are: 100, 200, 500 and 1000 signs. You can find the 1000 classes in the file MSASL_classes.json and they appear in the order of frequency. If you only require the MS-ASL 100 classes you only have to take the first 1000, and the same way for the other sizes.

*(dataset_example.png)

MEDIAPIPE

MediaPipe is an open-source framework developed by **Google** for building real-time, cross-platform **machine learning** and **computer vision** solutions, especially focused on video and image analysis.

Key Features:

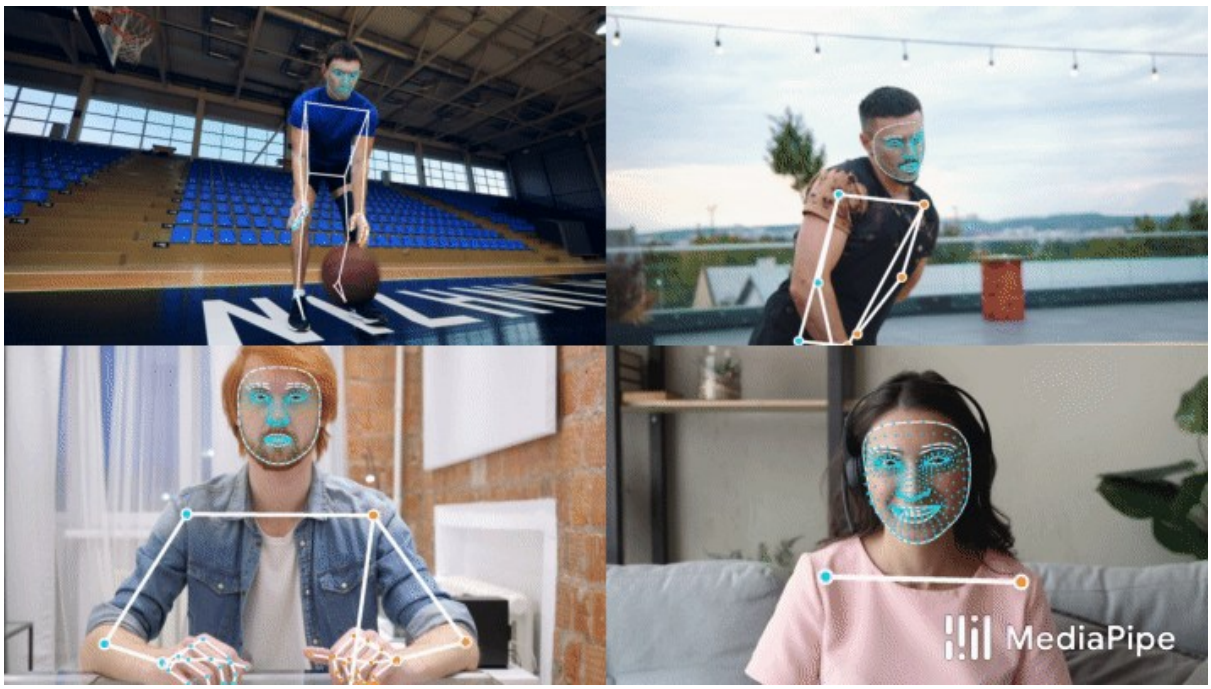
- Real-time performance: Optimized for applications like augmented reality, filters, gesture recognition, etc.
- Cross-platform: Works on Android, iOS, web (via WebAssembly), and desktop (Windows, macOS, Linux).
- Modular pipeline architecture: Uses customizable "graphs" made up of different processing blocks called calculators.
- Pre-built solutions: Includes ready-to-use ML models and tools for:
    - Hand Tracking
    - Face Mesh
    - Pose Estimation
    - Holistic Tracking (face + hands + body)
    - Objectron (3D object detection)
    - Selfie Segmentation

💡 Common Use Cases:

- Augmented reality and virtual try-ons
- Gesture and face recognition
- Fitness apps (e.g. posture tracking)
- Touchless user interfaces
- Real-time background removal

🛠️ Technology Stack:

- Built-in support for TensorFlow Lite
- GPU and CPU acceleration
- Web deployment with WebAssembly and WebGL



*(mediapipe_example.png)

COMPUTATIONAL RESOURCES

Google Virtual Machine g2-standard-4
- 4 virtual CPUs, 16 GB
- 1 NVIDIA L4 GPU, 24GB of GPU Memory

Lenovo Legion Pro 5
- AMD Ryzen 9, 32 GB
- 1 NVIDIA RTX 5060, 8GB of GPU Memory

Lenovo Ideapad 3
- Intel Core i7-12650H, 16GB
- NVIDIA RTX 3050, 8GB of GPU Memory

IMPLEMENTATION: KEY FRAME MLP MODEL

Introduction

Normally in the video of a language sign there are only one or few key frames/images. The rest only add confusion. So, we decided to try to train an MLP to infer the sign from each frame instead of all the video. Therefore, the model/algorithm begins to loop through the csv frames to find the frame that gets the maximum probability of some sign at its output. The results of the trial were positive enough.

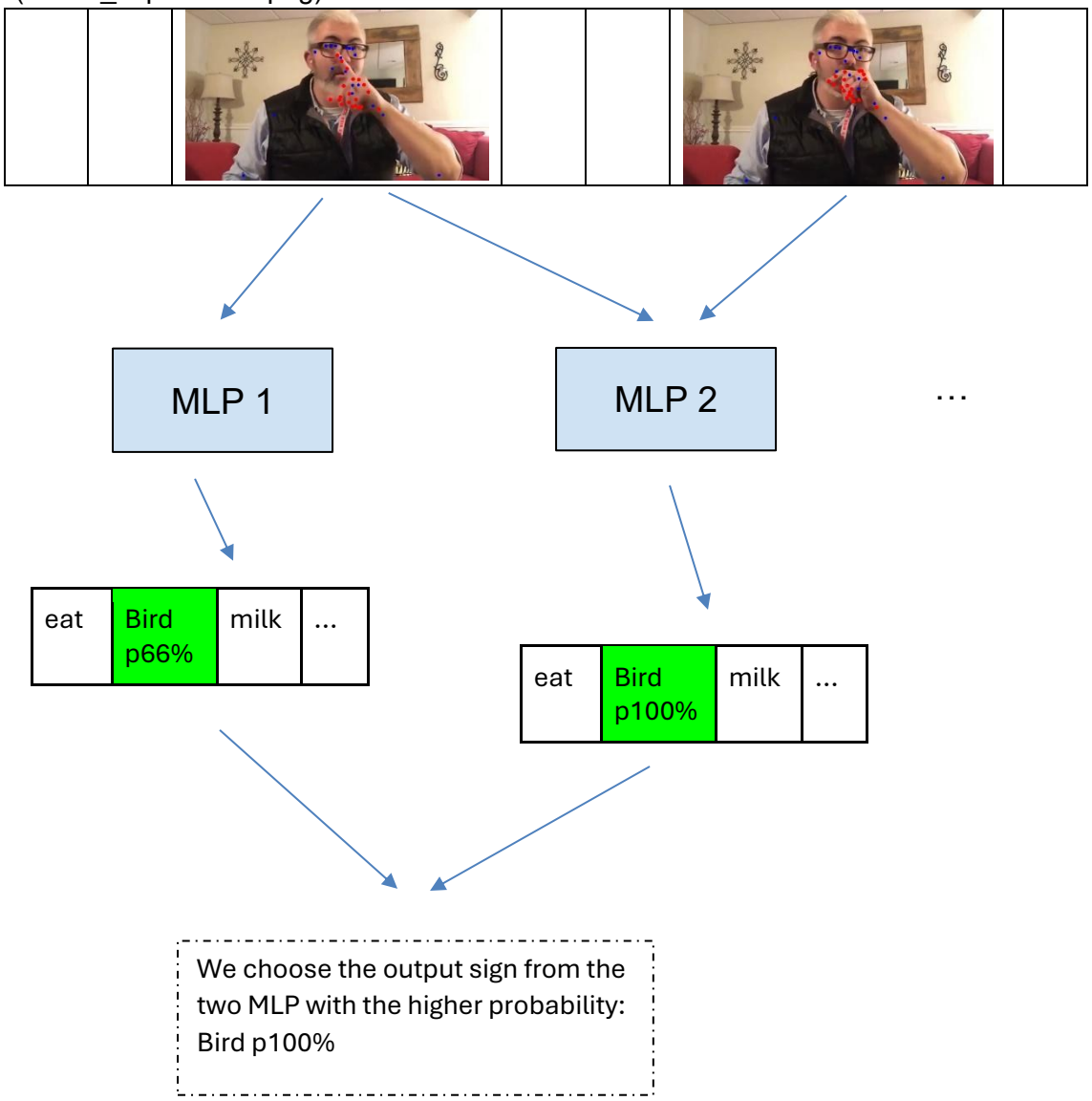
So next, we decided that with a second key frame it would be more probable to recognize the signs. So, we added a second MLP that at its input gets the key frame selected by the first MLP plus another frame from the csv/video. So, this second MLP also loops through the csv frames and we get the maximum probability/sign at its output.

Therefore, the first MLP must be trained first and the second afterwards. We could add more MLP's to get more key frames in this way. The only drawback is that the model needs some more time to train/infer.

Graphically it would be like this:

Video => CSV (with mediapipe coordinates) => Preprocess/normalization with angle extraction, speeds...

*(model_explanation.png)



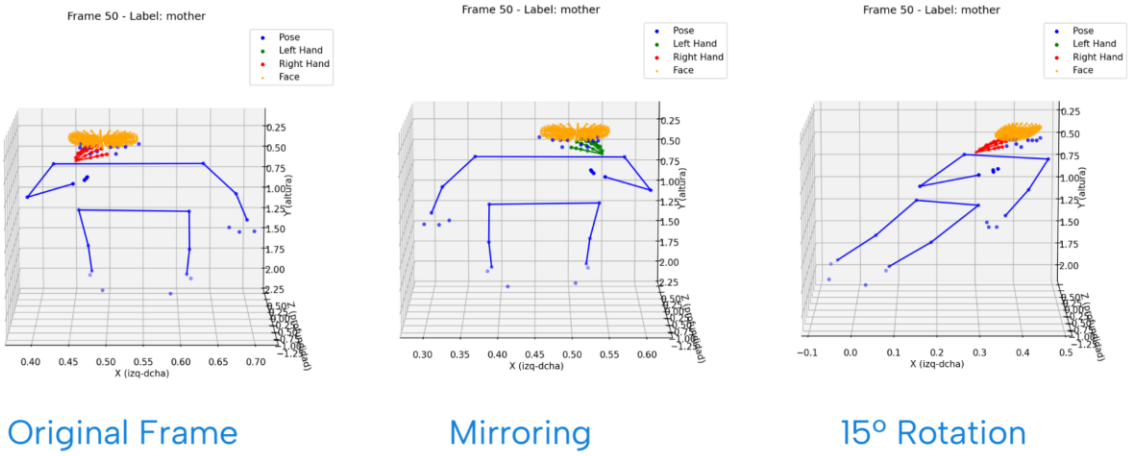
Also, to get some of the gesture dynamics we compute the change/increment between consecutive frames in the input features in the preprocess. Also, when concatenating the input frames of the MLP 2 we use the same order as de number/position of the frame.

MLP is a simple but a good model to recognize static frames... Also, with the preprocess / normalization with angles... we do before it's easier for the MLP to do its job.

Key Features

Data augmentation

We decided in this model to use the “mirror” data augmentation because it doubles the number of samples we have for the training and lets us to manage the problem with the left- and right-handed people. Also, in the MS-ASL there are left-handed signers.



*(data_augmentation.png)

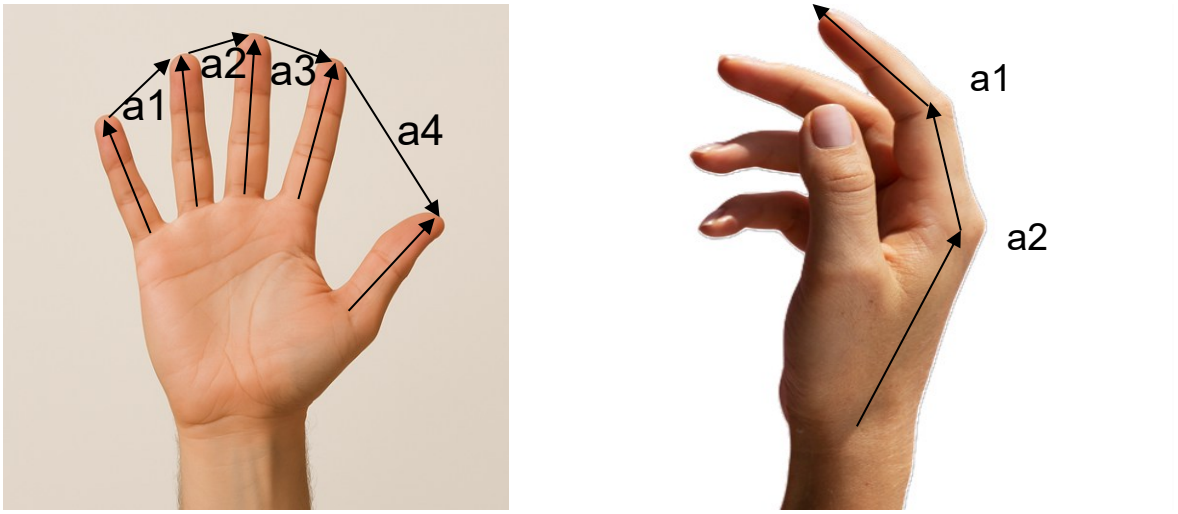
Preprocess

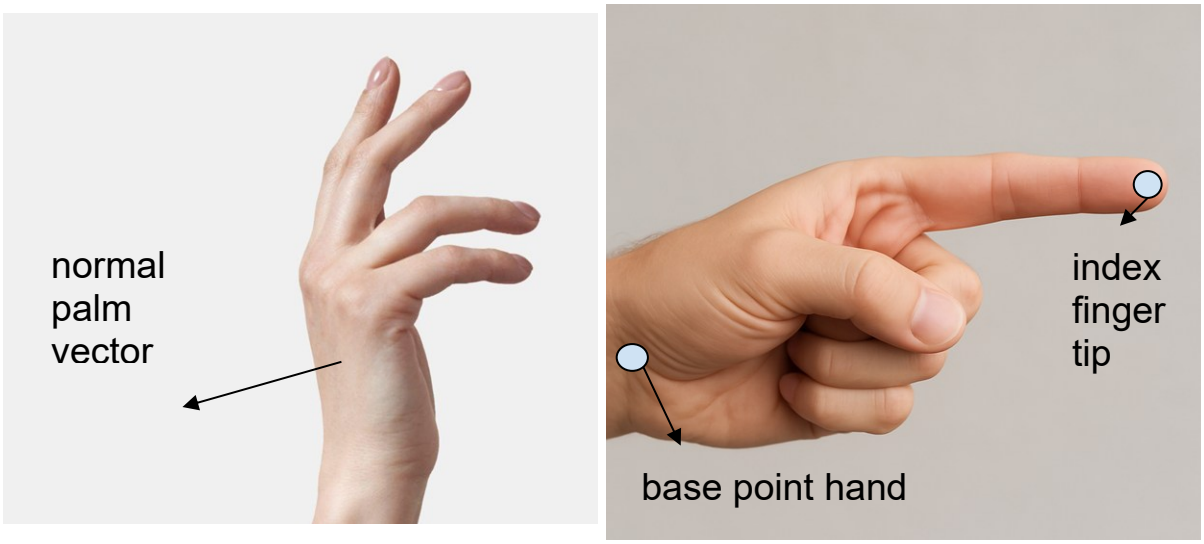
In this model we decided to try extracting the angles and only certain important coordinates (as the hands base or the index fingertip) to normalize the features and ease the work from the model. Also, it permits to reduce the number of inputs to the MLP’s.

Also, we normalize the few coordinates with the shoulder landmarks from the “pose” from mediapipe. So, we become independent from the size or distance of the sign from the camera.

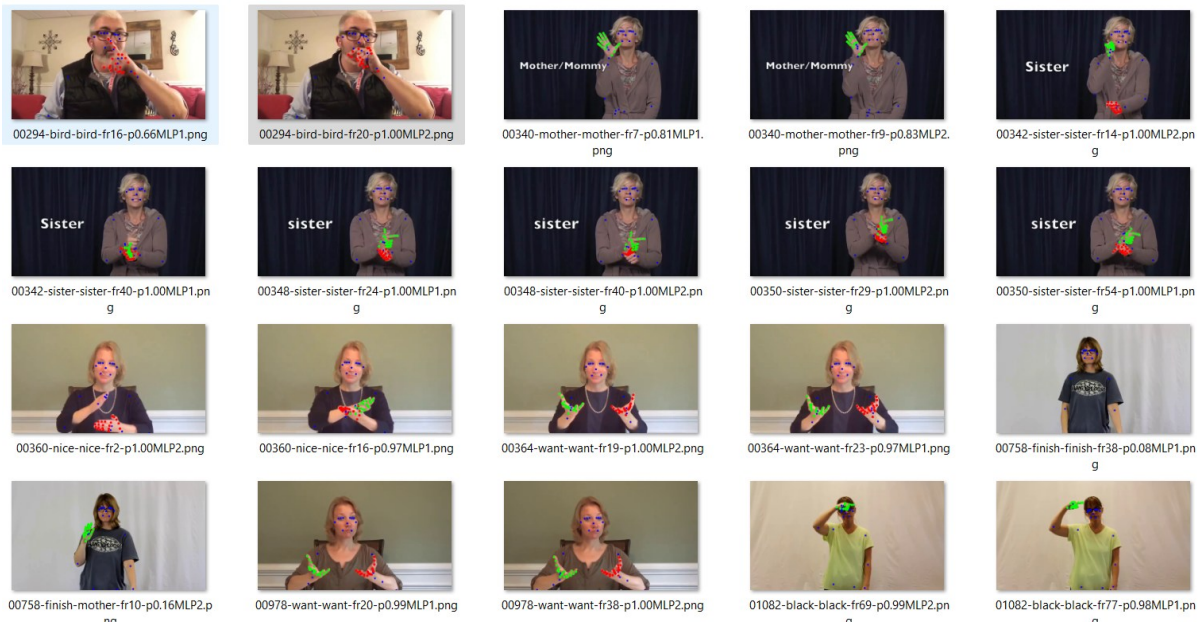
In the end we didn’t use the “face” landmarks from mediapipe because there were too many and we could use the pose landmarks from the face.

*(hand_angles.png)





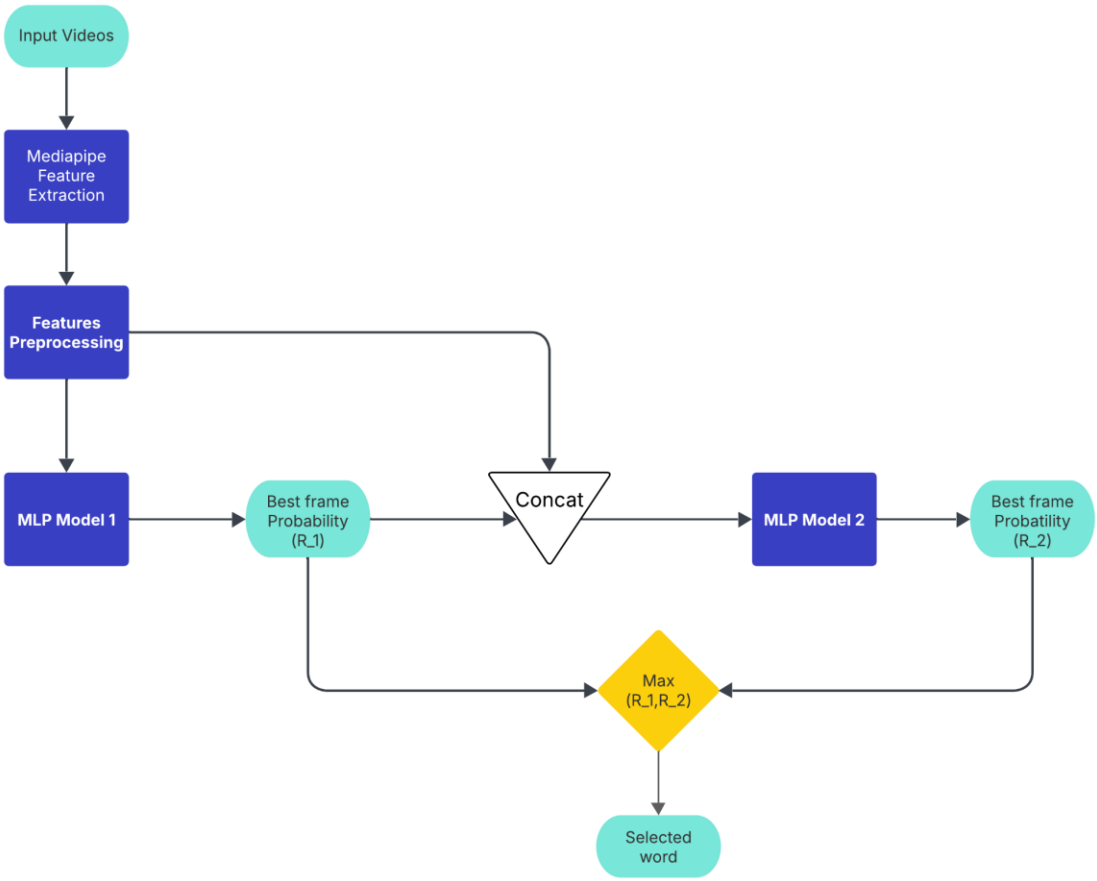
Here we have some graphic results from the test to see how it works. For each test file we can see the two key frames selected by MLP1 and MLP2, and its probability (0-1). (Internally the model works with preprocessed features, not images or normal mediapipe coordinates)



*(mlp1_mlp2_examples.png)

It can be seen as the MLP2 helps to improve significantly the result of the MLP1 alone (increasing the probability or correcting the MLP1 sign)

Model Explanation



*(model_algorith.png)

1. **Video Input:** The model starts by taking a set of **videos** as its initial input.
2. **MediaPipe Preprocessing:** Each **frame** from these videos is individually processed using **MediaPipe**. This step extracts relevant features (e.g., hand, body, or face landmarks), which form the initial tensors for each frame.
3. **Features Preprocessing.** Data augmentation and normalization.
4. **MLP 1 Module (Best Frame Selection):**
    - A first **Multi-Layer Perceptron (MLP 1)** takes the extracted frame features (and potentially a representation of the word being searched for) as input.
    - Its role is to identify the **"best frame"** within a video that most accurately represents the given word.
    - This MLP 1 outputs **the best frame** and a **"Best Result" (R_1)** associated with its selection.
5. **Concatenation and MLP 2 Processing:**
    - For *each individual frame's features* (from MediaPipe), the **representation of the "best frame"** (determined by MLP 1) is concatenated.

- The **concatenation is temporal**: it's added to the front of the current frame's tensor if the current frame appears *before* the "best frame," or to the back if it appears *after*. This provides temporal context to the model.
- These **modified (concatenated) tensors** are then fed into a second Multi-Layer Perceptron (MLP 2).
- MLP 2 processes each modified tensor, producing a **result (r_i)** for each frame.
- From all the results generated by MLP 2 for each frame, the **maximum value (R_2)** is selected, representing the best result that MLP 2 found across all frames.

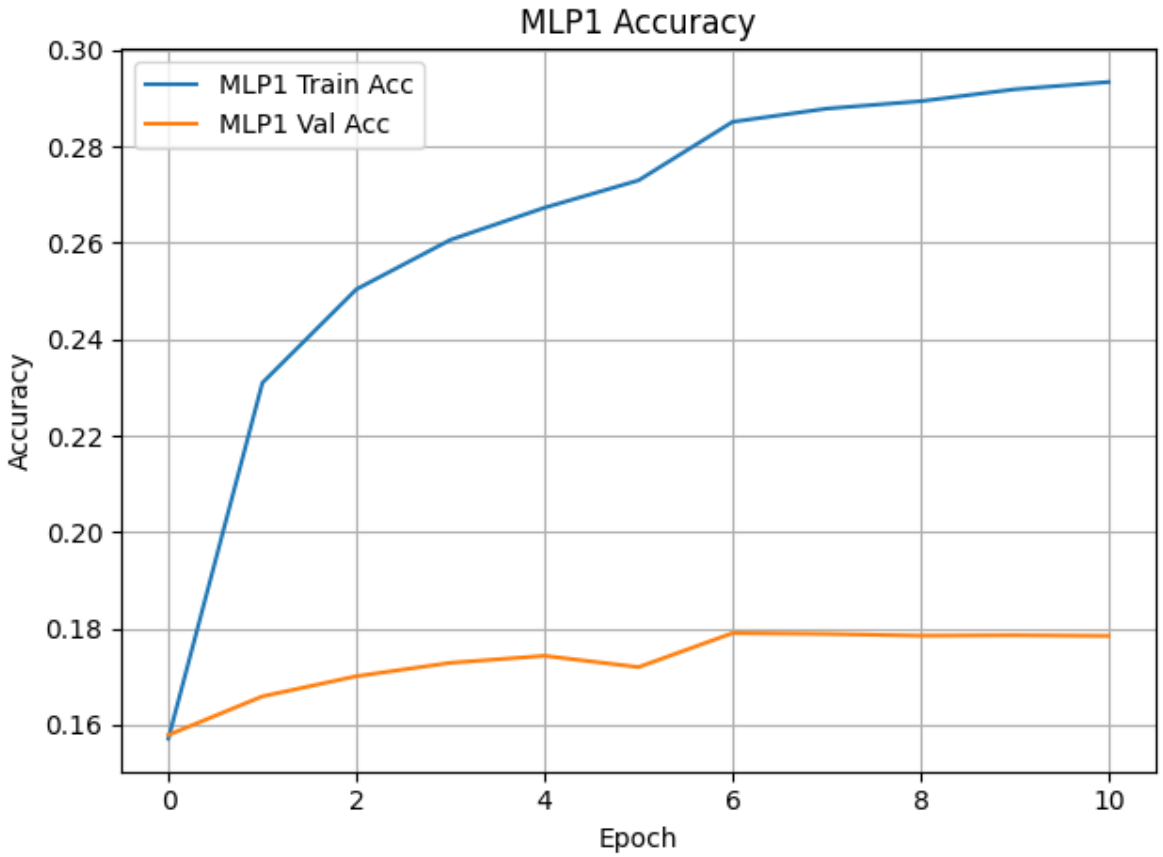
6. **Final Output:**
   - Finally, the system compares the **"Best Result" (R_1)** obtained directly from MLP 1 with the **"Best Result" (R_2)** obtained from MLP 2 (the maximum among the per-frame results).
   - The **final output of the model** is the maximum value between R_1 and R_2.

Results

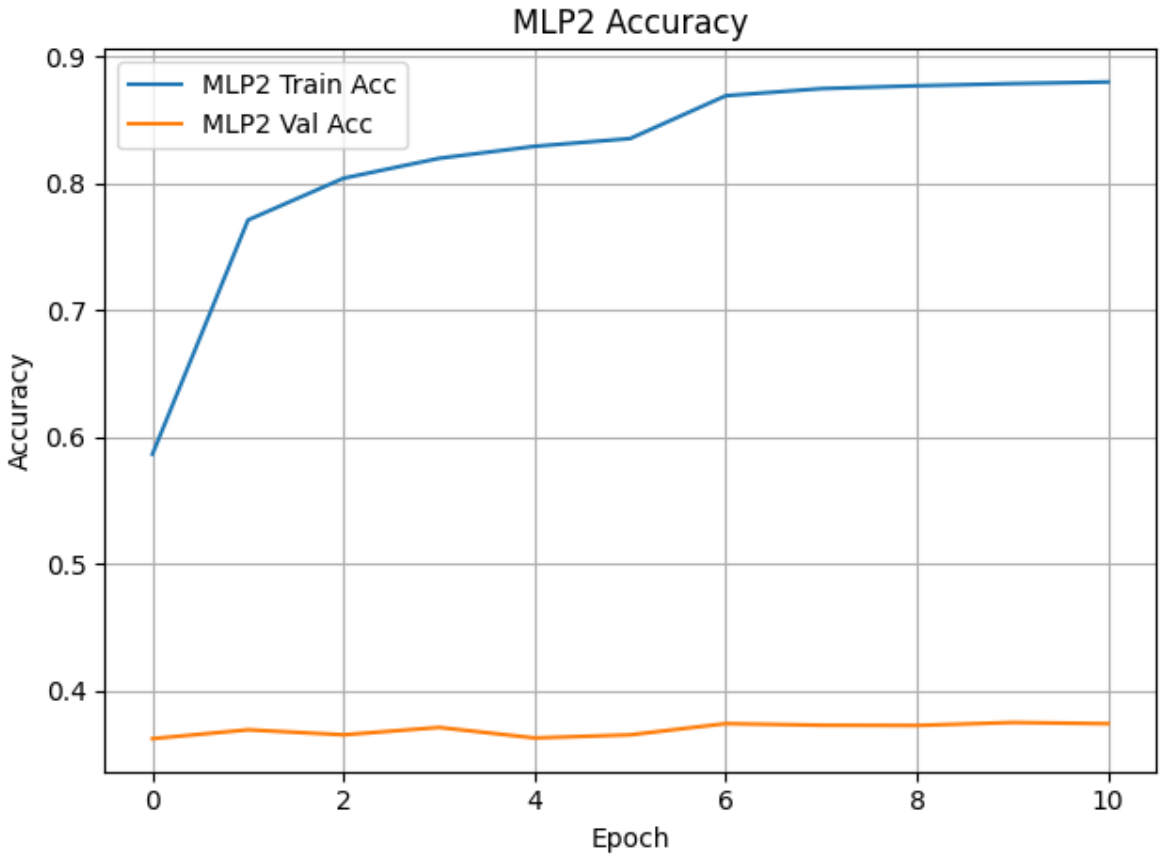
The **test accuracies** we got are (top-1):

	MLP 1 (overall)	**MLP1 + MLP2 (overall)**	**MLP + MLP2 (average/class)**
MS-ASL 100	82,18%	91,38%	91.83%
MS-ASL 200	75,80%	85,35%	86.98%
MS-ASL 500	56,53%	76,38%	75,21%
MS-ASL 1000	44,75%	64,91%	61,35%

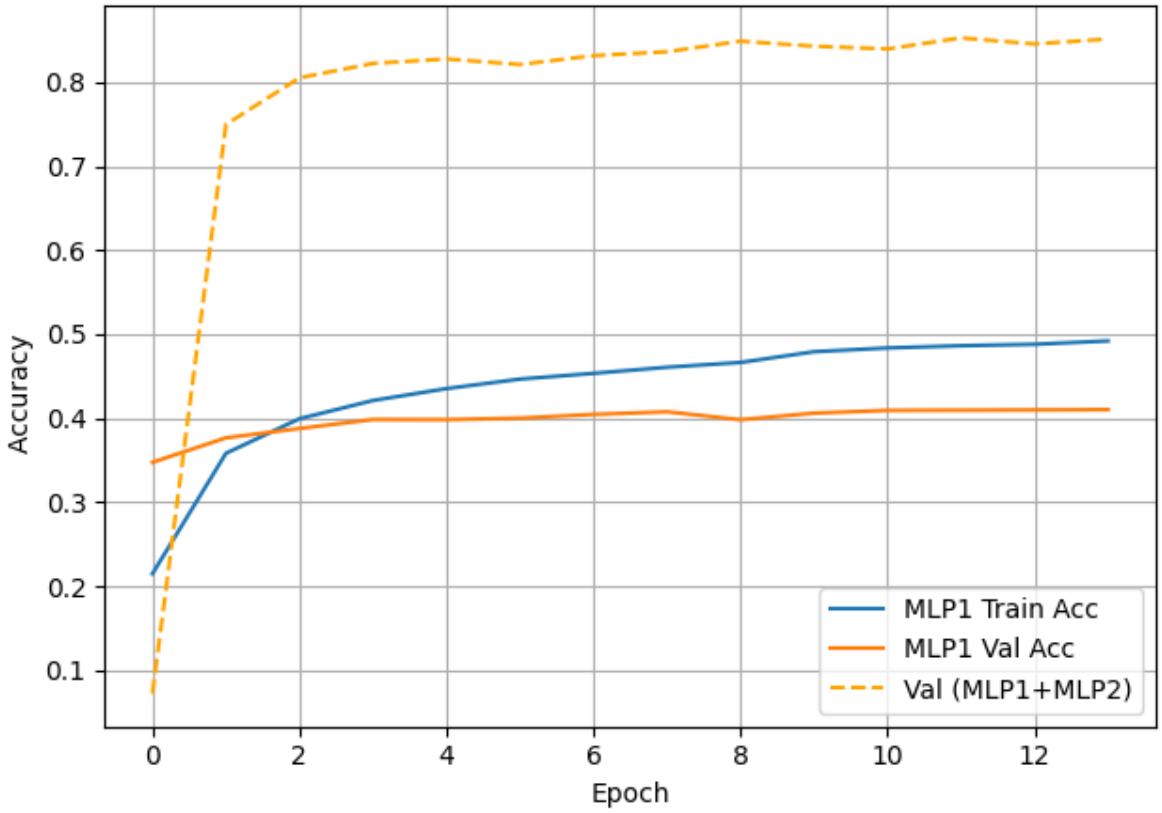
Because our model can be seen as an algorithm that uses two MLP's models and searches for the best key frames, it is normal that the next graphics are not very good (especially for MLP1), but a major part of frames do not have a clear class/sign winner (with a high probability).



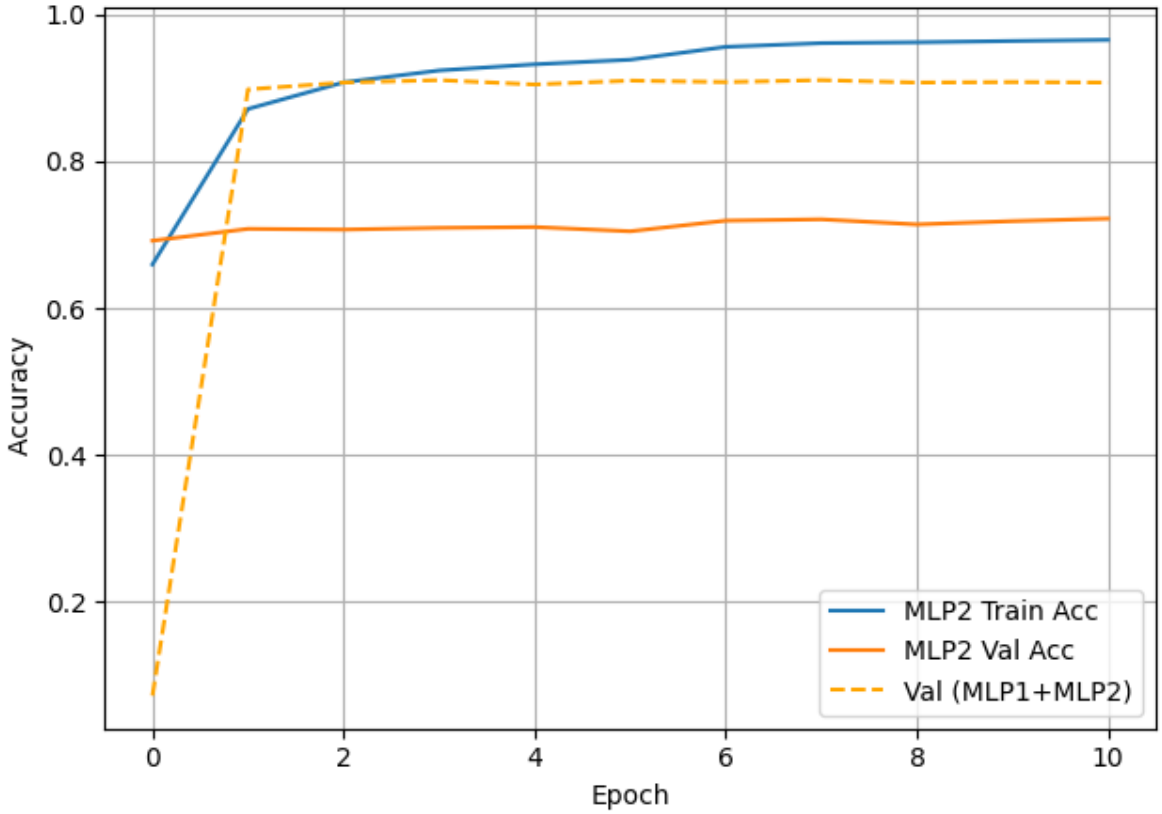
*(key_frame_acc_1.png)



*(key_frame_acc_2.png)

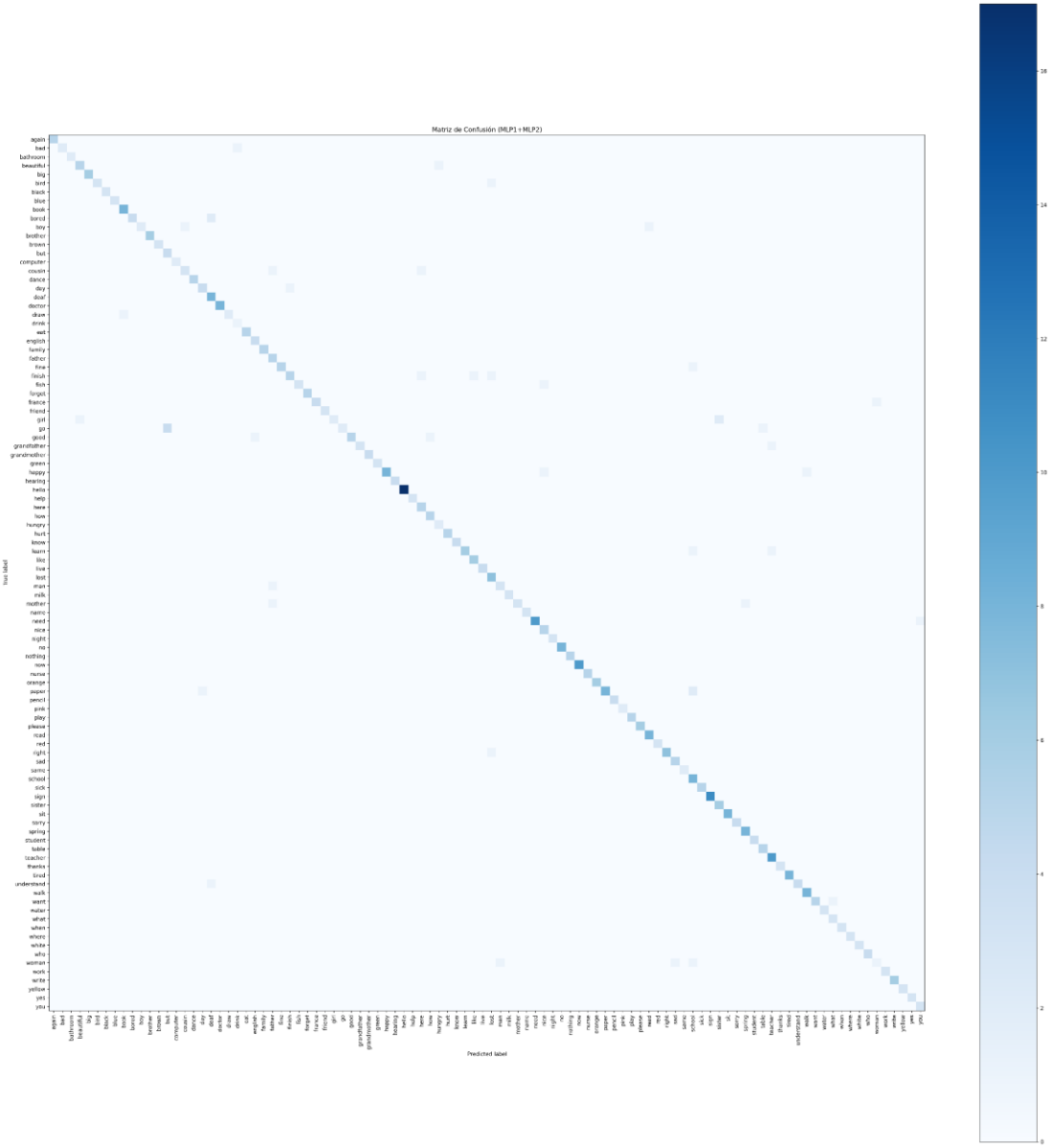


*(key_frame_acc_3.png)



*(key_frame_acc_4.png)

As an example, we show confusion matrix for MSASL 100.



*(key_frame_conf_mat.png)

State of the art

To be able to compare our model results we have searched for the current “state of the art” for the ASL recognition using MS-ASL 1000 and Mediapipe only (like our model). However, there are not many serious articles with result metrics useful for comparation with our model.

At the end we could compare with the original article of the MS-ASL dataset “Vaezi Joze, H. R., & Koller, O. (2018). MS-ASL: A large-scale data set and benchmark for understanding American Sign Language. *arXiv preprint arXiv:1812.01053*”. In this article they publish these top-1 accuracy results:

Method	ASL100	ASL200	ASL500	ASL1000
Naive Classifier	0.99	0.50	0.21	0.11
VGG+LSTM [17, 18]	13.33	7.56	1.47	-
HCN [77]	46.08	35.85	21.45	15.49
Re-Sign [39]	45.45	43.22	27.94	14.69
I3D [10]	**81.76**	**81.97**	**72.50**	**57.69**

Table 3: The average per class accuracy for baseline method on proposed ASL data sets.

The HCN (hierarchical co-occurrence network) model they use with body Key-points is the most similar to our mediapipe model to compare, although our model is also competitive against the I3D model referred here.

Also, we found the article “Boháček, M., & Hrúž, M. (2022). *Sign pose-based transformer for word-level sign language recognition*. In **Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) Workshops** (pp. 182–191). IEEE.”, although they use the WLASL dataset:

Model	**App.**	**Pose**	**Backbone**	**WLASL100**	**WLASL300**
I3D (baseline) [21]	✓	✗	✓	65.89	56.14
TK-3D ConvNet [22]	✓	✗	✓	**77.55**	**68.75**
Fusion-3 [13]	✓	✗	✓	75.67	68.30
GCN-BERT [36]	✗	✓	✗	60.15	42.18
Pose-TGCN [21]	✗	✓	✗	55.43	38.32
Pose-GRU [21]	✗	✓	✗	46.51	33.68
SPOTER (Ours)	✗	✓	✗	**63.18**	**43.78**

Table 4. Top-1 macro average recognition accuracy achieved by each model (by row) on the WLASL100 and WLASL300 subsets. *App.* denotes the usage of appearance representation (i.e. direct frame images) as the input to the model, *Pose* column then marks the usage of skeletal data as inputs. All the entries used data augmentation techniques.

Model Conclusion

This Key frame MLP model, as we named it, has become the model with the best test accuracy of all our trials. It came from a simple idea and using only MLP's, that on the other hand are good for static images and frame recognition. Some other models have "attention" for giving more importance to some "tokens". Our model does something similar but looking at each frame to find the best key frames.

Also, it has the advantage that it does not normalize the number of frames, so it does not lose any frame and no frame is changed in some way. Our model adapts without problem to any video length.

Another feature of our model is that it is designed also to get dynamic information of the mediapipe. It preprocesses and adds the speed or increment of features between each two consecutive frames. Also, the concatenation of the frames in the input of MLP2 is done with the same order of the frame number/position.

Furthermore, we have done data augmentation mirroring the videos/csv's, so our model is trained well for left-handed people also.

For now, we have tried two MLP's and we have got very good results. We could try in the future with a third MLP…

Perhaps because the MS-ASL dataset does not have a lot of samples per sign, our model with MLP's does better than other more complex.

DEAD END EXPERIMENTS

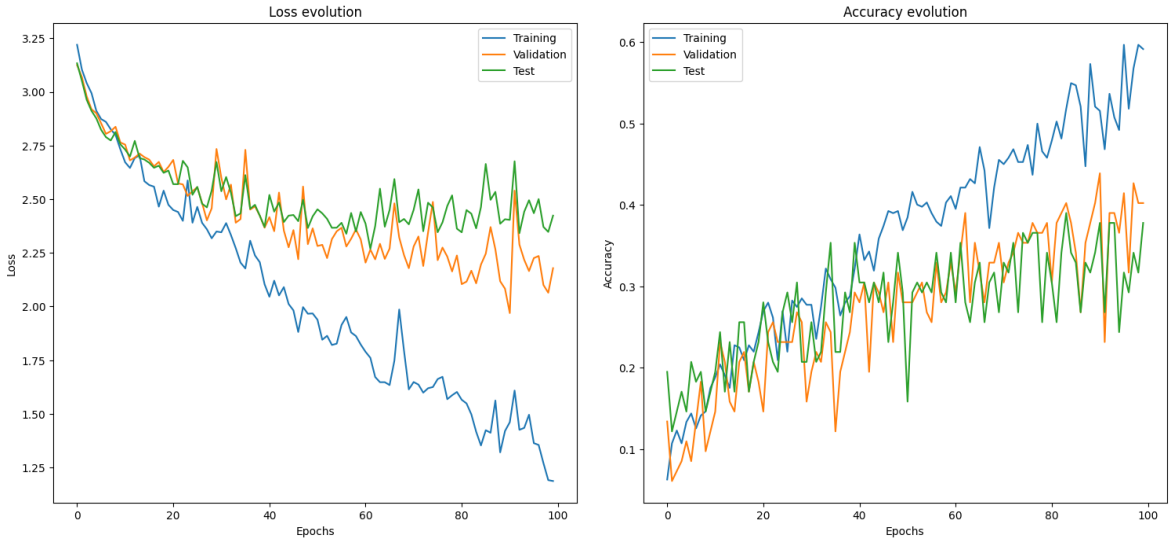
LSTM

Hypothesis: The hypothesis is that an LSTM-based model can effectively classify isolated American Sign Language (ASL) signs from the MSL dataset, even with a limited number of short video samples per class. The temporal nature of the data should allow LSTM to capture the sequential movement patterns characteristic of different signs.

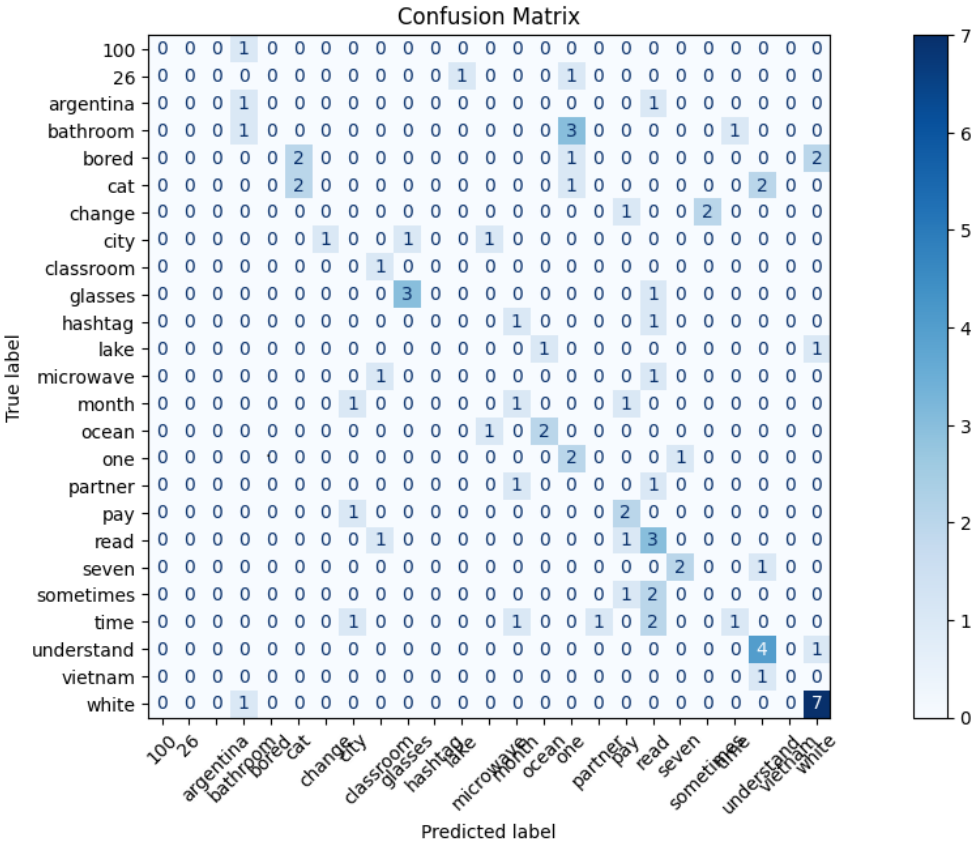
Experiment setup: The experiment consists of training a Long Short-Term Memory (LSTM) neural network to classify 25 different ASL signs from the MSL (Multimodal Sign Language) isolated dataset. Each video in the dataset represents a single sign and is represented by 150 frames of hand landmarks (21 key points per hand, 42 values per frame using only x and y coordinates).
The architecture of the model includes: input layer, LSTM layers to capture temporal dependencies and Dense layers for classification into 25 classes.

The dataset is imbalanced and contains a relatively small number of samples per class, which was a key variable tested in this experiment.

Results: The LSTM model achieved an accuracy of less than 30%, which is significantly below an acceptable threshold for practical use. Analysis of the confusion matrix reveals a high rate of misclassification, particularly among signs with visually similar motion patterns or low sample count. Overfitting was observed during training, with training accuracy much higher than validation accuracy.



*(lstm_accuracy.png)



*(lstm_conf_matrix.png)

Conclusion: The results suggest that the current dataset is insufficient in size and diversity to train a reliable LSTM model for ASL sign classification. The poor performance highlights the need for either:

- Augmenting the dataset to balance and increase the number of samples per class. Maybe data augmentation or transfer learning techniques could mitigate the effects of data scarcity
- Trying alternative architectures (e.g., attention-based Transformers or CNN-LSTM hybrids) that may generalize better
- Pretraining on a larger sign language dataset before fine-tuning on MSL

LSTM + Data Augmentation

Hypothesis: The hypothesis is that applying data augmentation techniques such as shifting, scaling, and adding noise to the original MSL dataset would effectively increase the training set size and diversity, allowing the LSTM model to generalize better and achieve higher classification accuracy.

Experiment setup: This experiment used the same LSTM architecture as in previous tests to classify 25 ASL signs from the MSL dataset. Since the original dataset had too few unevenly

distributed samples per class, data augmentation was applied to artificially increase and **balance** the dataset.

The final dataset was constructed with a fixed number of samples per class: 140 training samples, 30 validation samples and 30 test samples.

The augmentation techniques applied to generate new samples included:

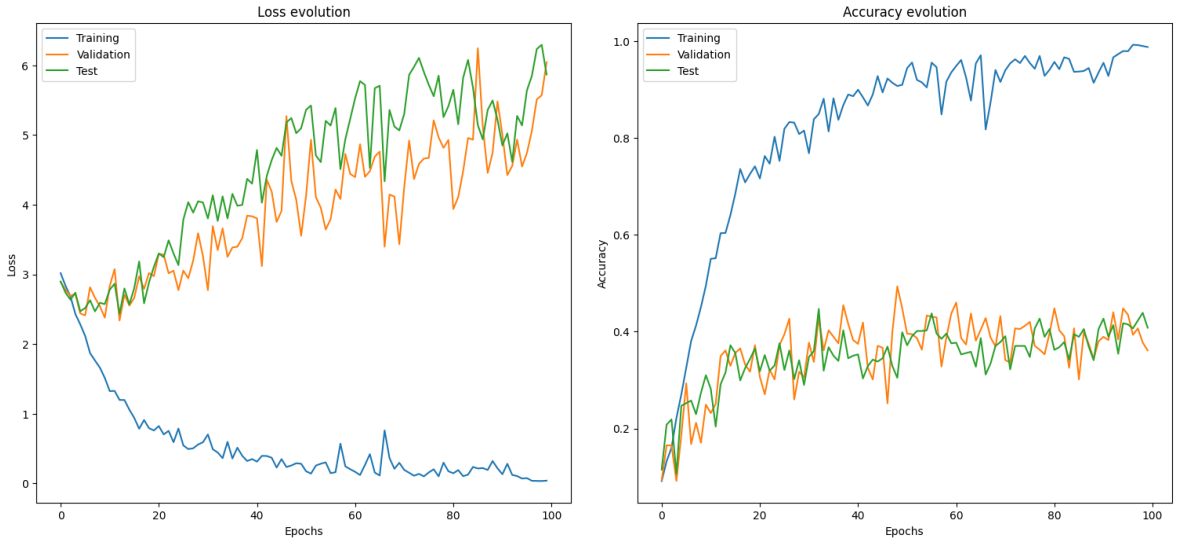
- **Shift**: small translations in the hand landmark positions
- **Scale**: zoom in/out effects on the coordinate space
- **Noise**: random Gaussian noise added to landmark values. Each video sequence was standardized to 90 frames of 42 values (21 key points per hand, x and y only).

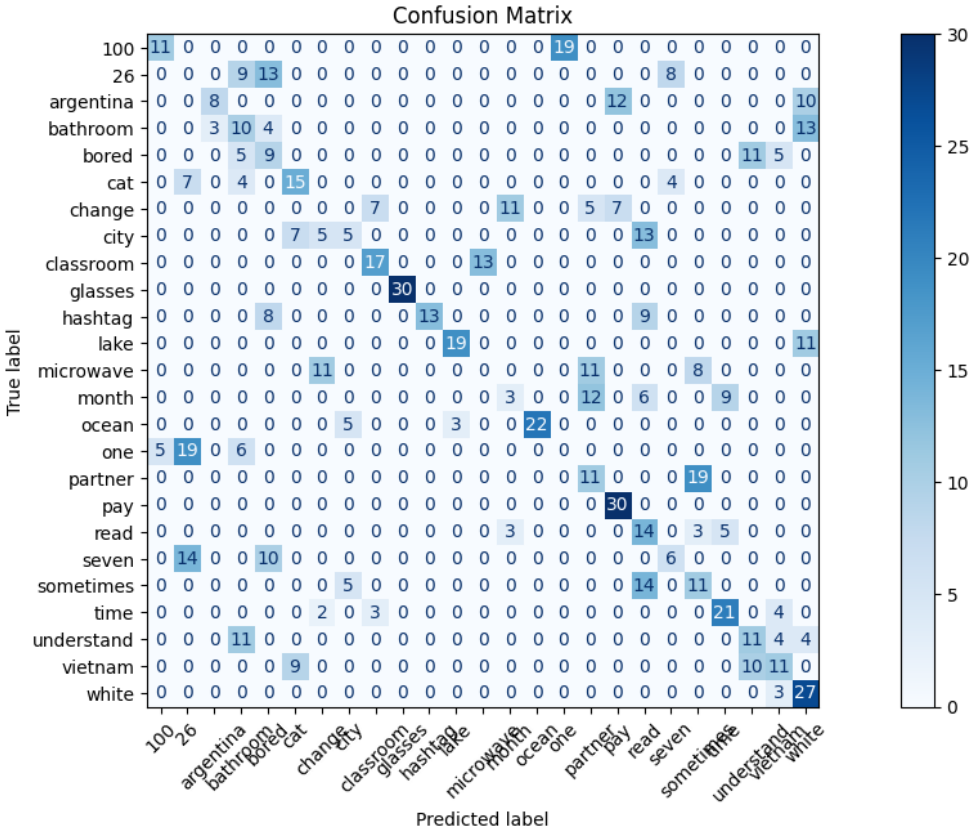
Results: Despite the increased dataset size and the effort to balance the number of samples per class, the model achieved only about **34% accuracy**, which is only a minor improvement over the 30% obtained using the original unbalanced dataset.

The confusion matrix showed that many classes were still frequently misclassified, suggesting that the model struggled to differentiate between signs even with more training data. Additionally, the learning curves displayed a high variance between training and validation accuracy, indicating possible overfitting or poor generalization.

One likely cause is the **nature of the data augmentation itself**: in particular, the addition of random noise to the hand landmark coordinates may have **introduced unrealistic or ambiguous gestures**, effectively confusing the model rather than helping it learn robust patterns. This could explain why the model failed to achieve the expected improvement in performance despite the larger and balanced dataset.

*(lstm_plus_da_accuracy.png)





*(lstm_plus_da_conf_matrix.png

Conclusions: Despite equalizing the number of samples per class, the accuracy remained low at around 34%. This indicates that **the type of augmentation is critical**: methods such as shifting, scaling, and adding noise may alter the temporal dynamics or spatial structure in ways that reduce gesture clarity, making them ineffective or even harmful for training LSTM models.
Additionally, this result reinforces the importance of collecting more real samples or exploring **transfer learning** approaches instead of relying solely on synthetic data generation to balance the dataset.

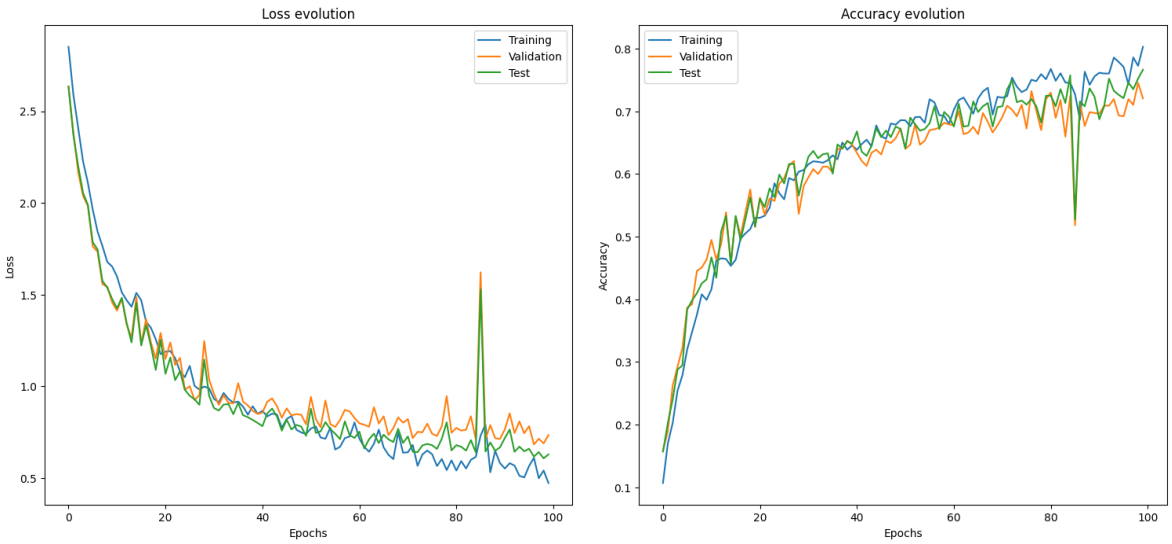
LSTM + Greek Dataset

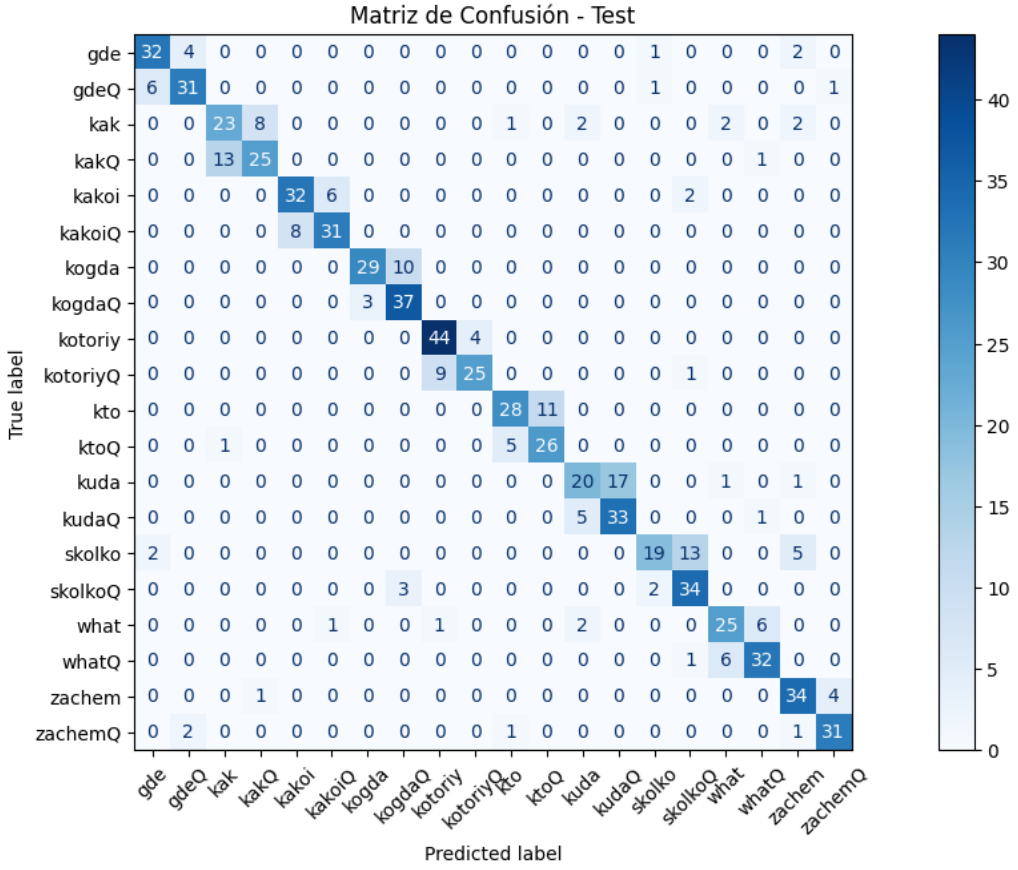
Hypothesis: The hypothesis is that an LSTM model trained on a dataset with a sufficient number of video samples per class (around 250) will achieve significantly better classification accuracy than when trained on a smaller dataset. This would demonstrate the importance of dataset size and balance in sequence-based sign language recognition tasks. In this experiment it is used the Greek Sign Language (GSL) dataset, which is a large RGB+D video dataset for Sign Language Recognition, recorded with 7 signers and annotated by experts. It includes 10,290 sentence instances and 310 unique classes. A subset with 20 isolated sign classes is available and was used in this case. The dataset provides RGB and depth data captured at 30 fps with 848×480 resolution.

Experiment setup: This experiment uses an LSTM neural network to classify 20 isolated signs from the Greek Sign Language (GSL) dataset. Each video sample is preprocessed into 90 frames of hand landmark coordinates (42 values per frame, using x and y for 21 points per hand).

Results: The LSTM model achieved an accuracy close to 80% on the GSL dataset. This is a substantial improvement over the 30% accuracy obtained on the MSL dataset. The model shows consistent generalization across classes, with lower confusion rates and a more balanced precision and recall profile.

*(lstm_greek_accuracy.png

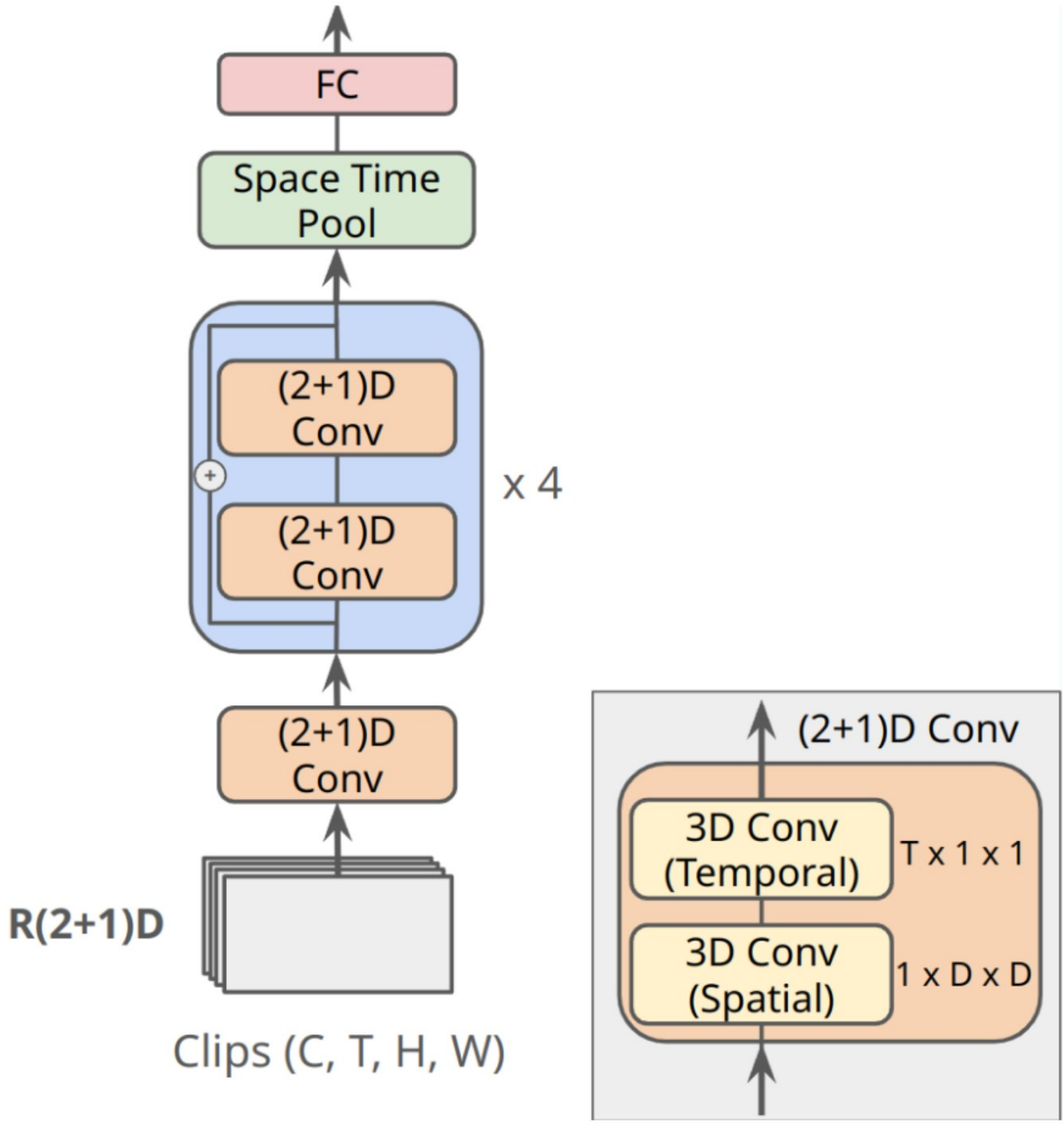




*(lstm_greek_conf_matrix.png)

Conclusions: These results confirm the hypothesis that the number of samples per class significantly influences model performance. With a more extensive and balanced dataset, the LSTM model can effectively learn the temporal patterns of different signs.
This comparison highlights a key limitation of the MSL dataset: its small number of videos per class. As a result, a new hypothesis emerges: increasing the volume and class balance of MSL through data augmentation or further collection will lead to a performance boost comparable to GSL.
Additionally, the success of the LSTM on GSL suggests that the architecture is suitable for the task, and that future improvements should prioritize data quantity and diversity rather than model complexity alone.

ResNet 2+1



*(resnet2_plus_1.png)

Hypothesis: The R(2+1)D network enhances video understanding by improving traditional 3D CNNs. It decomposes 3D convolutions into two simpler operations:

1. 2D Spatial Convolution: Extracts spatial features (height and width) within each frame.
2. 1D Temporal Convolution: Captures motion dynamics across frames (time).

This decomposition introduces non-linearity between spatial and temporal processing, improving learning efficiency. Residual connections further enhance the model by allowing shortcut paths, enabling stable training and effective feature propagation even in deeper networks.

The network follows a ResNet-inspired architecture, organizing layers into blocks to capture spatial and temporal features hierarchically. It processes video tensors of shape (batch size, channels, sequence size, height, width) and outputs action class predictions, making it efficient and powerful for video recognition tasks.

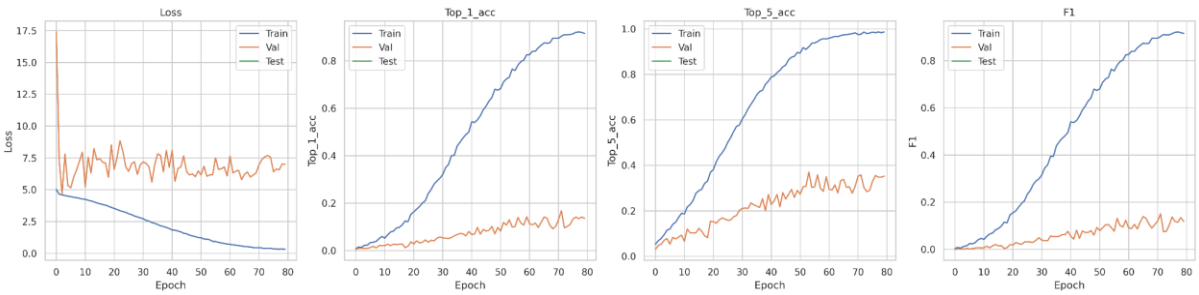
For detailed specifications, refer to the original paper.

Experiment setup:

- 101 words with more training samples
- frame size 224x224
- 30 frames per clip and
- 80 epochs

This experiment uses an LSTM neural network to classify 20 isolated signs from the Greek Sign Language (GSL) dataset. Each video sample is preprocessed into 90 frames of hand landmark coordinates (42 values per frame, using x and y for 21 points per hand).

Results: With a Top 1 overall of less than 20% we concluded this was not the right path.



*(resnet2_plus_1_accuracy.png)

Conclusions: These results confirm the hypothesis that the number of samples per class significantly influences model performance.

The Spoter Transformer

Hypothesis: SPOTER (Sign Pose-based Transformer for Recognition) is a Transformer-based model specifically designed for **spotting and recognizing isolated or continuous sign language gestures**. It leverages **pose data**, primarily hand landmarks, to recognize ASL (or other sign languages) more efficiently and with less reliance on raw video frames. SPOTER focuses on interpreting the **temporal evolution** of sign gestures, making it well-suited for spotting signs from continuous motion.

Key Concepts:

1. **Pose-Based Input**

   SPOTER uses **pose landmarks** extracted from signers' hands (and sometimes body and face) instead of raw images or video. This reduces noise and computational load while retaining essential gesture information.

2. **Transformer Architecture**

   The model employs the **Transformer** architecture, originally introduced in the "Attention Is All You Need" paper. Transformers use **self-attention** to model dependencies across time, allowing SPOTER to capture complex temporal relationships in hand motion data.

3. **Sign Spotting**

   Unlike general classifiers, SPOTER includes a spotting mechanism that **detects where in a sequence a sign occurs**, even in continuous video. It can distinguish between signing and non-signing motion.

We can find a deeper explanation in the paper Sign Pose-Based Transformer for Word-Level Sign Language Recognition.

Advantages of SPOTER

- **Efficiency**: Pose-based input is lightweight and efficient compared to processing full video frames.
- **Interpretability**: Using human pose landmarks makes the system easier to interpret and debug.

- **Temporal Attention**: The Transformer's attention mechanism captures subtle time-based patterns in sign gestures.
- **Scalability**: Works for isolated sign recognition and spotting in continuous sequences.

Challenges and Limitations

- **Pose Extraction Errors**: Inaccurate or missing pose data can affect model performance.
- **Data Requirements**: Transformers require large datasets to generalize well.
- **Lack of Context**: Pure pose-based input may miss facial expressions or contextual nuances important in sign language.

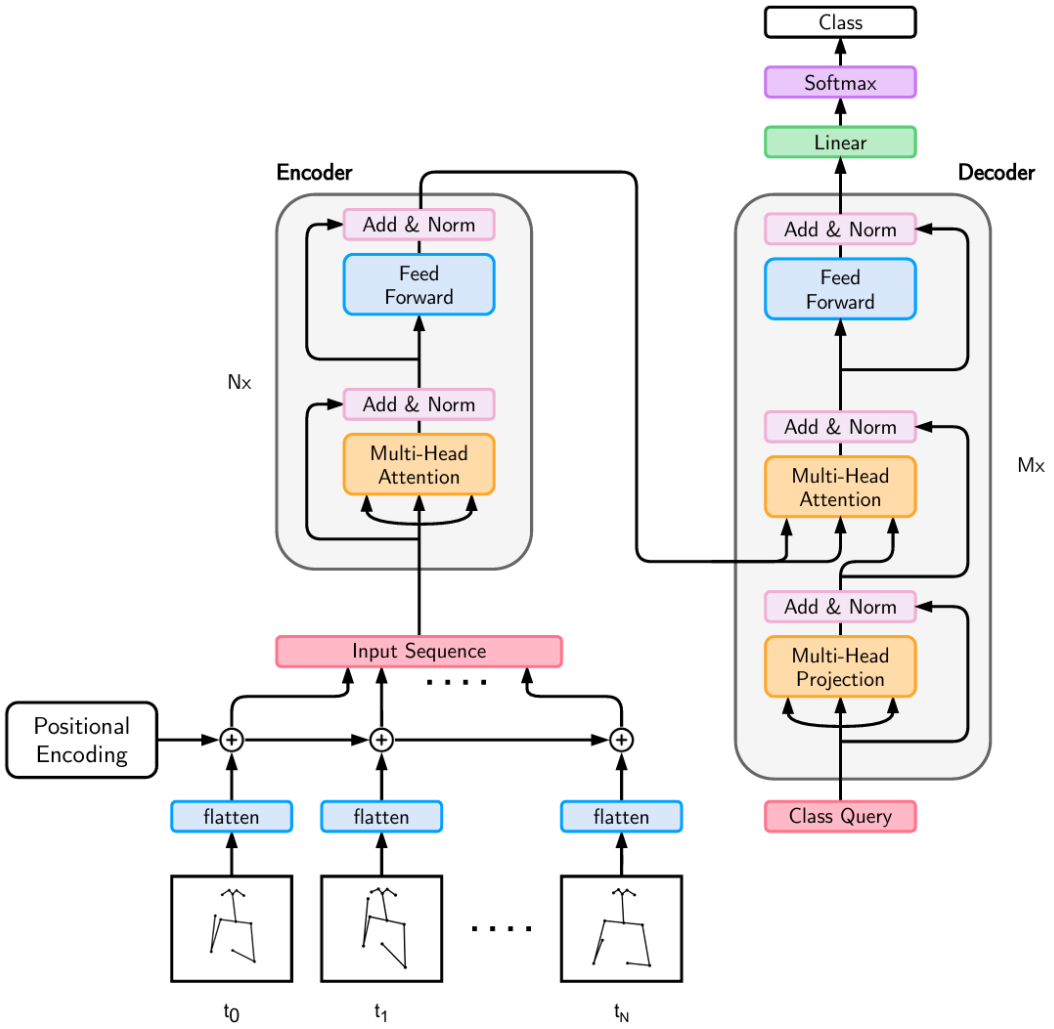
Experiment setup:

Data augmentation:

We used rotation through the z axis to the left and to the right (15 degrees). So we get three times more samples.

Data preprocess:

We calculated the angles between fingers and for each finger, and some important coordinates from the base of the hands and some palm angles to help the Spoter model job. Also, we normalized respect to the shoulder landmarks.



*(spoter_transformer.png)

1. **Input Representation**
    - Input to the model is a sequence of **pose vectors** representing the key points (e.g., from MediaPipe or OpenPose).
    - Each frame is a vector of 2D or 3D coordinates: $frame\_t = [x_1, y_1, x_2, y_2, ..., x_n, y_n]$ for n key points.

2. **Positional Encoding**
3. Since Transformers lack inherent temporal order, **positional encodings** are added to the pose sequence to retain the notion of time.

4. **Encoder Layers**
   The pose sequence is passed through multiple Transformer **encoder layers**, each consisting of:
    - Multi-head self-attention
    - Layer normalization

- Feed-forward networks

5. **Sign Classification Head**

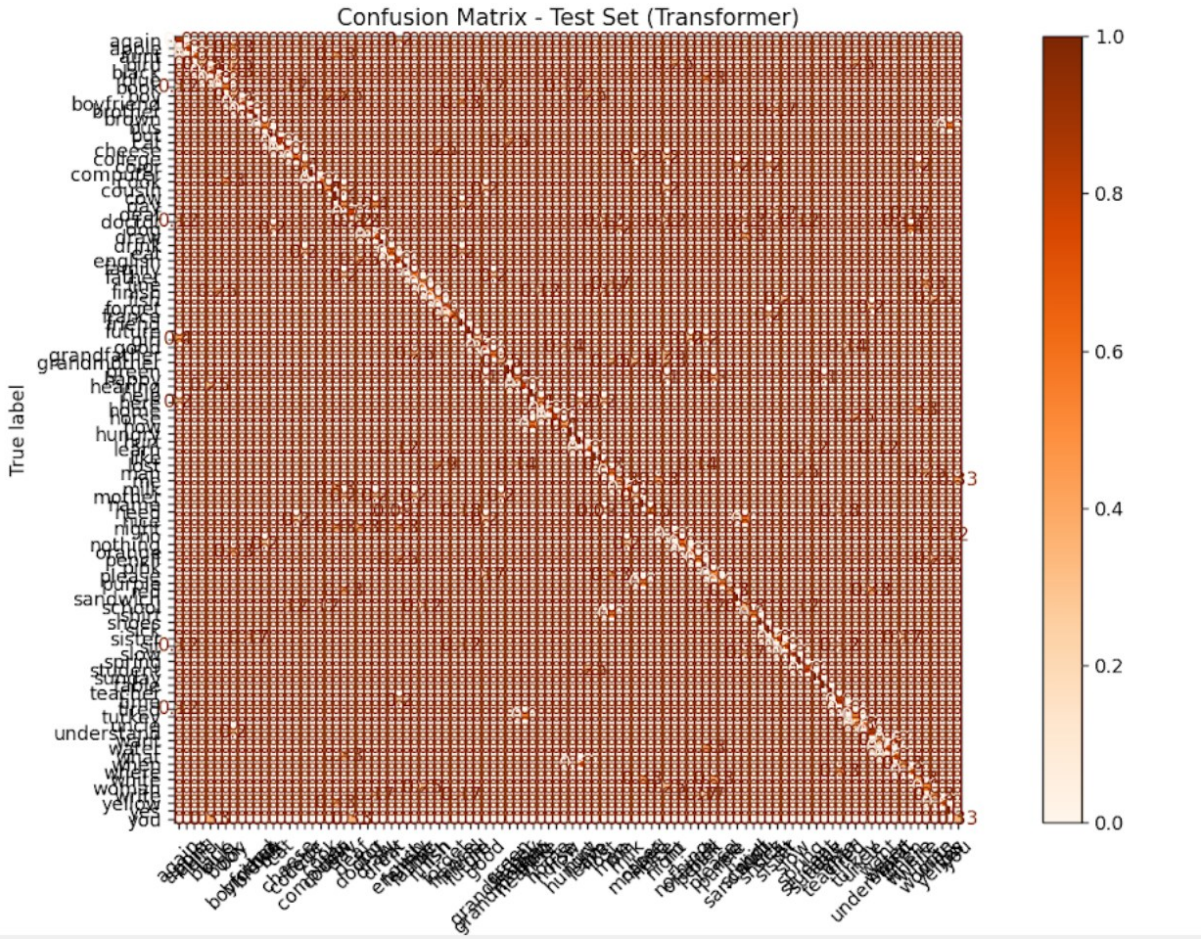
   After encoding the entire sequence, a **classification head** (often a fully connected layer or softmax) is used to predict the **sign class** or determine whether a sign is present.

6. **Spotting Module**

   - A separate module or token (like a `[SIGN]` token) may be used to **indicate where in the sequence the sign occurs**.
   - The model can be trained with **start-end annotations** or weak supervision (e.g., only knowing which sign appears, but not where).

Results:

For 100 classes Test overall Accuracy (video): 62.25%



*(spoter_transformer_conf_matrix.png)

For 1000 classes/signs from MS-ASL Test overall accuracy: 37%

Conclusions:

We saw the results were not bad, but the KEY Frames MLP model was better. And perhaps due to the lack of samples in the dataset this model didn't have as good accuracy as Key Frames MLP model.

CONCLUSIONS

1. The Key frame MLP model, as we named it, has got a very high test overall accuracy of 65% for MS-ASL 1000 (61% for average class). It came from a simple idea and using only MLP's, that on the other hand are good for static images and frame recognition. Some other models have "attention" for giving more importance to some "tokens". Our model does something similar but looking at each frame to find the best key frames.

2. Through experiments with LSTM and SPOTER, we **confirmed the critical role of dataset size and class balance**: the same LSTM model that failed on MSL (30%) reached ~80% accuracy when trained on the Greek Sign Language (GSL) subset with 20 well-populated classes.

3. We **learned deeply about pose-based modeling, Transformers, MLPs, and LSTM architectures**, and how preprocessing (e.g., angle extraction, speed computation, data normalization) can significantly impact model performance in sign language tasks.

4. Finally, we saw the limitations of augmentation techniques like noise or scale shifts and identified that **real data quality and smart preprocessing** are more impactful than synthetic expansion when working with temporal pose sequences.

5. Sometimes, the simplest solution is also the most effective. While complex or cutting-edge approaches may seem more attractive, our results confirm that straightforward methods—when well applied—can outperform more elaborate alternatives. It's a reminder that innovation doesn't always mean complexity, and that clarity and focus often lead to the best outcomes.

NEXT STEPS

- **Dataset Expansion**
  Build a custom dataset with significantly more videos per class to enhance model generalization and performance.

- **Real-Time System Integration**
  Deploy the model in a real-time environment to assess latency, responsiveness, and robustness under practical conditions.

- **User-Centric Evaluation**
  Involve native signers or interpreters to validate system accuracy and gather qualitative feedback on usability and reliability.

- **Use continuous sign language**
  Extend the current system to handle continuous sign language recognition, where signs are performed in natural, unsegmented sequences. This would involve adapting

the model to detect and classify signs within a stream, simulating real-world conversations.

- **Improve our model, e.g. increase the number of MLPs** of the Key frame MLP model to see what improvement can be achieved

How to run the code

Please see the repository:

https://github.com/cvilafer/AIDL-Project

* All figures and images in this article are also available in the repository.

REFERENCES

Boháček, M., & Hrúž, M. (2022). *Sign pose-based transformer for word-level sign language recognition*. In **Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) Workshops** (pp. 182–191). IEEE. https://openaccess.thecvf.com/content/WACV2022W/HADCV/html/Bohacek_Sign_Pose-Based_Transformer_for_Word-Level_Sign_Language_Recognition_WACVW_2022_paper.html

Xin Tran, **Hang Wang**, **Andrew Milz**, **A. A. Hoff**, y **M. S. Ryoo** (2018):
A Closer Look at Spatiotemporal Convolutions for Action Recognition.
Introduce el bloque **R(2+1)D**, donde cada convolución 3D se factorea en una **espacial (2D)** seguida de una **temporal (1D)**, mejorando la capacidad de optimización y el rendimiento en tareas de reconocimiento de acciones

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780.

Vaezi Joze, H. R., & Koller, O. (2018). MS-ASL: A large-scale data set and benchmark for understanding American Sign Language. *arXiv preprint arXiv:1812.01053*.

Adaloglou, N., Chatzis, T., Papastratis, I., Stergioulas, A., Papadopoulos, G. T., Zacharopoulou, V., Xydopoulos, G. J., Atzakas, K., Papazachariou, D., & Daras, P. (2020). A comprehensive study on sign language recognition methods. *arXiv preprint arXiv:2007.12530*.