**Instituto Tecnológico**

**y de Estudios Superiores de Occidente**

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación del 29 de noviembre de 1976.

Departamento de Electrónica, Sistemas e Informática

Maestría en Sistemas Computacionales



**Landmark-driven Transformer Encodings for real-time Mexican Sign Language Recognition.**

Trabajo recepcional que para obtener el grado de

Maestro en Sistemas Computacionales

Presenting: Diego Adrián Santa Cruz Baur

Advisor: Víctor Hugo Martínez Sánchez

Tlaquepaque, Jalisco. Haga clic aquí para escribir una fecha..

**Instituto Tecnológico**

**y de Estudios Superiores de Occidente**

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación del 29 de noviembre de 1976.

Departamento de Electrónica, Sistemas e Informática

Maestría en Sistemas Computacionales



**Codificadores transformer basados en marcadores espaciales para el reconocimiento de la Lengua De Señas Mexicana en tiempo real.**

Trabajo recepcional que para obtener el grado de

Maestro en Sistemas Computacionales

Presenta: Diego Adrián Santa Cruz Baur

Asesor: Víctor Hugo Martínez Sánchez

Tlaquepaque, Jalisco. Haga clic aquí para escribir una fecha..

AGRADECIMIENTOS (INGLÉS)

El autor desea dar las gracias a [Escriba los Agradecimientos].

[Para quienes hayan recibido apoyo CONACYT, se debe incluir aquí el agradecimiento por la Beca recibida, así como el número de beca – no confundir con número de CVU.]

[Se sugiere incluir agradecimiento al ITESO si se les apoyó con descuento o beca. Lo mismo para la empresa donde trabajan. También podrían incluir aquí agradecimiento a sus profesores, asesores o directores del trabajo.]

AGRADECIMIENTOS (ESPAÑOL)

El autor desea dar las gracias a [Escriba los Agradecimientos].

[Para quienes hayan recibido apoyo CONACYT, se debe incluir aquí el agradecimiento por la Beca recibida, así como el número de beca – no confundir con número de CVU.]

[Se sugiere incluir agradecimiento al ITESO si se les apoyó con descuento o beca. Lo mismo para la empresa donde trabajan. También podrían incluir aquí agradecimiento a sus profesores, asesores o directores del trabajo.]

DEDICATORIA (INGLÉS)

El autor dedica esta tesis a [Escriba la Dedicatoria].

DEDICATORIA (ESPAÑOL)

El autor dedica esta tesis a [Escriba la Dedicatoria].

ABSTRACT

[Hablar brevemente sobre las partes de este trabajo, y lo que se presenta en cada uno] Se presenta una breve introducción a [al problema principal que resuelve este trabajo], el cual tiene como objetivo principal [mencionar objetivo principal], resolviendo de manera particular [a los objetivos específicos. [Se presenta el desarrollo del trabajo y sus principales resultados.] Finalmente, [se presentan las conclusiones del trabajo]. [Este resumen cuenta con 250 palabras máx.]

A

RESUMEN

[Hablar brevemente sobre las partes de este trabajo, y lo que se presenta en cada uno] Se presenta una breve introducción a [al problema principal que resuelve este trabajo], el cual tiene como objetivo principal [mencionar objetivo principal], resolviendo de manera particular [a los objetivos específicos. [Se presenta el desarrollo del trabajo y sus principales resultados.] Finalmente, [se presentan las conclusiones del trabajo]. [Este resumen cuenta con 250 palabras máx.]

TABLA DE CONTENIDO

[Dar clic en el menú de Referencias/ Dar clic en Tabla de Contenido/ Seleccionar Tabla de Contenido Personalizada/ Seleccionar Formato Formal, 4 Niveles, Dar Clic en Opciones. En la ventana de Opciones, asignar Nivel de TDC 1 a Título Intro. Dar clic en Aceptar, y de nuevo en Aceptar].

[AGRADECIMIENTOS (INGLÉS) 3](#_Toc201326121)

[AGRADECIMIENTOS (ESPAÑOL) 4](#_Toc201326122)

[DEDICATORIA (INGLÉS) 5](#_Toc201326123)

[DEDICATORIA (ESPAÑOL) 6](#_Toc201326124)

[ABSTRACT 7](#_Toc201326125)

[RESUMEN 8](#_Toc201326126)

[TABLA DE CONTENIDO 9](#_Toc201326127)

[LISTA DE FIGURAS 10](#_Toc201326128)

[LISTA DE TABLAS 11](#_Toc201326129)

[LISTA DE ACRÓNIMOS Y ABREVIATURAS 12](#_Toc201326130)

[1. INTRODUCCIÓN 13](#_Toc201326131)

[2. ESTADO DEL ARTE o de la TÉCNICA 16](#_Toc201326132)

[2.1. Tema relacionado 1 17](#_Toc201326133)

[2.2. Tema relacionado 2 17](#_Toc201326134)

[3. MARCO TEÓRICO/CONCEPTUAL 19](#_Toc201326135)

[3.1. Concepto básico 1 20](#_Toc201326136)

[3.2. Esquema básico 2 20](#_Toc201326137)

[3.2.1. Si se requiere subtema 20](#_Toc201326138)

[3.2.1.1. Subsubtema 20](#_Toc201326139)

[4. DESARROLLO METODOLÓGICO 21](#_Toc201326140)

[5. RESULTADOS Y DISCUSIÓN 41](#_Toc201326141)

[5.1. Resultados 42](#_Toc201326142)

[5.2. Discusión 42](#_Toc201326143)

[6. CONCLUSIONES 43](#_Toc201326144)

[6.1. Conclusiones 44](#_Toc201326145)

[6.2. Trabajo Futuro 44](#_Toc201326146)

[BIBLIOGRAFÍA 45](#_Toc201326147)

LISTA DE FIGURAS

[Figure 1: Methodology overview 22](#_Toc200982460)

[En el menú de Referencias, seleccionar Insertar Tabla de Ilustraciones. Si se desea insertar para Figuras, solo se debe seleccionar el rótulo de Figura.]

LISTA DE TABLAS

[Table 1: Raw dataset samples 22](#_Toc200977930)

[En el menú de Referencias, seleccionar Insertar Tabla de Ilustraciones. Si se desea insertar para Tablas, solo se debe seleccionar el rótulo de Tabla.]

LISTA DE ACRÓNIMOS Y ABREVIATURAS

[Del lado izquierdo va el acrónimo o abreviatura y del lado derecho su significado. La columna de en medio se deja en blanco. Al final, se deben quitar los bordes de la tabla]

|  |  |  |
| --- | --- | --- |
| MSL |  | Mexican Sign Language |
|  |  |  |
|  |  |  |
|  |  |  |

[5 SALTOS DE LÍNEA CON ESTILO NORMAL ANTES DE PONER EL TITULO]

# INTRODUCTION

**Resumen:** En este capítulo se presenta brevemente los antecedentes del objeto de estudio, justificación del objeto de estudio, justificación y la definición del problema. [hipótesis si la investigación lo justifica].

Sign languages are full-fledged natural languages which develop organically within Deaf communities worldwide that use visual signals using primarily the hands to convey meaning rather than relying on spoken sounds [1]. Mexican Sign Language (MSL) is the native sign language of Mexico’s Deaf population [2]. It is used by approximately 300,000 people [2] as a means of communication, which represents around 13 % of the hard-hearing population in Mexico and 0.2 % of the general Mexican population [3], [4]. These numbers reflect the isolation that Deaf people sometimes live in. Enabling Deaf people to communicate with both the speaking and non-speaking Mexican population is necessary for a more inclusive society.

Creating real-time sign-language recognition systems can provide the foundation for simple educational and translation tools, improving access to classrooms, workplaces, and public services, and is an active area of research worldwide [5], [6].

Ultimately, the ambition for these systems is to tackle the full complexities of sign language. However, collecting and annotating the large, richly contextualized datasets required for these capabilities remains a significant challenge [5], [6], [7]. For example, some of the more advanced components of sign languages, such as their grammar, require continous sign language capture—where each sample corresponds to full sentences or beyond. While only a handful of such datasets exist globally [8], MSL has seen one notable effort: González-Rodríguez et al. [9] assembled 1 000 video samples for ten simple sentences using synchronized depth cameras; however, this resource is not publicly available. Other routine aspects of sign language like pointing signs [10]—where the signer will point directly to a specific object, person or place to refer to it—are also challenging to account for and, as far as the author of this work is aware of, have not been addressed by any Sign Language Recognition system.

As a result, many studies focus on more tractable subproblems for which annotated data exists. Particularly relevant to this work is the prior MSL research which has tackled the following challenges:

* Isolated Sign Language Recognition
  + Publicly accesible corpora in MSL concentrates on Isolated Sign Language – where each sample contains a single sign. Therefore most research in the field has concentrated on this subproblem [11], [7], [12], [13]
* Dynamic Sign Language Recognition
  + A division can be made between signs that require movement (called “dynamic”) and those that do not (called “static”). The former adds a complexity to the challenge that many modern machine learning architectures are well equiped to address, and therefore most recent works in the area focus on both dynamic and static signs, whether with separate models [11], or jointly [7], [12].

In this work, we concentrate on isolated sign recognition of both dynamic and static alphanumeric signs with a single model.

Drawing on the distinctive features of MSL outlined in [10], our approach treats each sign at the lexical level as a composition of six minor units:

* Hand Configuration (HC)
* Place of Articulation (PA)
* Hand Motion (HM)
* Direction of Motion (DM)
* Hand Orientation (HO)
* Non-Manual Cues (NMC)

We apply a complex data preparation methodology to extract the features units that concern our videos, and use it to develop what is, to our knowledge, the first highly efficient, real-time recognition unified system for both static and dynamic alphanumeric MSL signs that runs on consumer-grade hardware. Throughout this process, we also conduct experiments to evaluate the hypothesis that preprocessing focused on distilling the six minor units of MSL (HC, PA, HM, DM, HO, FE) enhances model performance.

## Mexican Sign Language background

Mexican Sign Language (MSL) is the predominant language of the Deaf community in Mexico [10]. It originated with the foundation of the Escuela Nacional de Sordomudos in Mexico City in 1867 [10], and is a fully developed natural language with its own grammar, syntax, and lexicon, distinct from spoken Spanish [10]. Mexican Sign Language was officially recognized as a national language by the mexican government in 2005 [14], and it exhibits regional variation across communities—most notably along the rural–urban divide [10].

This research focuses on a specific subset of Mexican Sign Language lexicon: alphanumeric signs. It includes the 26 signs corresponding to the letters in the english alphabet, as well as “LL” and “RR”. Although “Ñ” is generally considered as part of the MSL alphabet and has it’s own sign, it is not present in the dataset we used. Additionally, our dataset also includes the signs for the numbers from 0 to 10, although we will not use the data for 0 and 7 for reasons discussed later.

To understand the impact of choosing these alphanumeric signs for the scope of our project, this section discusses how signs are generally categorized and the portions that the subset we chose covers.

Cruz Aldrete et al. propose comprehensive guidelines to describe signs as combination of the following features in [10]:

* Hand Configuration (HC)
* Place of Articulation (PA)
* Hand Motion (HM)
* Direction of Motion (DM)
* Hand Orientation (HO)
* Non-Manual Cues (NMC)

The first five features can be applied to each of the two hands, and combining this either symetrically or freely adds further complexities [10]. In the case of our project, we are limited to single-hand sign recognition. Although a path extending the system to bimanual sign recognition is easy to envision using the same tools we already use, it would require a complete reconfiguration of our pipeline. This is the main limitation in terms of potential for rapidly extending the use of our project, but it also served to clearly define the boundaries for the development workflow.

For each of the features, Cruz Aldrete et al. [10] provide what they claim is a complete listing of the options in use for Mexican Sign Language. For the Hand Configuration element, Cruz Aldrete et al. identify 30 groupings with each one having between 3 and 26 variants. For Hand Orientation, 9 options are shown which can be seen in Figure [?]. For the Place of Articulation, 39 points are specified, primarily in the face, torso, arms and hands areas. The Direction of Motion element refers to the trajectory the hand follows throughout the articulation of the sign, and 10 options are identified, not accounting for signs in which the hand does not move. Then, 23 different movement patterns relating to the fingers, hands, wrists and arms are described, not accounting for static signs. Finally, 13 groups of Non-Manual Cue actions are grouped, mostly based on the body involved in the action (eyes, nose, eyebrows, …), with each group having from 1 to 12 specific actions.

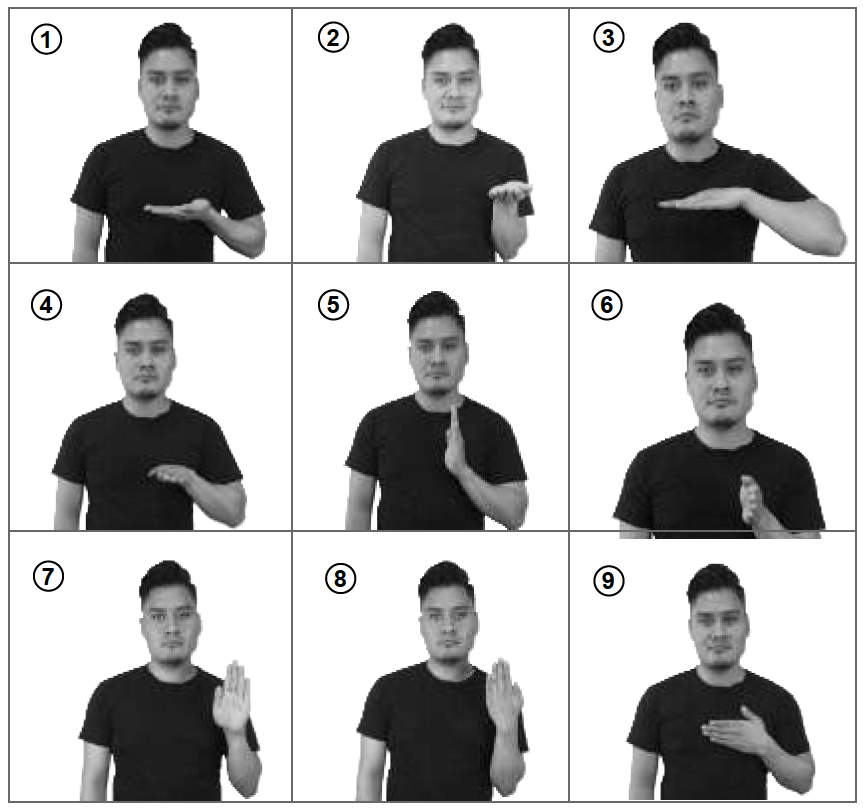
Achieving substantial coverage on all of these features is unrealistic given the current publicly available MSL datasets. This research focuses on hand configuration for four main reasons. First, when developing a complete Sign Language Recognition system under the framework that decomposes signs into the 6 previously mentioned units, it is apparent that hand configuration is essential because, while not every hand includes movement for example, it must have a specific hand shape. Second, the human hand—with its several fingers that move both independently and together—poses a challenging pattern-recognition problem which leads to new understanding of Sign Language as well as Machine Learning concepts. Third, modern real-time hand-landmarking models produce vector outputs that map directly to handshape descriptors, opening up opportunities to investigate this approach.

Amongst the signs in our dataset, we observe variance for 4 of the 6 features: Hand Configuration, Hand Motion, Direction of Motion and Hand Orientation. A case by case for each of the signs can be seen in table [?].

Out of the 30 hand configuration groupings listed in [10], our dataset covers 25 of them, missing only the “I-L”, “Dedo Medio”, “Dedo Anular”, “I-L” and “Pico” configurations. As mentioned, this was the driving factor in the choice of our dataset. Additionnally, although alphanumeric signs are not particularly prevalent in common MSL communication [10], their hand configurations do carry a special importance in the fact that some signs use the hand configuration for to the first letter of the word in Spanish corresponding to the sign in a phenomenon called “initialized signs” [10]. Although by no means conclusive data, in a comparison between American Sign Language (ASL) and MSL, V. Martínez-Sánchez et al. noted that out of 100 words, 12 were initialized in ASL while 37 were initialized for MSL [7].

For hand orientation, 6 out of the 9 options were covered, with a heavy presence of signs where the palm faces the camera, with the fingers pointing up (orientation number 7 in Figure [?]).

|  |
| --- |
| Figure : Hand orientations as shown in [10] |



For hand placement, there was minimal coverage, as for all of the alphanumeric signs selected the area of articulation is not of particular importance and generally defaults to the area in front of the chest. For the two “double letter” signs, there is somewhat of a specificity in terms of area however, in that they both use the “double letter” motion in which the hand moves from one side of the chest to the other.

Similarly, there is very little variance in terms of direction of movement. All of the signs, even those which include movement, happen in a single place, except for the “X” sign, which involves a “pulling” like movement in the depth axis.

We observe somewhat greater variation in hand motion—only 5 of the 24 movement types appear in our dataset, and most signs remain static. Although our focus is on hand configuration, motion recognition is a very interesting challenge: modeling the temporal dynamics and subtle trajectories of signs demands sophisticated techniques, and extending a recognition system to continuous signing adds the extra obstacle identifying the beginning and end of signs, which relies on movement pattern identification for accurate segmentation [?].

Finally, although alphanumeric signs don’t contain any mandatory non-manual cues, the signers in our dataset will often mouth the letter in Spanish corresponding to the alphabetical signs, an action which is not uncommon for this particular set of signs given their heavy connection to Spanish [10].

Table : Description of alphanumeric signs by element

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sign** | **Hand configuration** | **Hand orientation** | **Movement description** | **Movement direction** |
| A | A | 7 | Static | In place |
| B | B | 7 | Static | In place |
| C | C | 5 | Static | In place |
| D | D | 5 | Static | In place |
| E | E | 7 | Static | In place |
| F | F | 7 | Static | In place |
| G | L | 9 | Static | In place |
| H | H | 9 | Static | In place |
| I | I | 7 | Static | In place |
| J | I | 7->6 | Wrist rotation | In place |
| K | P | 5<->6 | Repeated “wrist nodding” | In place |
| L | L | 7 | Static | In place |
| LL | L | 7 | Linear | Side-to-side |
| M | M | 4 | Static | In place |
| N | N | 4 | Static | In place |
| O | O | 5 | Static | In place |
| P | P | 5 | Static | In place |
| Q | Q | 4<->7 | Repeated “wrist nodding” | In place |
| R | R | 7 | Static | In place |
| RR | R | 7 | Linear | Side-to-side |
| S | S | 7 | Static | In place |
| T | T | 7 | Static | In place |
| U | U | 7 | Static | In place |
| V | 2 | 7 | Static | In place |
| W | 3 | 7 | Static | In place |
| X | Q | 6 | Linear | Pull towards the body (depth axis) |
| Y | Y | 9 | Static | In place |
| Z | 1 | 7 | Zig-zag | Zig-zag |
| 0 | O | 5 | Static | In place |
| 1 | 1 | 8 | Static | In place |
| 2 | 2 | 8 | Static | In place |
| 3 | 3 | 8 | Static | In place |
| 4 | 4 | 8 | Static | In place |
| 5 | 5 | 8 | Static | In place |
| 6 | A | 9 | Static | In place |
| 7 | H | 9 | Static | In place |
| 8 | 8 | 9 | Static | In place |
| 9 | B->S | 8 | Retraction of fingers | In place |
| 10 | 5 | 8->4 | Wrist rotation | Down |

# STATE OF THE ART

**Resumen:** En este capítulo se presenta un resumen de los trabajos relacionados con [el objeto de estudio].

## Historical progression in Sign Language Recognition

The evolution of Sign Language Recognition systems has closely followed advancements in computer science and machine learning.

Early research in the field often approached the problem using statistical models. For example, in 1995, [15] achieved real-time recognition of American Sign Language word signs using Hidden Markov Models. This model is intended to be used on sequential data where it can be assumed that the relationship between successive observations contains valuable information [16]. It has been widely used for phenomena for which observations can be made over time, making it specially well suited for isolated dynamic signs [6]. However, Hidden Markov Models have been shown to scale poorly to large amounts of data because it becomes computationally costly to train these models and use them for inference [16]. They also do not perform well at identifying long-range relationships between observations, making it less effective for Continuous Sign Language Recognition [16]. It is also important to note that at to this point, the data on which these models were trained on was not directly the images or videos as is often the case today, but rather handcrafted features laboriously derived from the data with the use of special equipment and requiring domain-specific knowledge [17].

In the early 2000s, researchers began exploring alternative machine learning models such as Support Vector Machines (SVMs) and K-Nearest Neighbors (KNN) [6]. These models offered lower computational cost and a higher potential for scalability as compared to Hidden Markov Models but lost the intrinsic sequential component that helped HMMs to deal with dynamic signs. Therefore, advances were mainly made in static Sign Language Recognition. These models also still relied on the manual extraction of hand-crafted features. In the case of KNN, this is because it’s mode of operation is to compare instances in the data and find the most similar instances based on some distance metric, but it has no understanding of the structures within the data. For SVMs, it tries to find a way to separate classes as much as possible based on the raw data, but, again, it has no understanding of the structures within the data.

The popularization of neural networks in Machine Learning was a breakthrough in Sign Language Recognition because it greatly reduced the need to manually extract hand-crafted features [6]. Neural networks began to use the raw pixel information of the images (or of the frames in videos) as their direct input [18]. As opposed to most traditional Machine Learning models, neural networks are specially well suited to identify the structures and patterns within data. This is because their architecture, particularly in the case of Deep Neural Networks, allows then to identify features of growing complexity layer by layer. This opened the field to a wider section of researchers who were able to shift their focus from the laborious preparation of the data to explore the aspects particular to Sign Language Recognition that could be exploited by more sophisticated neural network architectures.

Convolutional Neural Networks (CNNs) [19] [20] are particularly well suited for Sign Language Recognition because of their ability to exploit spatial structures [21]. The core architectural feature of CNNs is the convolutional layer, where small filters are applied to local regions of the image to detect basic patterns like edges, textures and corners [18]. By moving the area this filter acts on across the image like a sliding window, the whole input is covered while the spatial relationship between nearby pixels is preserved. This is particularly useful in the case of Sign Language Recognition, where local patterns in small regions of the image, notably those where hands appear, are highly meaningful. Another important attribute for CNNs is the fact that the same filters are used on different parts of this image. These filters can be defined by a relatively low number of weights, making the model computationally lighter and usable for real-time inference [18].

|  |
| --- |
| Figure : Hand landmarks extracted by MediaPipe |

Sign Language Recognition projects using Convolutional Neural Networks directly to predict classes have obtained excellent results and have repeatedly advanced accuracy standards in the field [5] [22]. But in fact, CNNs have proven highly effective for object detection and classification in images in general, with Sign Language Recognition being only one of the many applications where their ability to exploit spatial patterns has revolutionized Machine Learning [18]. Because of this, some projects make use of this architecture for intermediate tasks such as image segmentation to identify areas in the image where hands appear, or even to extract features from the images, and then to use those features as the input for sing language specific models. One end-to-end example is [21], where first a CNN is used to extract features which are then fed into an HMM. Some projects also make use of pre-trained models based on CNNs to extract features. For example in [23], the authors use the OpenPose library [24] to extract features and then feeds them into a LSTM model. Other cases [11] like this make use of the Google MediaPipe framework which provides pre-trained CNN models for problems such as hand landmarking, pose detection, and face landmarking [25] [26]. The hand landmarking model extracts three dimensional coordinates for the 21 points which can be seen in Figure [?] from images containing hands. This eliminates the need for the labor-intensive process of manually extracting handcrafted features from raw pixel data.

Recurrent Neural Networks (RNNs) complement Convolutional Neural Networks by exploiting the temporal aspect of video data, making them especially useful for dynamic and continuous Sign Language Recognition. While basic RNNs struggle with capturing long-term dependencies because of vanishing gradients, variants such as Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) were developed to better handle information over longer sequences [18], [19]. Early systems often combined CNNs to extract spatial features with LSTM or GRU layers to capture temporal patterns, showing improved accuracy in various sign language recognition tasks [20], [21]. However, since RNNs process data sequentially, inference can be slower, leading researchers to explore Transformer architectures, which use self-attention to efficiently model these long-term relationships [22], [23].

## Mexican Sign Language Recognition

In this section, we review existing research on Mexican Sign Language Recognition.

Several existing systems rely on specialized hardware for both training and inference. For example, [13] and [12] use a Microsoft Kinect depth camera to record Mexican Sign Language videos. While this method can yield excellent results due to higher data quality, it also reduces accessibility for end users.

González-Rodríguez et al. [9] similarly employed a specialized depth-sensing camera to create a Continuous Mexican Sign Language dataset containing 10 sentences, each with 1000 samples. They used Google MediaPipe’s Holistic framework, which identifies hand, pose, and face landmarks simultaneously, to extract features from these videos. Several architectures—including RNN, LSTM, Bidirectional RNN, Bidirectional LSTM, GRU, and Transformer—were trained and evaluated on this data to develop a Mexican Sign Language–Spanish bidirectional translation system, achieving a top accuracy of 98.8%. They also analyzed the relative importance of different landmarks, concluding that hand landmarks contributed most significantly.

In 2023, another dataset of Mexican Sign Language was developed, comprising 5000 videos of 100 isolated dynamic gestures [7]. A neural network trained on this dataset achieved 99% accuracy. Such a dataset would be ideal for projects focusing on movement recognition in Mexican Sign Language, as its extensive vocabulary covers a wide variety of hand motions described in [10].

However, the present project’s primary goal is to focus on recognition of hand configurations, and the data collected by Rodriguez-Trejo et al. [27] [28] [29], originally created during another research project [11], serves our purpose more naturally, as we outlined in the introduction. Like the current project, that research used MediaPipe to extract hand landmarks, which served as inputs to several neural network models (GBL, SVM, LSTM, GRU). It resulted in two separate models—one for dynamic signs and another for static signs. This prior research is the closest to our current study, as it shares both the original dataset and an initial data-preparation method. The key differences between that project and ours include:

* the neural network architectures explored,
* the use of a unified system recognizing both static and dynamic signs,
* the addition of a secondary system to detect the active hand,
* a series of alternative data-preparation techniques explored in the current work.

# THEORETICAL FRAMEWORK

**Resumen:** En este capítulo se presentan las bases teóricas y conceptuales sobre [el objeto de estudio].

In this section, we introduce the theoretical background that supports the decisions made in this work. We organize the section in five parts: the lexical structure of signs in Mexican Sign Language, feature extraction and representation, feature reduction, neural network architectures used in Sign Language Recognition, and the evaluation metrics used to compare models.

## Lexical structure of Mexican Sign Language

Sign languages are natural human languages that use the body—primarily the hands—to convey meaning instead of sound. Like spoken languages, they have their own grammatical and syntactic rules. In the case of Mexican Sign Language (MSL), Cruz Aldrete et al. [10] propose that each sign can be described using six components:

* Hand Configuration (HC)
* Place of Articulation (PA)
* Hand Motion (HM)
* Direction of Motion (DM)
* Hand Orientation (HO)
* Non-Manual Cues (NMC)

This framework allows for a structured description of signs in a way that supports both linguistic analysis and computational modeling. Each component contributes to meaning, and the combination of features determines the identity of a sign. The components vary in relevance depending on the sign. For example, some signs may include motion or facial expressions, while others may not.

## Feature Extraction and Representation

## Convolution Neural Networks (CNN)

CNNs are a type of neural network designed to efficiently process data with a grid-like structure, such as images. Their key innovation lies in the use of convolutional layers, which apply small, learnable filters across local regions of the input. This approach exploits spatial locality while dramatically reducing the number of parameters through weight sharing. As a result, CNNs can build hierarchical representations of patterns: early layers typically detect edges and simple textures, while deeper layers capture increasingly abstract features like shapes or objects. This structure also confers a degree of translation invariance, allowing the network to recognize patterns regardless of their exact position in the input. Pooling layers, which reduce the spatial dimensions of intermediate representations, enhance robustness to small translations or distortions. For a more detailed theoretical explanation of CNNs and their design principles, see [20] and [27].

## Landmark Extraction with MediaPipe

Instead of working directly with image pixels, this project uses landmark coordinates as input. MediaPipe is a tool developed by Google that uses CNNs to detect landmarks of the hand and body in real-time from regular video input [25], [26]. It returns 21 three-dimensional points per hand and a small set of body points. This type of input is lighter than raw images, less sensitive to background noise, and already structured in a way that makes it easier to extract features like hand configuration or motion.

## Feature Extraction and Representation

High-dimensional inputs such as body or hand landmarks often include noise or irrelevant variation, which can negatively impact model performance and increase computational costs. Dimensionality reduction techniques aim to retain the most informative aspects of the data while discarding redundant or less useful components.

A widely used linear approach is Principal Component Analysis (PCA). PCA computes the directions, or principal components, along which the data varies the most, and projects the input onto a lower-dimensional subspace spanned by the leading components. This reduces dimensionality while preserving variance and orthogonality, making PCA both efficient and interpretable. For more details on the theoretical foundations, see [28].

For more complex data structures where linear assumptions fall short, Uniform Manifold Approximation and Projection (UMAP) [29] offers a powerful alternative. UMAP constructs a weighted graph of local relationships and optimizes a low-dimensional embedding that reflects both local topology and global structure. Unlike PCA, UMAP can capture curved or folded manifolds and is particularly well-suited for visualizing high-dimensional clusters or patterns. While slower to compute, its representations often yield better separability for downstream tasks.

## Neural Network Architectures for Sequential Modeling

Recurrent Neural Networks (RNNs) are designed to handle sequential data, where the order of inputs matters [30]. Unlike regular neural networks, RNNs maintain a hidden state that is updated over time as new inputs are processed. This allows the network to remember previous information, which is particularly useful for tasks like language modeling and Sign Language Recognition (SLR). In the context of SLR, RNNs are useful for recognizing dynamic signs, where hand positions evolve over time across a sequence of frames.

However, regular RNNs face two main problems when learning from long sequences: the vanishing gradient problem and the exploding gradient problem [31]. In the vanishing gradient problem, gradients become extremely small during backpropagation, which makes it hard for the model to learn long-term dependencies. On the other hand, the exploding gradient problem occurs when gradients become too large, leading to unstable model weights and causing the model to diverge. These issues limit the effectiveness of basic RNNs to learn long-range dependencies [31].

Long Short-Term Memory (LSTM) [31] and Gated Recurrent Units (GRU) [32] were introduced to overcome these issues in standard RNNs. Both LSTM and GRU use gating mechanisms to control how much information is passed along and how much is discarded, allowing them to remember important details over longer sequences. LSTMs are more complex, using three gates: the input gate, the forget gate, and the output gate. This structure helps LSTMs overcome the vanishing and exploding gradient problems by controlling the flow of information more effectively. GRUs are simpler and combine the forget and input gates into a single update gate, but are still highly effective. Both models are widely used for SLR because they can capture temporal dependencies in dynamic signs, which is crucial for accurate recognition [33], [9].

Transformers [34] provide an alternative to RNNs for sequential modeling. Rather than processing one frame at a time, Transformers use self-attention, which allows the model to look at the entire sequence at once. Self-attention computes the relationships between all parts of the sequence and assigns different weights to each part based on its relevance, enabling the model to focus on the most important information. To compensate for the lack of inherent order in parallel processing, Transformers include positional encodings, which inject information about the position of each element into the input. This design allows Transformers to model long-range dependencies more effectively than RNNs while also enabling faster training through parallelism.

BERT (Bidirectional Encoder Representations from Transformers) is a specific implementation of the Transformer model. It is bidirectional, meaning it looks at both the preceding and succeeding context in a sequence, rather than just one direction. This bidirectionality improves the model's understanding of context. BERT uses multiple layers of Transformer encoders, each layer applying self-attention and feedforward networks. In this project, we use BERT-based models to process sequences of landmark coordinates extracted from video frames and classify them into signs. BERT's ability to model complex patterns in sequential data makes it well-suited for Sign Language Recognition tasks [35].

## Evaluation Metrics for Classification Models

Evaluating classification models requires metrics that measure how well predictions match true labels. Below are the main metrics used, defined in the multiclass setting [36].

* **Accuracy**:
  + The proportion of correct predictions over the total number of predictions:
* **Precision (per class)**:
  + For a given class :
    - where is the number of true positives for class , and is the number of times class was predicted incorrectly.
* **Recall (per class)**:
  + For a given class :
    - where is the number of times class was the correct label but not predicted
* **Macro-Averaged Precision and Recall**:
  + where is the number of classes.
* **F1 Score (per class)**:
  + The harmonic mean of precision and recall for class :
* **Macro F1 Score**:
  + The average F1 score across all classes:
* **Top-2 Accuracy**:
  + The proportion of times the true class is among the top 2 predicted classes:
* **Z-score of Accuracy**:
  + Compares observed accuracy to what would be expected under random guessing.
  + Let be the number of classes, the number of samples, and the observed accuracy:
    - Expected accuracy:
    - Variance of accuracy:
    - Z-score:

These metrics provide different insights into how well a classification model performs, with the z-score being particularly useful to compare models across different numbers of classes.

# METHODOLOGY

**Resumen:** [En este capítulo se presenta en detalle el desarrollo metodológico que incluye [pasos o proceso a seguir] un resumen de los trabajos relacionados con [el objeto de estudio].]

##### 4.1 Methodology Overview

Inspired by the description of the distinctive features of MSL in [?], we base our methodology on the idea that, at the word level, MSL signs can be divided into 6 minor units:

* Hand Configuration (HC)
* Place of Articulation (PA)
* Hand Motion (HM)
* Direction of the Motion (DM)
* Hand Orientation (HO)
* Facial Expression (FE)

We attempt to materialize these units as vectorial information by extracting per-frame coordinates for a group of body landmarks, primarily in the hands, from our alphanumeric sign videos. From this core landmark dataset, we create multiple parallel versions:

* Raw landmarks
* Geometrically transformed (to distill the minor units of the sign)
* Dimensionality reduced (using a wide variety of reduction techniques)
* Transformed + reduced

We then perform a series of screening evaluations to select the most promising dataset variants, for which we train and benchmark a suite of BERT-based transformer architectures to identify the optimal dataset–model pairing. Once the best combination is determined, we fine-tune the selected model and integrate it into our live-inference pipeline for real-time MSL recognition.

We present an end-to-end pipeline for live-inference Mexican Sign Language (MSL) recognition, from raw video acquisition through final model deployment. Our ultimate goal is a working system that balances recognition accuracy with real-time performance on standard hardware. We first describe the raw dataset and compare it to available alternatives. Next, we detail our multi-phase data preparation process, including landmark extraction, optional geometric transformations, and optional dimensionality reduction, after which we obtain a wide range of datasets to train models. We then outline a two-stage pruning strategy to select the most promising dataset and model architecture before presenting our best model training and tuning pairing.

|  |
| --- |
| Figure : Methodology overview |

##### 4.2 Raw Dataset

This is for two reasons:

* There is a publicly available dataset with excellent organization and annotations with this data
* Although finger-spelling is relatively rare in common usage of Mexican Sign Language, alphanumerical signs still hold an important role in the language
  + Initialization is a phenomenom in which signs take the hand configuration for the first letter of the translation of a sign in a spoken language
  + V. Martínez-Sánchez mentions that when comparing MSL to ASL, it was observed that 37 out of 100 words in MSL were “initialized”, as opposed to 12 for ASL.

The primary objective of this work is to develop a robust Mexican Sign Language recognition system capable of running in real time on commonly owned consumer devices.

A secondary objective is to evaluate the validity of the hypothesis that Mexican Sign Language is, at a sign level,

|  |  |
| --- | --- |
|  |  |
|  |  |

We obtained a publicly shared MSL dataset of 11 signers, each performing the 28 letters (those in the english alphabet as well as “LL” and “RR”), and 11 numerals (0–10) approximately ten times—five repetitions per hand—for a total of 4,156 short video clips [9]. The recordings isolate single static and dynamic signs under controlled conditions, providing balanced coverage across signers, classes, and handedness. Table [?] shows 4 frames from video samples of our dataset for the letters A, B, C and D by four different people, two with each hand.

Table 2: Raw dataset samples

The dataset is exceptionally well labeled and organized, following a clear directory and filename convention. At the top level, each signer’s data sits in a folder named pXX, where XX is a number identifying the signer. Within that, subfolders use the pattern Ciclo\_Y\_5\_Z, where Y denotes the cycle number (which goes from 1 to 5) and Z is “Derecha”  or “Izquierda” for left- or right-hand, indicating the hand which is actively performing the sign. Inside each subfolder, video files are named Ciclo\_Y\_5\_Z\_S.mp4, Y and Z keeping the same meaning, and S indicating the specific letter or numeral. This hierarchy encodes signer ID, cycle, handedness, and class at a glance, simplifying downstream filtering and preparation, analysis and training.

It is also important to note that although the sign label is the “natural” or intended tag, the other annotations can in practice be used as labels. For example, we leverage the handedness annotation to solve a secondary classification problem — active hand detection — since all signs in this dataset are single-handed. While continuous MSL often involves two-handed expressions, this controlled handedness information proves invaluable for our project’s preprocessing (see Section 4.3.3).

We load each clip into a pandas DataFrame, annotating signer ID, sign class, hand used, and frame index to establish a structured foundation for all preprocessing phases.

Amongst the availabel Mexican Sign Language datasets [?, ?], we selected this one

###### 4.2.1 Data Organization and Filtering

|  |
| --- |
| Figure 4: Data Organization and Filtering |

We leverage the dataset’s existing directory hierarchy parse signer ID, cycle number, handedness, and sign class into DataFrame columns. We then apply three filters to ensure data quality:

* We drop clips with fewer than twelve frames to guarantee sufficient temporal context.
* We enforce class balance by requiring at least 100 videos per class; any under-represented classes are excluded.
* We remove the “7” class entirely because its handshape is visually indistinguishable from “G” in this dialect, which would otherwise confuse isolated-sign models.

These steps yield a clean, balanced set of isolated-sign videos.

##### 4.3 Data Preparation

###### 4.3.1 Preparation Overview

Data preparation proceeds in three steps. First, we extract frame-level landmarks to capture hand shape, pose, and detection context (we call this phase 1, or PH1). Next, we optionally transform these landmarks to isolate the principal components of signs—hand configuration, hand orientation, hand position relative to the body, and hand movement (we call this phase 2, or PH2). Finally, we optionally apply dimensionality reduction (PCA or UMAP with varying component counts) to condense high-dimensional features (phase 3, PH3). Each step generates dataset forks that feed into our evaluation and pruning pipeline.

|  |
| --- |
| Figure : Data Preparation |

###### 4.3.2 (PH1) Video to Landmarks

**4.3.2.1 Purpose of Using Landmarks Instead of Images**  
Signs in MSL decompose into four principal components [1]:

* hand configuration
* hand orientation
* hand movement
* hand position relative to the body

These are sometimes supplemented by facial or torso cues for semantic meaning [1].

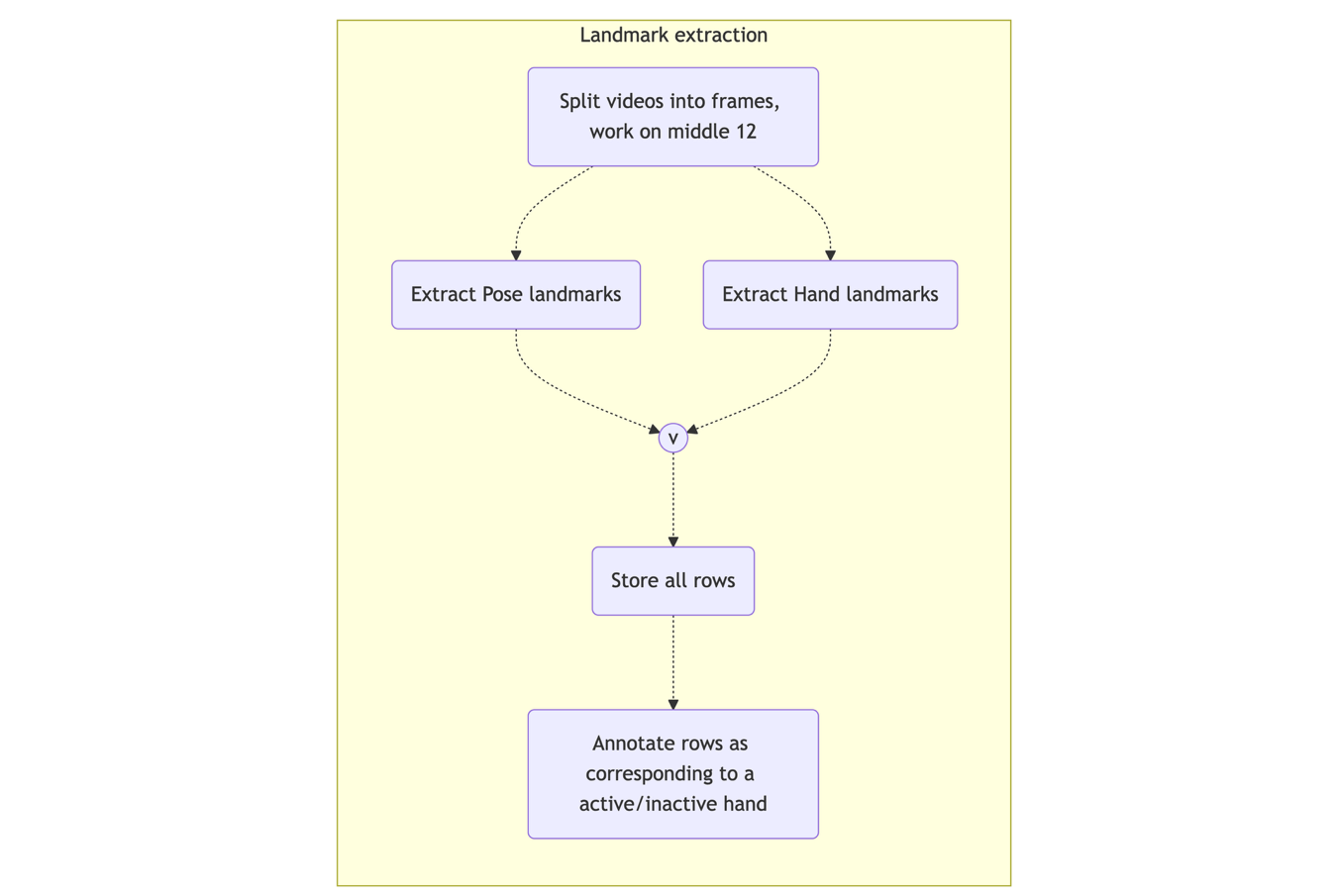
Extracting landmark coordinates offers a direct, low-dimensional representation of these components. Landmarks reduce data size by orders of magnitude compared to raw pixels, eliminate background clutter, and allow us to inherit robustness from MediaPipe’s training on diverse synthetic and real-world hand datasets.

**4.3.2.2 Basic Use of MediaPipe**  
MediaPipe Hands and Pose provide real-time hand and body landmark detection from a single RGB frame without the need for specialized hardware [5,6]. MediaPipe Hands returns 21 three-dimensional hand landmarks, which can be seen in Figure [?], as well as handedness labels, and confidence scores; MediaPipe Pose returns a small set of pose landmarks, which can be seen in Figure [?].

Integrating MediaPipe at the beginning of our system's pipeline allows us to inherit from it's robust capabilities. MediaPipe Hands was trained on a wide variety of data, with varying lighting, quality of image, camera angles, etc. Additionally, the model was partially trained on computer generated 3D images of hands, which allow it to infer depth coordinates even when using non-specialized cameras like MicroSoft Kinect. Lastly, the lightweight, optimized pipeline ensures frame-level feature extraction runs in under 10 ms per frame on CPU—critical for live inference.

|  |  |
| --- | --- |
| Figure : MediaPipe Hand Landmarks | Figure : MediaPipe Pose Landmarks |

**4.3.2.3 Use of MediaPipe for This Project**



|  |
| --- |
| Figure : Phase 1: Landmark Extraction |

A diagram summarizing this section can be seen in Figure [?].

We read each video using python's OpenCV package, and sample the 12 central frames. This strikes a balance between dataset size -- allowing us to obtain over 1000 frames per class -- and temporal coverage of the sign, particularly necessary in the case of dynamic signs. Additionally, it helps address a leakage issue with some of the videos in the dataset where the start (or end, respectively) of the video shows part of the previous (or following, respectively) sign which the signer performed. Through random exploration of the raw dataset, we found that 12 frames per video generally covered the totality of the sign's action. It is worth mentioning that the author of the paper which presented the dataset [9] used for this project, M. E. Rodriguez, used a strategy in a similar project where 30 frames per sign were obtained, and in cases where the video was too short, the last frame of the video was duplicated until the 30 mark was attained.

For each frame, MediaPipe Hands first detects potential hands. Then for each of these hands, it extracts 21 landmarks, the detected handedness, and a confidence score, as well as other information we don’t use here.

There is often more than one hand per frame. As mentioned previously, the signs in our dataset are all single-handed. Therefore, of the detected hands, one is often "inactive" and provides no information on the sign being performed, and we decided it would be better to train a second simpler model to for "active hand detection" which we use prior to the sign recognition model in our live-inference system. To train this model, we need a label for each frame that tells us whether the had is active or not. The frames or videos are not tagged in this manner directly, but the handedness tags on the video combined with the detected handedness tag and detected handedness confidence score provided by MediaPipe offer a proxy which we will describe shortly.

Similarly to the Hands framework, MediaPipe Pose extracts body landmarks, of which we use three (indices 0, 11, 12 in Figure [?]). Because there is only one person per video in our dataset, we never obtain more than one detection per frame.

We omit facial landmarks in this phase because expressions do not influence number and letter signs, and facial expression recognition is a challeging problem in it’s own right that would extend the scope of this project too much. While this decision simplifies our current pipeline, it limits the potential for extending this project to recognition of signs which incorporate facial expressions. It would be desireable in such a case to adapt the framework to integrate facial landmarks.

In total, for each frame we obtain a varying number of rows[[1]](#footnote-1) in a pandas dataframe: one for each hand detected. For each of these rows we apply the following logic:

* if it is the only row corresponding to it's frame in it's video, then we label that row as corresponding to the active hand
* if it is one row of multiple (there are never more than 2) corresponding to it's frame in it's video, but it is the only one where the detected handedness matches the tagged handedness for the video, then we label that row as corresponding to the active hand; we tag the other rows corresponding to that frame of that video as inactive
* if it is one row of multiple corresponding to it's frame in it's video, and it is not the only one for which the detected handedness corresponds to the tagged handedness of the video, then we sort the rows in that group by handedness confidence score, and label the one with the highest score as active and the rest as inactive

In this way we obtain a secondary label for each row indicating the activity of the detected hand.

It is important to note that because the data for inactive hands is not relevant for sign recognition, we will drop the rows labeled as inactive right before training sign recognition models. Also, at this point the unit of data in our dataset is a frame. We intend on taking advantage of the sequential aspect of videos for sign recognition, which should be especially useful for dynamic signs. For that, we will also reformat our data right before training so that the rows corresponding to the 12 frames of a single video are concatenated into a single row per video. We do these two dataset transformations right before transformations right before training instead of earlier to avoid repeated storage of data and application of data preparation processes.

**4.3.2.4 PH1 Output Description**  
The result of this first data preparation process is a pandas dataframe with 69,572 rows and 80 columns, namely:

* 4 video tag columns
  + fileid
  + person\_id
  + cycle\_num
  + handedness
* 3 label columns
  + class\_name and class\_numeric, which indicate the sign
  + active\_hand
* One column to later concatenate rows coming from a single video in the correct order if needed
  + current\_frame
* 21 hand landmark triplet columns
  + h{i}x, h{i}y, h{i}z for i in [0, ..., 20]
* 3 pose landmark triplet columns
  + p{i}x, p{i}y, p{i}z for i in [0, 11, 12]

###### 4.3.3 (PH2) Landmark Transformations

|  |
| --- |
| Figure : Phase 2: Landmark Transformations |

###### As mentioned previously, MSL signs consist of four core components—hand configuration, hand orientation, hand position relative to the body, and hand movement [1]—all of which we already encode via raw landmark coordinates. However, these components intermix in the original camera frame. In PH2, we isolate seek to isolate each component. The diagram in Figure [?] summarizes the following descriptions.

###### To obtain the hand orientation, we approximate the palm of the hand to a plane, obtain a pair of vectors which describe it, compute a vector normal to them which points the way the palm faces and keep the coordinates for these three vectors.

###### To obtain the hand configuration, we use these three vectors and the coordinates of the wrist to define a new frame of reference in which we obtain the coordinates for each of the 21 hand landmarks.

###### To obtain the hand's position relative to the body, we apply a process similar to the previous two, but using the 3 pose landmarks we extracted earlier to define a plane, obtain a vector normal to that plane, using this one and two which define a plane as a base for a new frame of reference with the nose as the origin. Then we compute the mean of the coordinates of the hand in the original frame of reference, and move this to our new body-centered frame of reference.

###### Finally, Time-based movement emerges naturally when we later aggregate per video.

**4.3.3.1 PH2 Output Description**

As mentioned earlier, this phase of data preparation is "optional", meaning that we will keep both this version of the data and that without it to train models. The data that does go through this process exits as a pandas dataframe of 69572 rows and 83 columns, namely:

* 4 video tag columns
  + fileid
  + person\_id
  + cycle\_num
  + handedness
* 3 label columns
  + class\_name and class\_numeric, which indicate the sign
  + active\_hand
* One column to later concatenate rows coming from a single video in the correct order if needed
  + current\_frame
* triplet columns for each of the three vectors of the base of the "wrist's frame of reference"

###### h\_v{i}x, h\_v{i}y, h\_v{i}z, for i in [1, 2, 3]

###### 21 hand landmark triplet columns in the wrist's frame of reference

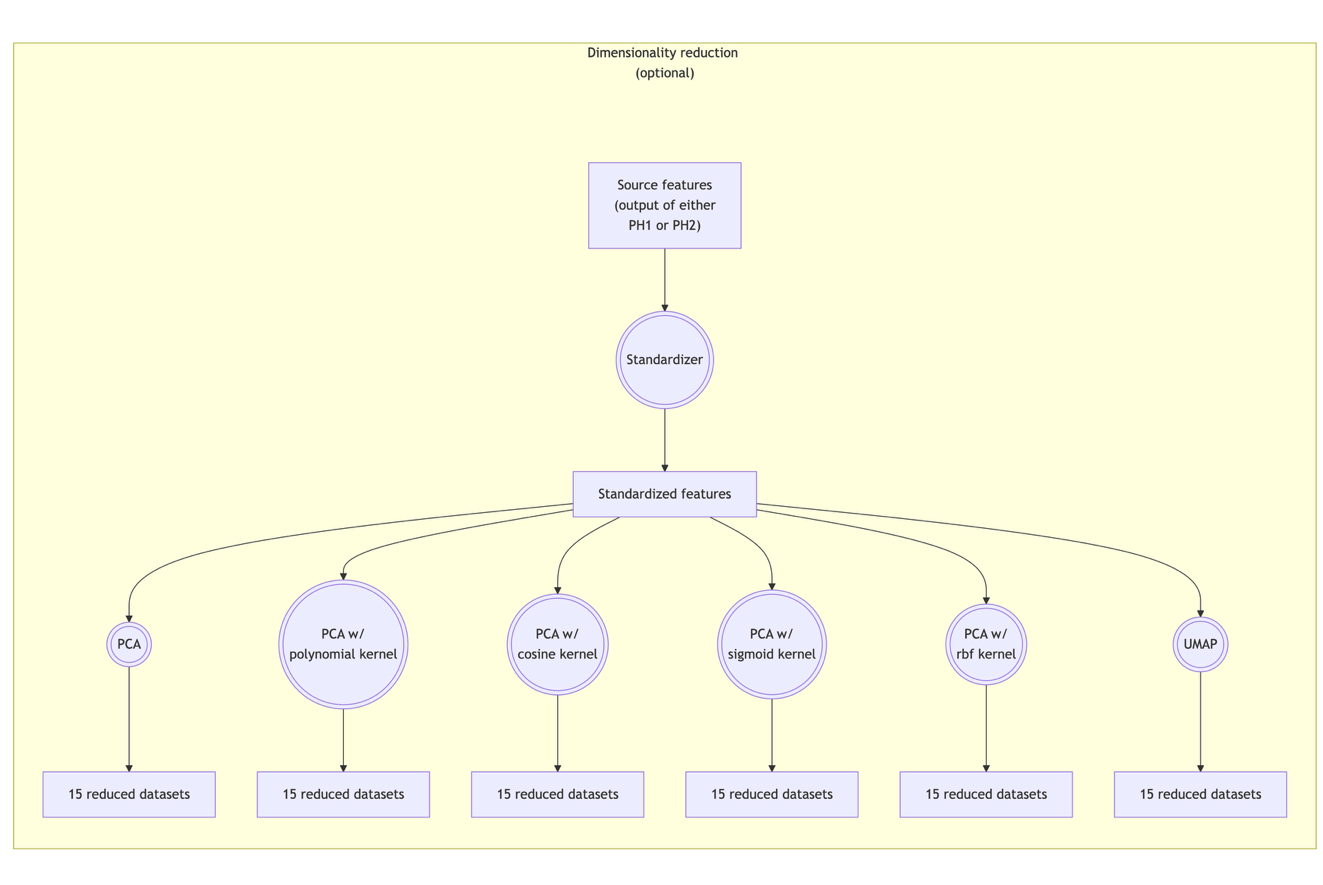
###### wh{i}x, wh{i}y, wh{i}z for i in [0, ..., 20]

###### 1 triplet of colums for the mean hand's coordinate in the "body's" frame of reference

###### p{i}x, p{i}y, p{i}z for i in [0, 11, 12]

###### Note that we no longer keep any pose landmark data.

###### 4.3.4 (PH3) Dimensionality Reduction



|  |
| --- |
| Figure : Phase 3: Dimensionality Reduction |

##### This phase of data preparation explores reducing feature dimensions via PCA and UMAP. We initially looked into this option early in this projects development when we still weren't using BERT based models, but rather K Nearest Neighbors models, which struggle to balance high-dimensional data. However, it proved to be a useful process in our final pipeline, and also provided us a way to visually analyze the quality of our data through graphics using 3 component reducers.

##### Figure [8] shows an overview of the actions performed in this phase. It begins with the output of either PH1 or PH2, which is optional. We first standardize all feature columns—excluding video tags, labels, and frame indices—to zero mean and unit variance. This normalization ensures that each dimension contributes equally to downstream calculations and mitigates scale disparities.

##### Next, we systematically apply dimensionality reduction across multiple reducer–kernel combinations and component counts. We test PCA and UMAP, as well as PCA with polynomial, RBF, sigmoid, and cosine kernels, for reduction component counts going from 1 to 15. We selected this range as roughly a quarter of our ∼ ⁣70 landmark features, balancing information retention with computational cost. Our choice of reduction techniques is due to their mature, easy-to-use implementations in scikit-learn and umap-learn, and their potential for on-device real-time use.

##### Using kernels prior to the PCA reducer proved to slow the process down significantly. Computing the full kernel matrix requires evaluating pairwise similarities for every sample in the dataset—an operation whose time and memory requirements scale quadratically with the number of points. To overcome this bottleneck, we turn to the scikit-learn’s implementation of Nyström approximation: rather than forming the complete n×n kernel matrix, a randomly sampled subset of m≪n “landmark” points is selected, with which only the n×m and m×m kernel blocks are computed, and then use these to reconstruct an approximation of the full matrix. This approximation reduces both computation and storage from O(n2) to O(nm). We used 500 as our value number of landmarks for the approximation, as it greatly reduced the time while mantaining visually similar results in the 3 dimensional case.

##### Using kernels prior to the PCA reducer proved to slow the process down significantly. Computing the full kernel matrix requires evaluating pairwise similarities for every sample in the dataset—an operation whose time and memory requirements scale quadratically with the number of points. To overcome this bottleneck, we use scikit‐learn’s implementation of the Nyström approximation: rather than forming the complete kernel matrix, a randomly sampled subset of “landmark” points is selected, only the and kernel blocks are computed, and then these are used to reconstruct an approximation of the full matrix. This approximation reduces both computation and storage from to We used 500 as our number of landmarks for the approximation, as it greatly reduced the time while maintaining visually similar results in the three‐dimensional case.



##### Each (reducer, kernel, n) combination yields a new dataset fork of the same 69572 rows and columns, where 8 corresponds to the same video tagging, labeling and frame indexing columns as before, and is the number of reduction components.



**4.3.4.1 3-Component Reduction Visualization Insights**

**[?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?]**

##### 4.4 Dataset and Model Selection

###### 4.4.1 Goal

###### We aim to deploy transformer-encoder classifiers for live MSL inference under strict latency and compute constraints. Our objective is to decode continuous sign input with minimal delay, maintain robust accuracy across signer variations, and support on-device inference on standard hardware. To this end, we select five lightweight encoders—bert-tiny, bert-mini, bert-small, bert-medium, and DistilBERT— whose respective sizes are compared in Table [?] and use a structured pruning pipeline to identify the best dataset-model combination without exhaustively training every pair.

|  |  |  |  |
| --- | --- | --- | --- |
| **Encoder** | **Number of Layers** | **Hidden Dimension size** | **Number of parameters** |
| **bert-tiny [?]** | 2 | 128 | 4.43 million |
| **bert-mini [?]** | 4 | 256 | 11.3 million |
| **bert-small [?]** | 4 | 512 | 29.1 million |
| **bert-medium [?]** | 8 | 512 | 41.7 million |
| **DistilBERT [?]** | 6 | 768 | 66 million |

Table 3: Size comparison of different BERT inspired architectures as seen in [?]

###### As our dataset preparation pipeline can generate hundreds of variants—each defined by different combinations of preprocessing steps, dimensionality reducers, and data‐unit formats—pairing every one of these with each of our five candidate encoder architectures would quickly become both too computationally and time consuming for this project.

###### To address this, we implement a two-stage pruning strategy that filters both datasets and models before committing to full training. First, we employ lightweight proxy evaluations—such as K-Nearest Neighbors on reduced data—to identify the most promising preprocessing and reduction settings without training the encoders. Once the top dataset variants are determined, we then perform a second screening across our encoder architectures using a reduced number of epochs. With this two-staged approach we are able to focus our resources on the dataset-model configurations that appear to have the most potential. Figure [?] summarizes this process.

|  |
| --- |
| Figure 11: Dataset and Model Selection |

###### 4.4.2 Dataset Selection

As mentioned in the previous section, the challenge at hand is reducing the number of candidate dataset-model combinations. We began with 546 dataset variants generated by combining:

* PH2 (2 options)
  + Applied
  + Not applied
* PH3 options (91 total):
  + PCA for n components, with n going from 1 to 15, making for 15 options
  + PCA and one of four kernels (Polynomial, Cosine, Sigmoid or RBF) for n components, with n going from 1 to 15, making for options



* + UMAP for n components, with n going from 1 to 15, making for 15 options
  + PH3 not applied (1 option)
* Data-unit formats (3):
  + all rows
  + active-hand rows per frame
  + active-hand rows aggregated per video

As we can see, the largest factor here is the number of PH3 options. Therefore, we first focused on finding what the best combinations in that area of choices would be, and discarding the rest. Figure [?] shows the strategy used to do so.

|  |
| --- |
| Figure : Dataset Selection |

**4.4.2.1 Selection of subsets using K-Means**

To have a more robust interpretation of the value of the different data preparation processes, we wanted to evaluate their impact as the number of classes grew. To do so, we decided to train KNN models on each of the datasets generated throughout the data preparation phases, but to do so on filtered versions of those datasets where only rows corresponding to certain subsets of classes were kept.

To select the subsets of classes we would filter our datasets with, we decided to use the data generated directly after landmark extraction with MediaPipe, and use a K-Means model to obtain the centroid for each class. Then, for each possible subset of classes, we computed a difficulty score based on the cosine similarity between the centroid corresponding to the classes in that subset. The closer the centroids were, the more difficult they should be to differentiate in theory. Finally, for each number of classes going from 2 to the total number of classes (37), we sorted all of the subsets for that number of classes by their difficulty score, and chose 3 for number of classes: one “easy”, which had the difficulty score closest to the the 10th percentile among candidate subsets; one “average”, which had the difficulty score closest to the the 50th percentile among candidate subsets; and one “hard”, which had the difficulty score closest to the the 90th percentile among candidate subsets.

The description above was a simplification of the process. We actually did not compute difficulty scores for every possible subset of classes due to computational limitations. Instead, for each number of classes from 2 to 37, we randomly sampled across all the candidate subsets at most 1000 of them, and computed the difficulty scores for those.

**4.4.2.2 Fit K Nearest Neighbors models for each subset, for each dataset**

Originally, our intention was that now that we had the three selected subsets for each number of classes, we would take each of the 546 datasets, and filtered them for each of the subsets, and fit several K-Nearest Neighbors models to the data. However, once again to be computationally resource-conscious, we only used the subsets corresponding to a reduced number of classes, namely: 2, 9, 16, 23, 30 and 37.

For each the datasets, we fit K-Nearest Neighbors models with the number of neighbors being amogst 1,3,5 and 7.

Then we computed the accuracy of each model. Because we were trying to compare models quality across different numbers of classes, comparing models’ metrics directly gave an advantage to models trained on lower numbers of classes.

To compensate for this, we decided to compute a significance score for the observed accuracy for each of the models, and compare them that way. We did this by considering that a score was more significant the better it was when comparing it to what random guessing would result in for the number of classes in question.

*4.4.2.2.1 Computing significance scores based on observed accuracy*

Let’s first recall the definition of multiclass accuracy. Given an n-class problem with N total examples, let be the number of examples whose true label is i and whose predicted label is j. Then



The accuracy is the proportion of correct predictions:



We assume two things under our random‐guessing model:

1. **Uniformly random true labels.** The dataset has basically no class imbalance, so each class i is equally likely as the true label:



1. **Uniformly random, independent guesses.** The classifier has no information and picks each class jj with equal probability, independently of the true label:



Because we treat “true label” and “guess” as independent under this model, the joint probability of any specific ordered pair (true=i,  guess=j)(true=i,guess=j) is the product of the two marginals:



Now, assuming uniform random guessing among n classes:

1. Each ordered pair (true=i, guess=j) has probability



1. Correct predictions occur when i=j. There are n such diagonal events, each of probability 1/n², so  
   .



Hence the expected (mean) accuracy under random guessing is



Under random guessing, each of the N examples has probability 1∕n of being correctly classified. We model the total number of correct predictions, C, as a Binomial(N, 1∕n) random variable.

1. Expected value of C



1. Variance of C



1. And we have that accuracy is C / N, therefore the variance of accuracy is:



1. And



1. For an observed accuracy Â, the Z-score is



*4.4.2.2.2 Comparing models across number of classes*

Now that we had significance scores for each of the models across number of classes we could compare them more fairly.

Figures [?] and [?] show that PCA using no kernel was generally the best reduction technique, and Figure [?] shows that it was best used with 15 components.

Additionally, we observe that using per frame data generally resulted in better models than per video data for KNN models, as can be seen in Figure [?]. This can intuitively be explained with the fact that KNN models have no concept of sequential data and therefore can not take advantage of the time-series structure of videos. However, we should also note that, as can be seen in Figure [?] all our models performed better for higher numbers of classes, which is proportionally related to the number of samples to train on. Because of this the fact that per frame datasets have both 12 times as many samples and 12 times as many feature columns, we believe that this could also explain why per frame models performed so much better than per video models for KNN.

Another insight that this exercise provided was that generally speaking PH2 landmark transformations negatively impacted our models, as can be seen in Figure [?]. In spite of this, we continue to consider these datasets, even thoug they generally double our traning dataset candidates, as an effort to validate the description of Mexican Sign Language as divided into 4 separate components which is one of the principal hypothesis the present project tries to prove.

In any case, moving forward we greatly reduced the number of candidate datasets, as we fixed PH3 = PCA(15), collapsing 91 options down to two (PCA(15) or none) and reducing our grid to 2 × 2 × 3 = 12 variants.

*4.4.2.2.3 Active Hand model*

In addition to this dataset reduction, we also obtained an excellent active-hand prediction model during exploration using the following configurations:  
[INSERT CONFIGURATION DETAILS HERE]

Therefore, we will no longer be exploring the datasets containing rows for inactive hands, and can further reduce our candidate datasets to 2 × 2 × 2 = 8 final datasets to explore with our 5 candidate model architectures.

###### 4.4.3 Model Selection Methods

We fine-tune each of the 8 datasets with all 5 encoders for a moderate epoch budget approximately proportional to the number of parameters per model seen in Table [?].

For each combination, we log accuracy, top 2 accuracy (meaning the rate at which the true label appears in the top 2 likeliest classes according to the model), macro precision, macro recall and macro F1 score.

We then compare the top 3 scores for different metrics for each model, and based on Figures [?], [?] and [?], where we see that …, we conclude that bert-mini is the best architecture for our usecase.

###### 4.4.4 Final Selection

BERT-mini is trained for ~5× more epochs on each of the 8 datasets than in section 4.4.3. We compare final metrics—prioritizing top-1 accuracy with top-2 and F1 score as tiebreakers—and select the sign-only, PH2-off, PH3-off dataset for deployment.

##### 4.5 Best Model Finetuning

The chosen dataset–BERT-mini pair undergoes extended training for 2× the epochs of Section 4.4.4. The final configurations used are:

# RESULTS AND DISCUSSION

**Resumen:** [En este capítulo se presentan los resultados obtenidos del desarrollo de este trabajo y una discusión sobre [el objeto de estudio]].

## Results

[Teclee los resultados en pasado. Ponga título a sus tablas y gráficos. Hacer referencia explícita utilizando la numeración. Ejemplo: …, como se muestra en la Figura 10. NO referenciar mencionado: como en la siguiente figura, o similar.]

## Discussion

[resultados más relevantes de este trabajo, los más relevantes de otros trabajos, comparar, referir a nuevos trabajos que puedan surgir de aquí, o problemas.

# CONCLUSIONS AND FUTURE WORK

**Resumen:** [En este capítulo se presentan las conclusiones y trabajo futuro en relación a [el objeto de estudio]].

## Conclusions

[Las conclusiones deben responde a los objetivos establecidos]

## Future work

[Se refiere a recomendaciones o descripciones sobre líneas de investigación que abre este trabajo, aplicaciones inmediatas que se derivan, o desarrollo de componentes o extensiones del desarrollo.]

# Bibliography

|  |  |
| --- | --- |
| [1] | C. Valli, C. Lucas, K. Mulrooney and M. Villanueva, Linguistics of American Sign Language: An Introduction, Washington, DC, USA: Gallaudet Univ. Press, 2011. |
| [2] | SIL MÉXICO, "Lengua de Señas en México," [Online]. Available: https://mexico.sil.org/es/lengua\_cultura/lenguas-de-señas-en-méxico. [Accessed 28 Nov 2024]. |
| [3] | G. d. M. Secretaría de Salud, "530. Con discapacidad auditiva, 2.3 millones de personas: Instituto Nacional de Rehabilitación," 28 Nov 2021. [Online]. Available: https://www.gob.mx/salud/prensa/530-con-discapacidad-auditiva-2-3-millones-de-personas-instituto-nacional-de-rehabilitacion?idiom=es#:~:text=En%20México%2C%20aproximadamente%202.3%20millones,ciento%20son%20niñas%20y%20niños.. |
| [4] | INEGI, "Población," 2020. [Online]. Available: https://www.inegi.org.mx/temas/estructura/. |
| [5] | N. Mohamed, M. B. Mustafa and N. Jomhari, "A Review of the Hand Gesture Recognition System: Current Progress and Future Directions," *IEEE Access,* vol. 9, pp. 157422-157436, 2021. |
| [6] | I. A. Adeyanju, O. O. Bello and M. A. Adegboye, "Machine learning methods for sign language recognition: A critical review and analysis," *Intelligent Systems with Application,* vol. 12, 2021. |
| [7] | V. Martínez-Sánchez, I. Villalón-Turrubiates, F. Cervantes-Álvarez and C. Hernández-Mejía, "Exploring a Novel Mexican Sign Language Lexicon Video Dataset," *Multimodal Technologies and Interaction,* 2023. |
| [8] | J. Forster, C. Schmidt, O. Koller, M. Bellgardt and H. Ney, "Extensions of the Sign Language Recognition and Translation Corpus RWTH-PHOENIX-Weather," in *International Conference Language Resources and Evaluation*, Reykjavik, Iceland, 2014. |
| [9] | J. R. González-Rodríguez, D. M. Córdova-Esparza, J. Terven and J. A. Romero-González, "Towards a Bidrectional Mexican Sign Language-Spanish Translation System: A Deep Learning Approach," *Technologies,* vol. 12, no. 1, 2024. |
| [10] | M. Cruz Aldrete, "Gramática de la lengua de señas mexicana," Oct 2008. [Online]. Available: http://elies.rediris.es/elies28/. [Accessed 28 Nov 2024]. |
| [11] | M. E. Rodriguez Trejo, O. Oubram and N. Lakouari, "Modelos Computacionales Aplicados a la Lengua de Señas Mexicana," 2023. |
| [12] | K. Mejía-Pérez, D.-M. Córdova-Esparza, J. Terven, A.-M. Herrera-Navarro, T. García-Ramírez and A. Ramírez-Pedraza, "Automatic Recognition of Mexican Sign language Using a Depth Camera and Recurrent Neural Networks," *Applied Sciences,* 2022. |
| [13] | G. García-Bautista, F. Trujillo-Romero and S. O. Caballero-Morales, "Meixcan sign language recognition using kinect and data time warping algorithm," in *International Conference on Electronics, Communications and Computers (CONIELECOMP)*, Cholula, Mexico, 2017. |
| [14] | C. p. P. y. E. l. D. d. l. CDMX, "Lenguas de señas son fundamentales para el desarrollo de las personas sordas y el acceso a sus derechos," 23 September 2019. [Online]. Available: https://copred.cdmx.gob.mx/comunicacion/nota/lenguas-de-senas-son-fundamentales-para-el-desarrollo-de-las-personas-sordas-y-el-acceso-sus-derechos. [Accessed 20 June 2025]. |
| [15] | T. Starner and A. Pentland, "Real-time American Sign Language Recognition from video using Hidden Markov Models," in *International Symposium on Computer Vision*, Coral Gables, FL, USA, 1995. |
| [16] | S. C. Dimri, R. Indu, H. S. Negi, N. Panwar and M. Sarda, "Hidden Markov Model - Applications, Strengths, and Weaknesses," in *International Conference on Device Intelligence, Computing and Communication Technologies*, Dehradun, India, 2024. |
| [17] | U. Farooq, M. Rahim, N. Sabir, A. Hussain et al., "Advances in machine translation for sign language: approaches, limitations, and challenges," *Neural Computing and Applications,* 2021. |
| [18] | F. Chollet, Deep Learning with Python, Second Edition, Manning Publications, 2021. |
| [19] | Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE,* vol. 86, no. 11, pp. 2278-2324, 1998. |
| [20] | K. O'Shea and R. Nash, "An Introduction to Convolutional Neural Networks," 2 Dec 2015. [Online]. Available: https://arxiv.org/abs/1511.08458. [Accessed 1 Dec 2024]. |
| [21] | O. Koller, S. Zargaran, H. Ney and R. Bowden, "Deep Sign: Enabling Robust Statistical Continuous Sign Language Recognition via Hybrid CNN-HMMs". |
| [22] | J. Huang, W. Zhou, H. Li and W. Li, "Sign Language Recognition using 3D convolutional neural networks," in *IEEE International Conference on Multimedia and Expo (ICME)*, Turin, 2015. |
| [23] | B. Mocialov, G. Turner, K. Lohan and H. Hastie, "Towards Continuous Sign Language Recognition with Deep Learning," in *Workshop Creating Meaning Robot Assistants*, 2017. |
| [24] | C. P. C. Lab, "OpenPose: Main Page," [Online]. Available: https://cmu-perceptual-computing-lab.github.io/openpose/web/html/doc/index.html. [Accessed 1 Dec 2024]. |
| [25] | F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C.-L. Chang and M. Grundmann, "MediaPipe Hands: On-device Real-time Hand Tracking," 2020. [Online]. Available: https://arxiv.org/abs/2006.10214. |
| [26] | Google, "Hand landmarks detection guide," Google, 21 May 2024. [Online]. Available: https://ai.google.dev/edge/mediapipe/solutions/guide. [Accessed 28 Nov 2024]. |
| [27] | I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, Cambridge, MA, USA: MIT Press, 2016. |
| [28] | I. T. Jolliffe, Principal Component Analysis, New York, NY, USA: Springer, 2002. |
| [29] | McInnes et al., "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction," *Journal of Open Source Software,* 2018. |
| [30] | J. L. Elman, "Finding Structure in time," *Cognitive Science,* vol. 14, no. 2, pp. 179-211, 1990. |
| [31] | S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation,* vol. 9, no. 8, pp. 1735-1780, 1997. |
| [32] | K. Cho, B. van Merrienboer, D. Bahdanau and Y. Bengio, "On the Properties of Neural Machine Translation: Encoder-Decoder Approaches," 7 October 2014. [Online]. Available: https://arxiv.org/abs/1409.1259. [Accessed 22 June 2025]. |
| [33] | M. Rodriguez, O. Oubram, A. Bassam, N. Lakouari and R. Tariq, "Mexican Sign Language recognition: Dataset creation and performance evaluation using MediaPipe and machine learning techniques," *Electronics,* vol. 14, no. 7, 2025. |
| [34] | A. Vaswani et al., "Attention is all you need," in *Advances in Neiral Information Processing Systems (NeurIPS)*, 2017. |
| [35] | Devlin et al., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *NAACL,* 2019. |
| [36] | C. D. Manning, P. Raghavan and H. Schütze, Introduction to Information Retrieval, Cambridge University Press, 2008. |
| [37] | O. Koller, N. C. Camgoz, H. Ney and R. Bowden, "Weakly supervised learning with multi-stream CNN-LSTM-HMMs to discover sequential parallelism in sign language videos," *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 42, no. 9, pp. 2306-2320, 2020. |
| [38] | Z. Cui, X. Liu and H. Zhang, "Deep learning-based sign language recognition using convolutional neural networks and LSTM," *Mutlimedia Tools Appl.,* vol. 78, no. 11, pp. 15137-15332, 2019. |
| [39] | N. C. Camgoz, O. Koller, S. Hadfield and R. Bowden, "Sign language transformers: Joint end-to-end sign language recognition and translation," in *IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, 2020. |
| [40] | M. E. Rodriguez-Trejo, O. Oubram, B. ALI and N. Lakouari, "Mexican Sign Language's Dactylology and Ten First Numbers - Labeled images and videos. From person #1 to #5," 30 May 2023. [Online]. Available: https://data.mendeley.com/datasets/5s4mt7xrd9/1. [Accessed 1 Dec 2024]. |
| [41] | M. E. Rodriguez-Trejo, O. Oubram, B. ALI and N. Lakouari, "Mexican Sign Language's Dactylology and Ten First Numbers - Labeled images and videos. From person #6 to #11," 30 May 2023. [Online]. Available: https://data.mendeley.com/datasets/67htnzmwbb/1. [Accessed 1 Dec 2024]. |
| [42] | OpenCV, "OpenCV - Open Computer Vision Library," [Online]. Available: https://opencv.org. [Accessed 28 Nov 2024]. |
| [43] | Scikit-Learn, "PCA," [Online]. Available: https://scikit-learn.org/dev/modules/generated/sklearn.decomposition.PCA.html. [Accessed 28 Nov 2024]. |
| [44] | Scikit-Learn, "KNeighborsClassifier," [Online]. Available: https://scikit-learn.org/dev/modules/generated/sklearn.neighbors.KNeighborsClassifier.html. [Accessed 28 Nov 2024]. |
| [45] | K. Cho et al., "Learning phrase representations using RNN encoder-decoder fir statistical machine translation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014. |

**APENDIX A. Título**

**APENDIX B. Título**

1. MediaPipe Hands sometimes, although rarely, fails to detect existing hands in a frame. In our case, out of the 3904 videos in our filtered dataset, on each of which we used the model to detect hands in the 12 middle frames, there were only 3 frames were no hands were detected and there should have been. We decided to drop the videos containing those frames, as their low number did not justify the investment needed to adapt our process to them. [↑](#footnote-ref-1)