Chair of Data Science

Isolated Sign Language Recognition Using Machine Learning Methods on Skeleton Data

Master's thesis by

Gaspard Serpinet

118544

1. Supervisor	2. Supervisor
Prof. Dr. Harald Kosch	Prof. Dr. Michael Granitzer

May 24, 2026

Contents

Abstract

Sign Language Recognition aims to automatically recognize signs performed by a signer. This work addresses practical constraints such as limited consumer hardware and the use of skeleton-only inputs.

A configurable pipeline is proposed for consumer hardware that allows for easy experimentation with different models, data augmentation techniques, and window selection strategies.

Multiple models, data augmentation techniques, and training configurations are systematically explored and compared to identify configurations that work best for skeleton-based Isolated Sign Language Recognition (ISLR).

Among tested architectures, Temporal Convolutional Network (TCN) outperforms Gated Recurrent Unit (GRU) and Multi-Head Attention (MHA) baselines in both accuracy and robustness to overfitting by achieving 65% top-1 test accuracy while training on a 2 GB GPU.

These results indicate that a modular skeleton-based pipeline, combined with appropriate temporal sampling and augmentation strategies, makes ISLR feasible under severe data scarcity and consumer hardware constraints and provides a practical basis for implementing and comparing new methods.

Acknowledgments

First and foremost, I would like to thank my thesis advisor Prof. Dr. Stephane Bres for inspiring many ideas and concepts explored in this thesis.

Sincere appreciation goes to INSA and the languages center for organizing French sign language courses every year for free for every student. Particular thanks go to the professors Natalie Matic and Laurine Pincon for teaching me sign language during two years and for their help in understanding the deaf and hard-of-hearing culture.

I wish to thank my family and my friends (particularly Myriam) for the discussions and feedback about this thesis, and explaining my work often helped clarify it for me.

The use of Perplexity and Cursor's language models is also acknowledged. These tools were occasionally used for code generation, brainstorming, and reformulation, but all design decisions, code, and written content were ultimately chosen by the author.

Finally, I am grateful for this thesis itself, which gave me the opportunity to carry out a large-scale project that made me learn a variety of things and that I could fully shape according to my own ideas.

List of Figures

List of Tables

1 Introduction

> The limits of my language mean the limits of my world.
>
> Ludwig Wittgenstein

Sign language serves as the primary communication medium for over 70 million deaf individuals worldwide. Yet, computer science has left this field underdeveloped, and unequal availability of machine learning tools could deepen existing inequalities.

1.1 Motivation

My personal motivation for this research originates from direct experience with French Sign Language (LSF) through two years of courses at INSA Lyon. This experience highlighted the practical challenges facing the deaf and hard-of-hearing community, particularly the lack of accessible technological tools for sign language learning and communication.

The absence of feedback applications for sign language learning creates a significant barrier. Students cannot easily verify the correctness of their signing without instructor feedback. This gap in available technology motivates the development of Sign Language Recognition (SLR) systems that can provide automated feedback and support language learning.

From a research perspective, this topic remains comparatively underexplored in machine learning, despite its strong societal relevance. Its interdisciplinary nature, at the intersection of computer vision, sequence modeling, and language processing, required synthesizing multiple fields of machine learning. The scarcity of existing work made the

learning experience even more valuable, pushing me to dive deep into novel problems and expand my technical knowledge across a wide spectrum of concepts and techniques.

This research could improve sign language accessibility by helping people learn sign language through an interactive feedback loop: if the system correctly recognizes a sign, it provides evidence that the sign was produced accurately. This changes both the learning approach and the expected quality of results by offering immediate, automated feedback instead of relying solely on instructor corrections.

1.2 Research Questions and Objectives

This thesis focuses on skeleton-based ISLR, where raw videos are first converted into sequences of body and hand Landmarks. Using skeletons instead of RGB videos reduces input dimensionality, improves privacy, and allows training and experimentation on modest hardware. Within this scope, the work addresses the following research question:

> How do different temporal architectures compare for skeleton-based ISLR under data scarcity and consumer hardware constraints?

This central question encompasses four specific research objectives, including making the approach practical on consumer devices:

1. **Pipeline Development**: Design a modular skeleton-based ISLR pipeline that processes raw videos into machine-learning-ready representations, supporting multiple datasets and configurations.
2. **Systematic Comparison**: Systematically explore and compare multiple models (e.g., recurrent, convolutional, attention-based), window selection strategies, and augmentation techniques to identify what works best for skeleton-based ISLR.
3. **Overfitting Mitigation**: Develop regularization and augmentation strategies to maximize performance on skeleton-based sign classification while addressing the fundamental challenge of $\approx 10^5$ features (see 2.92.9) with only ≈ 40 samples per class.

4. **Consumer Device Optimization**: Implement techniques to reduce pipeline training time and memory usage, enabling deployment on consumer hardware (e.g., 2 GB GPU).

1.3 Main Contributions

This thesis makes the following contributions to skeleton-based ISLR:

Modular Pipeline Architecture: An end-to-end pipeline from videos to trained models that supports class selection, skeleton extraction via MediaPipe, conversion to multiple formats, and fully configurable training.

Temporal Sampling Strategies: Six distinct window selection strategies including a *merge* strategy: iteratively averaging adjacent frames to reduce sequence length. This novel approach together with a clear API, enables systematic ablation on how frames are sampled from variable-length sequences.

Reproducible Experimentation: Comprehensive reproducibility via a uniform logging framework, allowing deterministic seeding, and saved configs for each run and supporting benchmark and future extensions.

Resource-Constrained Deployment: This thesis shows that the pipeline runs efficiently on a 2 GB GPU through careful memory and compute optimizations, enabling experimentation on consumer hardware.

Unified Model Interface: Fully configurable training script that simplifies adding new models and augmentation or cleaning techniques. In practice, swapping from one model to another or enabling a new augmentation usually amounts to changing a single configuration variable in the training script, making experimentation straightforward.

1.4 Thesis Structure

This thesis is organized following the IMRaD (Introduction, Methods, Results, and Discussion) structure to provide a systematic presentation of the research methodology and findings.

2 Background

> The more you know, the more you realize how much you don't know.
>
> Aristotle (attributed)

2.1 Sign Language Recognition Taxonomy

Sign languages are visual, gestural languages where meaning is expressed through handshapes, movements, facial expressions, and body posture. In this context, a *sign* can be either *static* (a single pose held for a short duration) or *dynamic* (a sequence of movements), and may encode a word, a phrase, or a more abstract concept.

There are no universally established definitions for all related tasks. This thesis follows the definitions proposed by Zhang et al. [ZJ24].

SLR: Recognition of sign language through machine learning methods. Researchers usually divide it into two main tasks: ISLR and Continuous Sign Language Recognition (CSLR).

ISLR: Recognition of isolated signs. Each video contains a single sign that is classified independently (often corresponding to one word).

CSLR: Recognition of continuous sign language sequences. The goal is to align a sequence of signs with the corresponding sentence.

Static Task: The task of recognizing Static Signs from a single image or pose

Dynamic Task: The task of recognizing Static Signs or Dynamic Signs from a sequence of images or skeletons

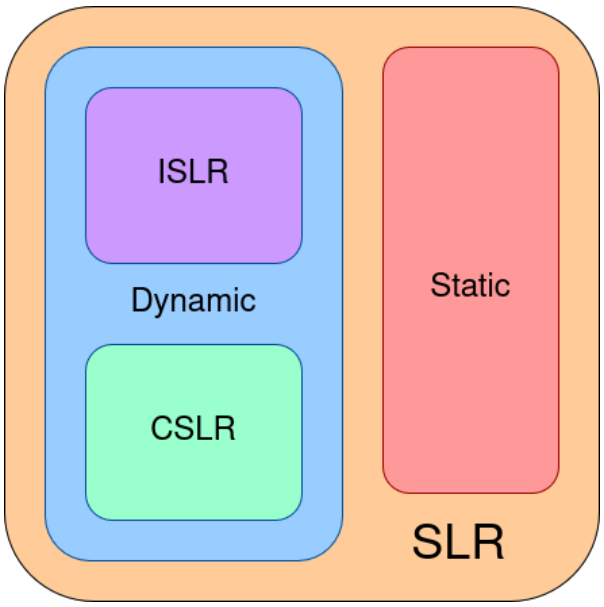

Figure 2.1: Venn diagram of SLR tasks definitions

ISLR focuses on recognizing a single sign from a sequence, treating each sign as an isolated unit.

CSLR addresses the recognition of continuous sequences of signs, it aims for the alignment of sequence between sign language and the corresponding class.

The definition of Sign Language Translation (SLT) is uncertain but according to [Li+25] it is the next step after CSLR, it aims for a general translation of the sentence and not the word by word translation.

Standard Evaluation Metrics

Researchers use several standard metrics to evaluate SLR systems:

- **Top-1 Accuracy**: This primary evaluation metric measures the percentage of correctly classified signs when considering only the highest-confidence prediction.
- **Top-k Accuracy**: Using typically k=5, this metric provides additional insight into model confidence by considering whether the correct class appears among the top k predictions.

- **Per-class Metrics**: Precision, recall, and F1-scores reveal performance variations across different signs. These metrics prove particularly important given the class imbalance common in sign language datasets.

2.2 Evolution of Input Modalities

The progression of SLR input modalities reflects broader advances in sensor technology and computer vision capabilities. Each modality addresses specific limitations while introducing new challenges.

Early SLR systems relied on intrusive sensor technologies. These included data gloves with embedded sensors and color markers attached to the hands [Ke+13]. While these approaches provided precise positional data, user intrusiveness, data scarcity, and calibration complexity severely limited their practical adoption.

Vision-based approaches using RGB videos subsequently emerged. However, these introduced several challenges: light conditions, camera quality, camera-signer relative position, signer characteristics, and other environmental sensitivity issues.

These limitations motivated the introduction of depth-enhanced recognition in 2010, followed by skeleton-based approaches in 2017.

The development of OpenPose [Cao+21] and MediaPipe [Lug+19] enabled a fundamental shift toward skeleton-based recognition. Laines et al. [Lai+23] catalyzed this transition by demonstrating that skeleton data alone could achieve competitive performance in ISLR.

2.3 Common Datasets

Sign language is different for each country and there are few datasets for ISLR, with most large-scale resources focusing on American Sign Language (ASL). Each dataset also comes with a particular way of splitting the data into training, validation, and test sets. Two common regimes are:

- **Signer-dependent** splits, where the same signer can appear in both training and test sets. This setting is easier but overestimates real-world performance.
- **Signer-independent** splits, where signers in the test set never appear in the training set. This is more challenging but better reflects practical deployment.

In the following, we briefly present the main datasets and then summarize the performance of representative models on each of them. Note that the reported numbers are not always directly comparable because authors use different splits (signer-dependent vs signer-independent) and metrics (e.g. top-k accuracy vs recall@k).

Word-Level American Sign Language dataset (WASL): average of 10 videos per class and 21k videos with 119 signers in a signer independent split [Li+20]. It is the most well known dataset in the literature.

Model	**Modality**	**Acc.**
Spatial-Temporal GCN (ST-GCN)	Skeleton	50.78%
Sign POse-based TransformER (SPOTER)	Skeleton	63.18%
I3D	RGB	65.89%
Sign Language Bart (SignBart)	Skeleton	78.00%
SignBERT(H)	Skeleton	76.36%
Sign Language BERT (SignBERT)(H+P)	Fusion	79.07%
SignBERT(H+R)	Fusion	82.56%
Dual-SignLanguageNet (DSLNet)	Skeleton	**93.70%**

Table 2.1: Performance of common models on WASL-100 dataset signer independent

Note: After the name of a dataset, you can sometimes see a number, it represents the number of classes in the dataset. WASL for example have 2000 classes by default but also exists in version 1000 or 100 classes. To take a reduction of a dataset, the classes with the most samples are kept.

Argentinian Sign Language dataset (LSA64): average of 50 videos per class and 64 classes with 10 signers. Argentinian Sign Language dataset [Ron+23]. It is one of the first dataset, containing only 64 classes and being one of the easiest. It is unknown if the dataset split is signer independent or dependent.

Model	**Modality**	**Params** (M)	**Top-1 Acc.**
SPOTER	Skeleton	5.92	**100.0%**
HWGAT	Skeleton	10.76	98.59%
ST-GCN	Skeleton	3.60	92.81%
Sign Language GCN (SL-GCN)	Skeleton	4.87	98.13%
3D Graph Convolutional Network (3DGCN)	Skeleton	-	94.84%
DSLNet	Skeleton	46.3	99.79%
SignBart	Skeleton	**0.75**	96.04%

Table 2.2: Performance of common models on LSA64 dataset

Note: The number of parameters of 3DGCN is unknown.

American Sign Language Citizen dataset (ASL-Citizen): average of 30 videos per class and 83k videos [Des+]. While not being the most well known, it is definitely the easiest to get (simple zip file). Also, it is not constituted only by professionals and represents better a real world scenario.

Model	**Modality**	**Metric**	**Score**
Inflated 3D ConvNets (I3D)	RGB	Accuracy	63%
I3D	RGB	Recall@10	**86%**
ST-GCN	Skeleton	Accuracy	59.5%
ST-GCN	Skeleton	Recall@10	82.7%
SignBart	Skeleton	Accuracy	**75.22%**

Table 2.3: Performance of common models on ASL-Citizen dataset

Microsoft American Sign Language dataset (MS-ASL): impossible to download the dataset and really few papers about it. 1000 classes with 222 signers and 25k videos with signer independent split [Joz]

American Sign Language Signs dataset (ASL-Signs): ∼400 signs per class, but only 22 signers in total and 100k skeletons (already data augmented)

Note: These tables data mostly come from [NT25].

2.4 Reference Models

Now that we have seen the major datasets in the literature and their evaluation protocols, we turn to the most representative models and how they differ in terms of input modality (RGB, skeleton, or fusion) and architectural choices.

I3D [Li+20] is an RGB-only model that inflates a 2D image into a 3D tensor by operating over space and time. It is pretrained on Kinetics and uses a 3D convolution on the RGB video.

SignBERT [Hu+21] treats hand pose at each frame as a visual token which embeds a gesture state with a Graph-CNN, a temporal index and the hand chirality. It is primarily a skeleton hands only method but can optionally be a fusion approach: hands Landmarks + RGB or hands Landmarks + full-body skeleton.

ST-GCN [WZA22] uses a ST-GCN architecture on skeleton only. It combines spatial and temporal information into a single matrix by doing a Depth-first Traversal of the skeleton graph to make one axis and the time to make the other. Then it applies a GCN on the matrix.

SPOTER [BH22] compact transformer encoder that use only skeleton 2D body+hands coordinates. It uses a sophisticated pose normalization that separates hand and body coordinates systems and a joint rotation augmentation to improve the accuracy. They showed how augmentation and normalization can improve accuracy.

Multi-Stream Neural Network (MSNN) [Mar+24] is a multi-stream neural network that uses three streams: RGB full frame, local image stream (cropped hand and face regions) and skeleton stream (body and hands) from pose estimation.

Unified Sign Language Recognition (Uni-Sign) [Li+25] is a fusion approach combining RGB and pose with Prior-Guided Fusion (PGF). PGF is a fusion module that uses attention mechanism to inject RGB information into pose features. It improves accuracy and is an efficient fusion approach.

DSLNet [Liu+25a] use a dual-stream ST-GCN architecture. One stream use isolated hand landmarks and the other use the normalized entire body.

SignBart [NT25] is skeleton only method using MediaPipe's holistic and BART encoder-decoder on x and y coordinates independently. It is very lightweight and uses 0.75 to 3.8M parameters depending on the number of classes.

2.5 Data Augmentation and Notable Techniques

As the data is few-shot and collecting new annotated videos is costly, data augmentation becomes a key tool to improve model performance and robustness without acquiring additional real-world samples.

SPOTER [BH22] introduce some data augmentation techniques on skeleton landmarks that simulate camera movements while preserving semantic meaning. It includes an in-plane rotation, squeeze and a perspective transformation. Additionally, they implemented a sequential joint rotation which simulates slight variances in the execution of the sign.

Skeleton-Based Isolated Sign Language Recognition with Part Mixing (SKiM) [Lin+24] is a data augmentation technique that swaps the corresponding Landmarks within one region (e.g., hand) between two randomly selected samples and combines their labels linearly as the new label. Even with only 2 values of percentage of swapping (let say 5% and 10% of class B and 95% and 90% of class A), it squares the number of samples.

$$2 \times \binom{2}{n} = 2 \times \frac{n(n-1)}{2} = n^2 - n = \mathcal{O}(n^2)$$

Roh et al. [Roh+] focus on robust preprocessing of Landmarks extracted with MediaPipe Holistic. For missing Landmarks (especially hands), they use bilinear interpolation to reconstruct the Landmarks.

Adversarial Vulnerability-Seeking Networks (AVSN) [NJ25] treats data augmentation for SLR as an adversarial game between two networks: the generator and the discriminator. The generator takes real skeletons as input and learns to output perturbed versions that remain plausible human motion but are intentionally difficult for the recognition model to classify correctly.

$$\mathcal{L}_G = -\mathbb{E}_{(x,y),z}[\ell_{\text{CE}}(D_\theta(\tilde{x}), y)] + \lambda \mathcal{L}_{\text{reg}}(x, \tilde{x})$$

The first term makes synthetic sample adversarial (opposite of the loss of the discriminator) and the second is a distance from original and generated skeletons. E.g. mean-squared error over joint coordinates, velocity or acceleration constraints, bone lengths, time ranges, ...

Conversely, the discriminator (a standard skeleton-based classifier) is trained jointly on raw and generated samples, forcing it to become robust against such targeted perturbations while still preserving high accuracy on clean data.

Joint Mixing Data Augmentation (JMDA) [XW25] is a mixing strategy between two random skeleton sequences. It uses a spatial mixing (SM) and a temporal mixing (TM) to create a new motion patterns that maintain realistic human motion.

2.6 Topic Limitations

Recognizing signs in realistic conditions requires accounting for multiple sources of difficulty that directly impact model performance. We group these limitations into dataset-related issues and task-specific challenges.

Dataset Limitations: When using a subset of the dataset, the available classes are really strange. In citizen-50 for example, there are words like HALLOWEEN, LETUCE and CANCER which are not necessarily representative of high-frequency everyday vocabulary. When using the full dataset or a big subset, you get an imbalanced dataset with really few samples per class which can lead to statistical issues.

Signer Diversity: This represents the most significant limitation across existing datasets. Most resources suffer from limited signer participation per class, severely hampering the development of Signer-independent systems. This limitation proves particularly problematic because sign language exhibits significant individual variation in execution style, speed, and spatial extent.

Data Accessibility: Distribution challenges undermine research reproducibility. Many large-scale datasets face distribution difficulties, relying on web scraping methods that become obsolete over time rather than providing stable download links.

Vocabulary Coverage: Even the largest datasets remain limited relative to natural sign language use. They represent only a fraction of vocabulary used in everyday deaf community communication.

Annotation Quality: Quality varies significantly across datasets. Many datasets lack consistent quality control and limited verification of sign production accuracy by native signers.

Sign language complexity

Sign language uses a huge variety of ways to transmit information, sometimes the information is carried by position, sometimes by movement, sometimes by both. Sometimes the information is manual, sometimes it is carried by the face or the body. Some signs like the ones who have letters are identical for right- and left-handed signers, while others display symmetry. The same sign can produce a different meaning depending on the relative position of the signer and the viewer.

ISLR specific limitations

The ISLR signer independent task is more complex due to the tendency for the differences between signers like morphology (hand/body size and shape). But the biggest challenge across signers is the way of signing or the dialect that changes the number of repetitions, speed or even give a completely different sign.

The image quality and frame rate of the videos can also affect skeleton extraction and therefore the recognition.

CSLR limitations

The difficulty of CSLR is the same as translating from one written language to another: Some words from the language A can be translated to two words in language B (like drinking have nearly as many signs as the number of possible drinks), or the order of the words can be different (in French sign language the order is time, location, subject, action).

Another difficulty is putting boundaries between words, the movement between two signs should not be interpreted as a sign and can modify the start or end of a sign.

Note: As the task of CSLR is really complex and depends on the task of ISLR, we will limit the scope to ISLR in this thesis.

2.7 Approaches Comparison

There are multiple ways to represent and process sign language videos. At the lowest level, one can work directly with raw RGB frames or short video clips. Alternatively, it is possible to extract more compact intermediate representations, such as body and hand skeletons, and use these as the main input to the model. Finally, hybrid approaches combine both sources of information (e.g., RGB and skeletons) in a fusion architecture.

Intuitively, recognition is driven by the motion and configuration of the signer's body and hands over time. Skeleton-based representations explicitly encode these movements, while RGB-based methods must learn to infer them from pixel values. In this section, we summarize the main trade-offs between the most common approaches.

Approach	**Benefits**	**Drawbacks**
RGB-based	+ well-established methods	− capture unnecessary information
	+ transfer learning possible	− bigger model ($\approx 10\times$)
Skeleton-based	+ computational efficiency	− pose estimation bottleneck
	+ privacy preservation	− occlusion sensitivity
	+ environmental robustness	− small literature in holistic skeleton
	+ smaller dataset size	− less precision of hand movements
Fusion-based	+ sometimes increase performance	− bigger and more complex models

Table 2.4: Comparison of RGB-based, skeleton-based, and fusion-based approaches

With all these reasons and because we want an efficient approach that runs on consumer hardware, we will focus on skeleton-based approaches.

2.8 Choice of Skeleton Model

Building on the comparison of input modalities and our focus on skeleton recognition, we must now choose a concrete pose estimation model that will provide the body and hand landmarks used by the rest of the pipeline.

Requirements

There are a lot of body pose estimation models, but really few ones are Holistic. Most approaches estimate only body pose or a subset of joints, whereas only a few models provide holistic body, hand, and face landmarks in a single pipeline, such as OpenPose [Cao+21] and MediaPipe Holistic [Lug+19].

SignBart [NT25] used body, right- and left-hand landmarks and showed that enriching a body-only skeleton with detailed hand landmarks substantially improves accuracy on LSA64 and other datasets. Of course, using an open-source model that is well-documented and easy to use is preferable. Having the model on GPU is also a plus.

MediaPipe was selected over **OpenPose** based on comprehensive evaluation across multiple criteria:

1. It is already used in the literature by SignBart [NT25] for example.
2. It is computationally efficient, easy to use and comes with a python library.
3. MediaPipe holistic allows to get all body parts in a single forward pass.
4. The coordinates are normalized and there is a visibility score for each landmark.
5. MediaPipe comes with a python library, no repository is needed.

2.9 Skeleton Representation

Now that we have chosen the model to extract the skeleton, let's see how the data is represented.

The skeleton is in 4 parts: body, left hand, right hand and face.

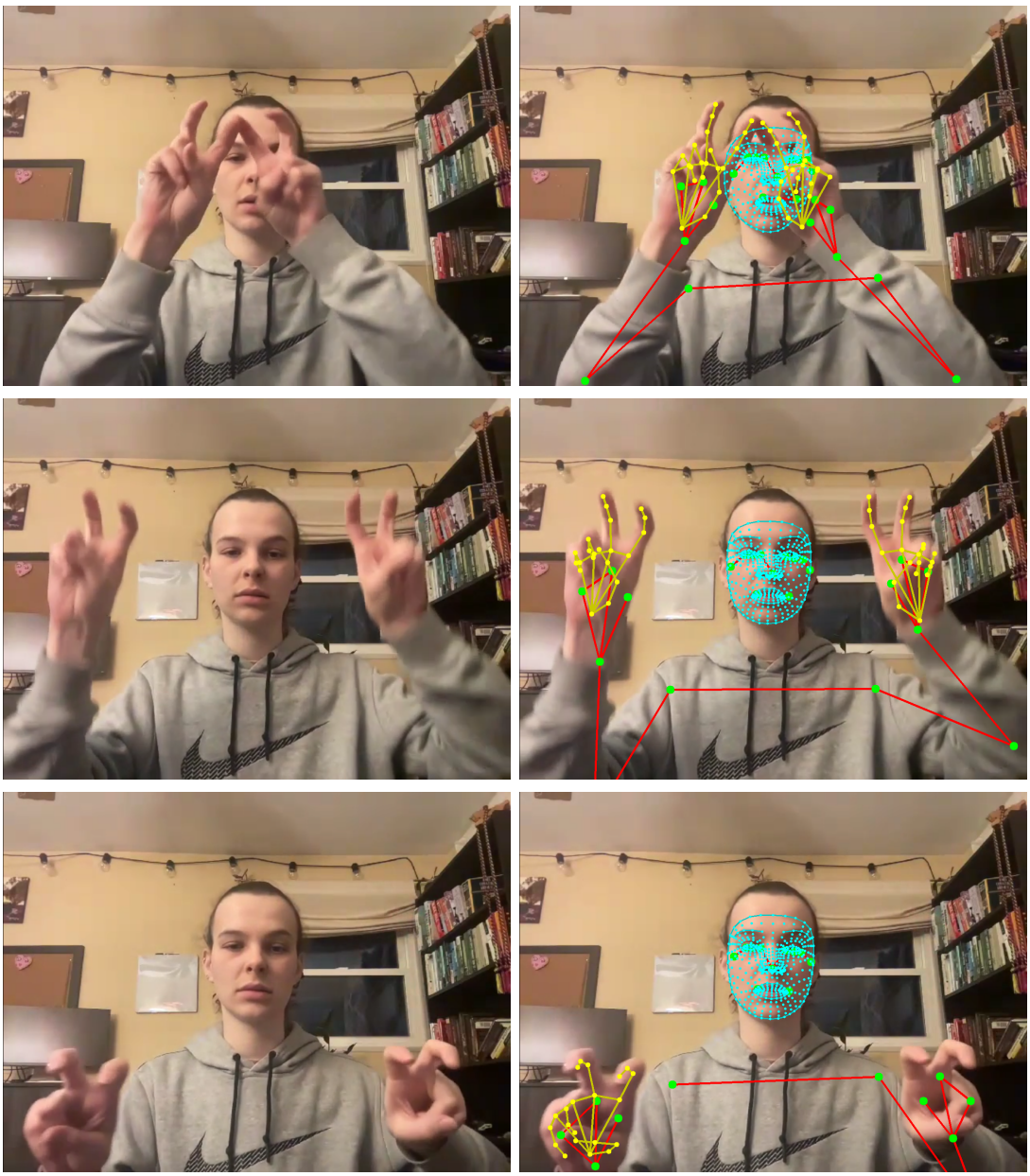

Figure 2.2: Video vs. skeleton representation comparison

A Landmark corresponds to a point in the skeleton, it can be 2D or 3D coordinates and includes a visibility score. To convert an image to a skeleton, we need to extract the landmarks from the image. To convert a video to a skeleton, we just iterate over the frames and extract the landmarks. It is possible to take advantage of the previous frame to help the extraction of the landmarks.

Data representation

MediaPipe's holistic pose estimation [Lug+19] provides a comprehensive skeletal representation particularly well-suited for sign language recognition. The framework extracts multi-modal Landmarks that capture the essential information of sign language communication.

Each temporal sequence consists of multiple frames. Each frame contains:

- **Body Pose**: 33 Landmarks with 3D coordinates plus visibility scores ($33 \times 4 = 132$ values)
- **Hand Landmarks**: 21 Landmarks per hand with 3D coordinates plus visibility ($2 \times 21 \times 4 = 168$ values)
- **Facial Landmarks**: 468 Landmarks with 3D coordinates plus visibility ($468 \times 4 = 1872$ values)

This representation yields a high-dimensional feature space. Most of the videos are 2-second sequences with around 100 frames:

$$\text{features} = \text{number of frames} \times \text{features per frame} = 100 \times \big(132 + 168 + 1872\big) = \approx 10^5$$

10^5 features is a lot, that's why the visibility is removed as it is just an indicator of the reliability of the landmark. Also, the z-axis and head are optionally removable.

It's makes $100 \times \big(66 + 84\big) = \approx 10^4$ features which is much lower.

MediaPipe normalizes all coordinate values to the range [0,1] relative to the input image dimensions, with (0,0) representing the top-left corner. This normalization provides scale invariance across different video resolutions.

2.10 Data Characteristics

The Holistic skeleton representation create some challenges:

The data show a significant number of challenges: starting with being **high-dimensional**, while being **few-shot**. Moreover, the data augmentation might not be the ultimate solution because it does not address the signer diversity difficulty.

The data is **temporal** and with inconsistent number of frames and inconsistent number of features per frame. On top of this, the precision can be important for some features (like the hand Landmarks).

Bone length, number of repetitions, speed, etc. can vary across signers. That's why the Signer-independent task is more challenging than the Signer-dependent task.

2.11 Methodological Gaps

Limited Data Augmentation Strategies: Skeleton-based ISLR lacks well-developed augmentation strategies. While researchers have established RGB augmentation techniques, skeleton-specific augmentation that preserves sign language semantics requires specialized approaches.

Cross-Signer Generalization: Current methods inadequately address this challenge. Most show significant performance degradation when evaluated across different signers.

2.12 Technical Opportunities

Multi-Scale Temporal Modeling: This approach presents opportunities for capturing both fine-grained hand movements and coarse body positioning within unified architectures.

Uncertainty Quantification: Providing confidence estimates could improve practical deployment for assistive technology applications.

Efficient Architecture Search: Systematic exploration may identify optimal model architectures for the unique constraints of skeleton-based ISLR, balancing accuracy with computational efficiency.

This comprehensive background establishes the theoretical foundation and current state-of-the-art necessary for understanding the methodological contributions presented in

subsequent chapters. The identified challenges and research gaps directly motivate the modular pipeline architecture and ensemble approaches detailed in the following Methods section.

3 Methods

> All models are wrong, but some are useful.
>
> George E. P. Box

This chapter presents a comprehensive methodology for skeleton-based ISLR that addresses the fundamental challenge of high-dimensional feature spaces with limited training data.

The approach centers on a modular data routing that selects and transforms videos into skeleton data and converts them to optimized formats. This is followed by a highly configurable training script that performs data loading, augmentation, window selection, model training, and evaluation. During training, each sample is processed through a window selection strategy and window augmentation to produce fixed-length sequences. Everything is designed to run on consumer hardware.

3.1 Data Routing Overview

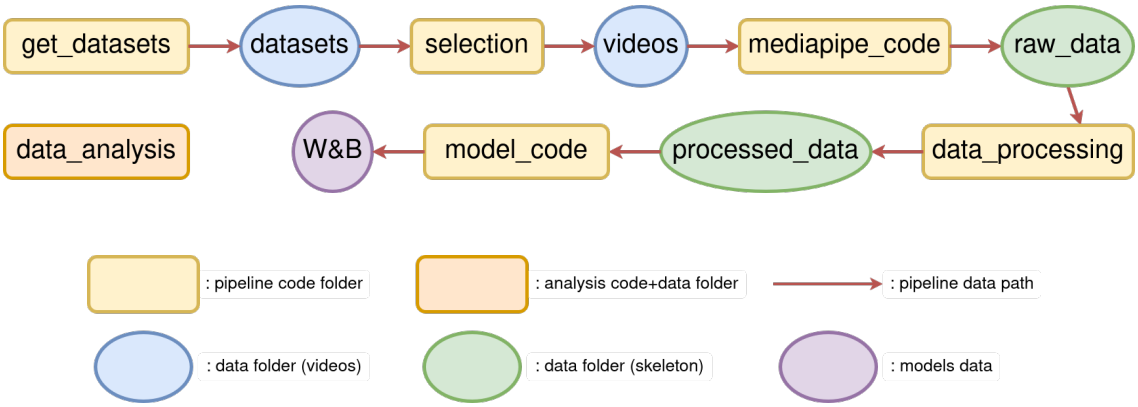

Figure 3.1: Data routing from datasets acquisition to training script

Data Format: JSON files are converted into PyTorch tensors and saved in NPZ, PT, H5, or Zarr formats (NPZ serves as the default format). Each format stores identical data, only performance characteristics differ.

The dataset can be saved with either 2D (x, y) or 3D (x, y, z) coordinates.

It was observed that head Landmarks comprised more than 80% of the data. Therefore, head-specific Landmarks were removed (the body pose retains a few head Landmarks). Visibility scores were also removed and z-coordinates were made optional. This yields $(33 + 21 \times 2) \times 2$ or $3 = 150$ or 225 coordinates per frame.

3.2 Datasets Acquisition

The following datasets are used: WASL and ASL-Citizen. Data acquisition consists of two steps: the datasets are downloaded, then a `metadata.csv` is created. The goal of **get_datasets** is to uniformize the data from the datasets and make it ready to be used in the pipeline.

Acquiring the dataset was surprisingly more difficult than expected. They are often constituted of a simple repository with a JSON containing links of videos and a script to download them. Because of that, it was often difficult to download the datasets. For

example, MS-ASL could not be downloaded because the script was not working. Another repository online claimed to fix the issue, but it did not work either. For WASL, most videos were downloaded but not all of them. And for ASL-Citizen, it was a zip file, so everything went smoothly.

3.3 Class Selection Process

Some datasets are named with a suffix (e.g. WASL-100, WASL-2000) indicating the number of videos per class when selecting the top N represented classes.

Using **data_analysis**, information about the dataset can be obtained and the top N represented classes can be selected.

Then in **selection**, all data elements that correspond to the given classes are retrieved and moved to form a new dataset.

The ASL-Citizen-50 dataset comprised **51 classes with** ≈ 40 **samples each**, totaling $\approx 2,000$ video samples with stratified splits: 70% training, 10% validation, 20% test. This implies that each class has roughly 28 training, 4 validation, and 8 test videos on average. Although four validation samples per class is relatively small, increasing it would further reduce the amount of training data, so this split was kept as a practical compromise between reliable validation and reasonable training set size.

3.4 MediaPipe

MediaPipe uses the following models to extract the skeleton:

- **BlazeFace**: Facial landmark detection providing 468 Landmarks
- **BlazePose**: Body pose estimation providing 33 Landmarks
- **BlazePalm and HandLandmark**: Hand pose estimation providing 21 Landmarks per hand

MediaPipe produces a structured JSON output containing video metadata and frame-by-frame landmark data.

MediaPipe generates the skeleton frame by frame, but there is an option to use neighboring frames to help the model predict the skeleton, as described in the original framework [Lug+19].

The detection and tracking confidence were set to 0.2 to balance the number of detected landmarks and their reliability. The model complexity was set to 1 because, on the available hardware, higher-complexity variants were substantially slower while providing no clearly observable qualitative improvement in landmark precision. A more systematic quantitative study of these parameters (e.g. measuring downstream accuracy as a function of confidence threshold and complexity) would be valuable future work but was beyond the scope of this thesis.

3.5 Dataset API

The result from MediaPipe is a list of JSON files, which are human-readable but expensive in terms of storage and loading time. To address this, a dataset API was created. JSON parsing is accelerated using cysimdjson, an optimized SIMD-based parser that significantly reduces load times. For storage, the dataset can be converted into NPZ, PT, HDF5, or Zarr. NPZ was chosen for storage optimization, simplicity and legacy reasons.

Format	**Size** (MB)
JSON	7300
Zarr	284
PT	53
NPZ	25
HDF5	21

Table 3.1: Storage size of the different formats for 2D no head citizen-50 dataset

The code also converts from Parquet format to comply with the ASL-Signs dataset.

Beyond storage reduction, the dataset API allows keeping or removing head Landmarks and choosing 2D or 3D coordinates.

The decision was to remove the head Landmarks because it multiplied the input data size by 7 even though Nguyen and Tran showed on LSA64 that enriching a body-only skeleton with detailed hand Landmarks improves accuracy from 86.97% to 96.04%. Other work on Arabic SL reports that combining hand and face Landmarks improves accuracy by about 4% compared to hand-only Landmark input.

The conversion also ensures frame size consistency: All missing values are filled by zero. Depending on the number of dimensions of the coordinates, all frames in the dataset have 150 or 225 features per frame.

3.6 Window Selection Strategies

The videos in the datasets have a varying number of frames, sometimes ranging from short clips to much longer sequences. However, most of the models tested in this thesis cannot directly accept variable-length inputs, and even for models that technically can (e.g., recurrent networks), training with arbitrary sequence lengths would be inefficient and difficult to tune. To keep training time manageable and make models comparable, we therefore normalize all samples to a fixed temporal length.

In the next sections, we will make a distinction between `num_frames`, which is the number of frames present in the original video or in the `raw_data`, and `window_size`, which is the length of the sequence after putting all samples to a constant length through a window selection strategy.

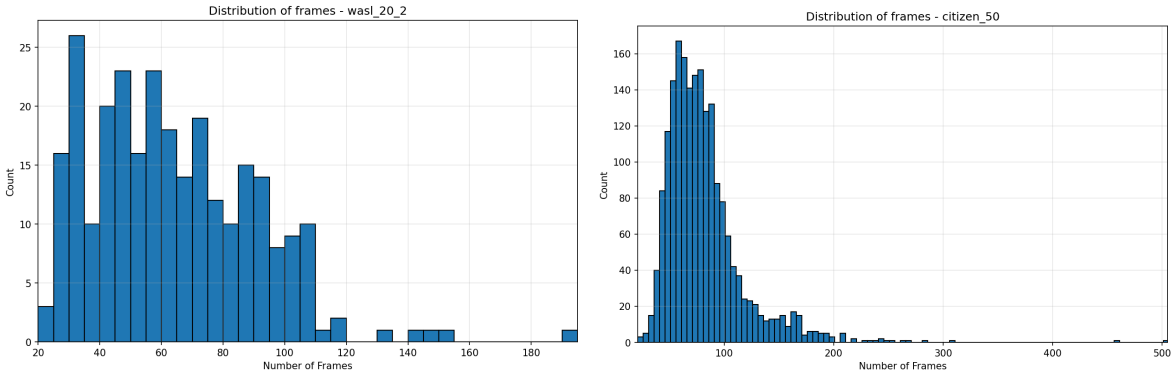

Figure 3.2: Distribution of the number of frames per video in WASL-20 and ASL-Citizen-50

Both datasets exhibit substantial variability in sequence length, motivating the use of window selection strategies to normalize inputs. For this reason, we implemented six ways to select the frames.

Uniform: random frame sampling, **Gaussian**: sampling weighted by a Gaussian PDF centered on the sequence, **beta**: asymmetric sampling via a beta distribution with parameters α and β, **center**: downsample by a reduction factor, then take the center window, **merge**: iteratively average adjacent frames until the target window size is reached and **sliding**: random start position during training and then select the contiguous frames, it allows keeping a constant signing speed.

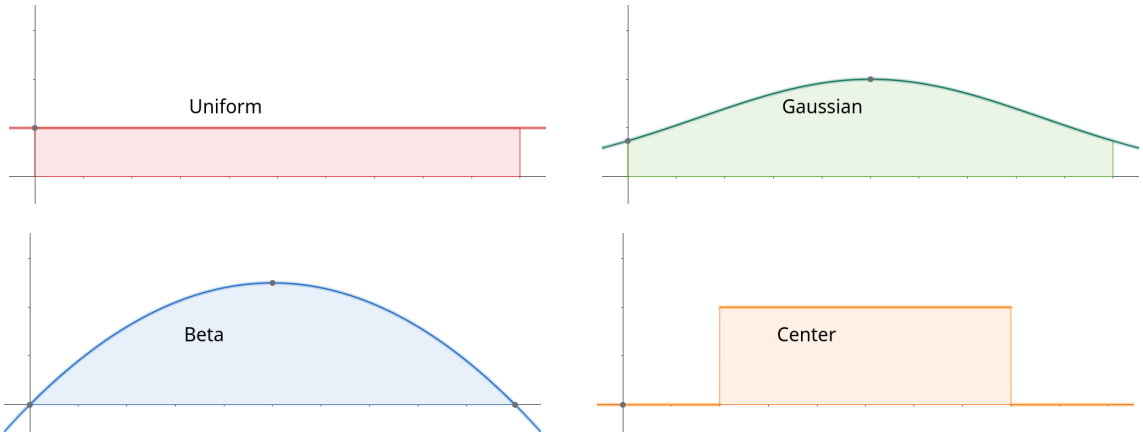

Figure 3.3: Distributions of frame selection strategies

These parameters can be different for each set. In all cases, the goal is to obtain fixed-

length sequences while preserving as much discriminative temporal information as possible.

3.7 Window Augmentation Methods

Reducing the number of frames is good, but there is a high standard deviation in the number of frames. The minimum number of frames in citizen-50 is 30 which is far bellow the average of 100 frames. That is why we also augment the number of frames.

When the original number of frames is lower than the target window size, multiple window augmentation techniques were implemented: **zeros** (center the sequence in a zero-padded tensor of length `window_size`), **copy** (repeat frames to reach the target length), and **interpolation** (nearest, linear, or bicubic) which uses previous and next frames to synthesize in-between frames. Note that according to the PyTorch documentation, linear and bicubic interpolation can be inaccurate when using float16 on CUDA.

3.8 Landmarks Augmentation

Now that we have a constant number of frames, we can augment the landmarks. We choose that even after augmentation, all the samples should still form a valid pose. That's why we restrain ourselves to camera movements and video to skeleton conversion imprecision.

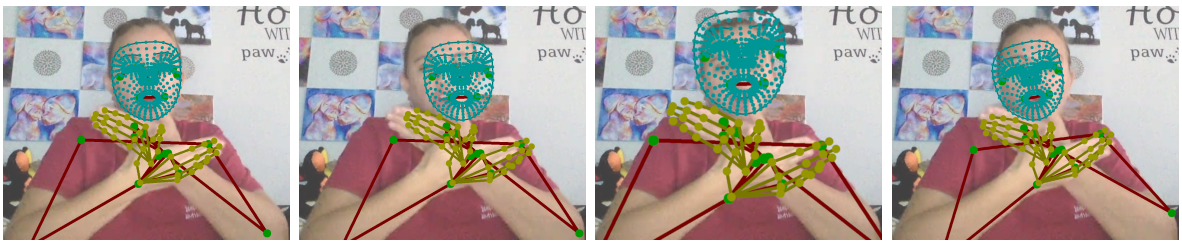

Figure 3.4: Landmarks no change, translation, scaling and rotation

Window selection acts as a form of data augmentation, with window-level techniques preserving temporal coherence. Landmark augmentation can also be applied.

Global translation, scaling, rotation are applied to the whole frame uniformly. The distribution used for the transformation is uniform.

Gaussian noise is also added individually to the values in the data, meaning that each number in a data element can be affected by the noise independently. It simulates the imprecision of MediaPipe converting the video to a skeleton.

For every transformation, there are parameters to control the intensity of the transformation and also a probability of applying the transformation. The values best values of each parameter will be found in a dedicated sweep in Section 4.7 4.7. Four presets: `none`, `light`, `medium`, `strong` bundle these parameters for quick experimentation.

No vertical flip was applied because some signs are not symmetrical, if they contain a letter or a number for example. It was decided to not check which sign can be flipped and which not and that no data augmentation should create a sign that doesn't exist.

There is also a dropout given by Skorch to improve the robustness of the model.

3.9 Models

The main implemented models were (in order of implementation): Linear (Lin), Long Short-Term Memory (LSTM), GRU, MHA, Transformer architecture (Transformer), TCN.

All these architectures share a common PyTorch base class exposing a uniform interface. This design makes it straightforward to add new models or switch between existing ones by simply changing the model name in the training configuration, without modifying the rest of the pipeline.

Lin, LSTM, and Transformer are well-established and not described in detail. Just note that GRU works similarly to LSTM but uses fewer parameters for similar performance. And MHA is a component of the Transformer.

TCN uses 1D causal convolution across time to avoid using future frames for predictions.

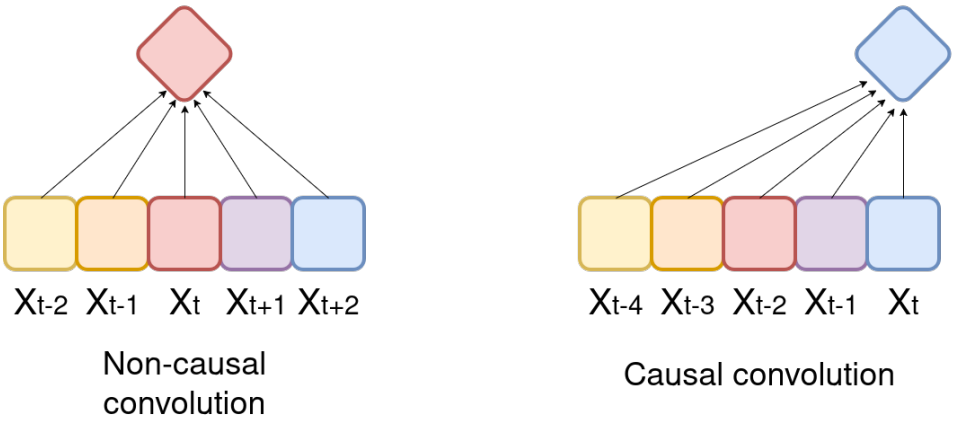

Figure 3.5: Causal vs non-causal convolution

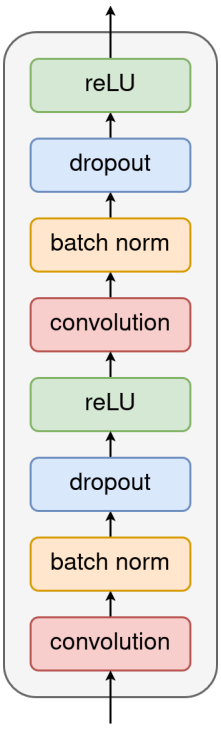

Figure 3.6: Architecture of a TCN temporal block

The causal convolution is the backbone of the temporal blocks. Which are stacked multiple times, each with a dilation factor of 2^{k-1} where k is the level index.

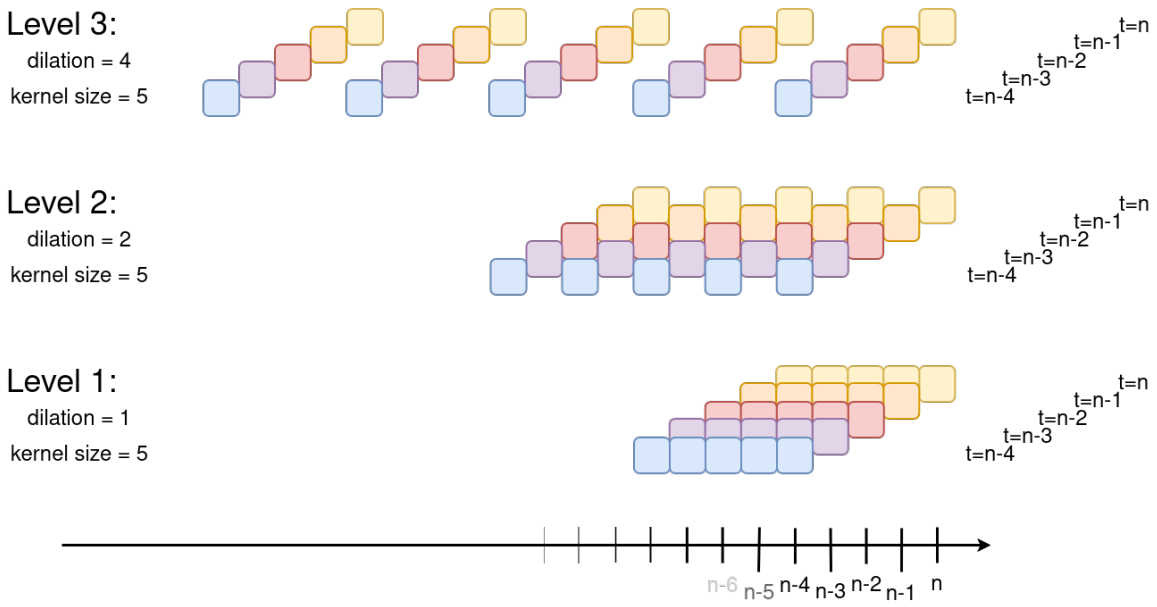

Figure 3.7: TCN levels with dilation factors

3.10 Training Script

Every parameter or variable is logged in W&B to ensure reproducibility. Optimizations are applied depending on the Compute Capability (e.g. CUDA Deep Neural Network for convolution, TF32 on compatible GPUs). The training uses an `AcceleratedNet` that wraps Skorch's `NeuralNetClassifier` with Hugging Face Accelerate, enabling mixed precision (bfloat16 or float16) and gradient accumulation for effective batch sizes larger than GPU memory allows. After instantiating precision and transform methods, the dataset is loaded and converted into tensors. The Skorch net is instantiated with 50+ configurable parameters, followed by the training loop and evaluation.

The code is highly configurable: changing variables in W&B updates the Skorch net immediately. **Comprehensive Seeding**: All random operations deterministically seeded.

W&B logs the git repository and the commit hash, so it's easy to reproduce the experiment.

Two possible data splitting approaches are employed: Either, the splits provided by the original dataset are used (stratified 70%-10%-20% train/validation/test split for

citizen-50) Or random splits are created without signer independence (with stratified configurable split ratios)

3.11 Combining Mechanisms

Not all frames are used in training, but more frames can be used in testing. To take advantage of this, multiple combining mechanisms were designed:

- **simple weighted sum**: gate_prob = $\frac{\text{num\_classes} - \text{perplexity} - 1}{\text{num\_classes}}$
- **threshold** on gate_prob and then sum
- **Mixture of Experts (MoE) approach**: learn gate_prob to do weighted sum knowing data (train by adding Laplace/Gaussian noise)
- **threshold-based adaptive method**: if max(confidences) $\geq$ threshold: use only the most confident classifier else: weighted_combination
- **gate_prob** correlated to where there is the most movement

These mechanisms are designed but not integrated in the final pipeline. If implemented, they would enable the use of a sliding window at inference time: when the input sequence is longer than `window_size`, multiple overlapping windows are processed and their predictions combined (e.g. by weighted sum) instead of using a single window.

3.12 Hyperparameter Optimization

The fully configurable training script with 100+ hyperparameters has a drawback: finding the best set of hyperparameters values is a challenge.

To find the best values, W&B's sweep was used, along with an LR scheduler and early stopping.

Search Strategy: Bayesian optimization balancing exploration and exploitation while minimizing computational waste. It's like random search, but it does not explore all the space, it is looking for the best hyperparameters.

A maximum of 200 epochs was configured, with an LR scheduler that divides the LR by 2 if no valid_acc improvement over 8 epochs, stop if no valid_acc improvement over 25 epochs, and stop if valid_loss is not improving over 20 epochs.

Combinatorial Explosion Management:

The total number of experiments equals the number of datasets × the number of window sizes × window augmentations × window selections × landmark augmentations × models × model hyperparameter values × learning rates × effective batch sizes × training hyperparameter values × other forgotten hyperparameter values, which evaluates to about 10^9 combinations.

This search space is enormous. Aggressive pruning and assuming some hyperparameters are uncorrelated (e.g., separating window selection/augmentation from landmarks augmentation or assuming that data should be treated the same for every model) limit actual evaluations to manageable numbers.

3.13 Implementation and Hardware Constraints

One of the main challenges was to run the code on consumer hardware (8-year-old laptop with 2 GB of GPU memory and Compute Capability 6.1).

Making all the computations on the CPU was too slow, so the GPU had to be used. Its limited memory made it challenging to fit the data and run efficiently. Other problems were encountered due to a very recent CUDA version and a very old GPU, so finding the right PyTorch version was difficult and using cuML revealed to be impossible.

The code was ultimately run successfully on the available hardware. While it may work on other configurations, there was an effort to make it usable on different hardware.

Convolution and Optimizations

A performance issue was encountered: training time increased tenfold for unknown reasons. Enabling CUDA Deep Neural Network (an optimized CUDA library) and its benchmark parameter helped but wasn't enough, then Torch Profiler and TensorBoard were used to locate the bottleneck.

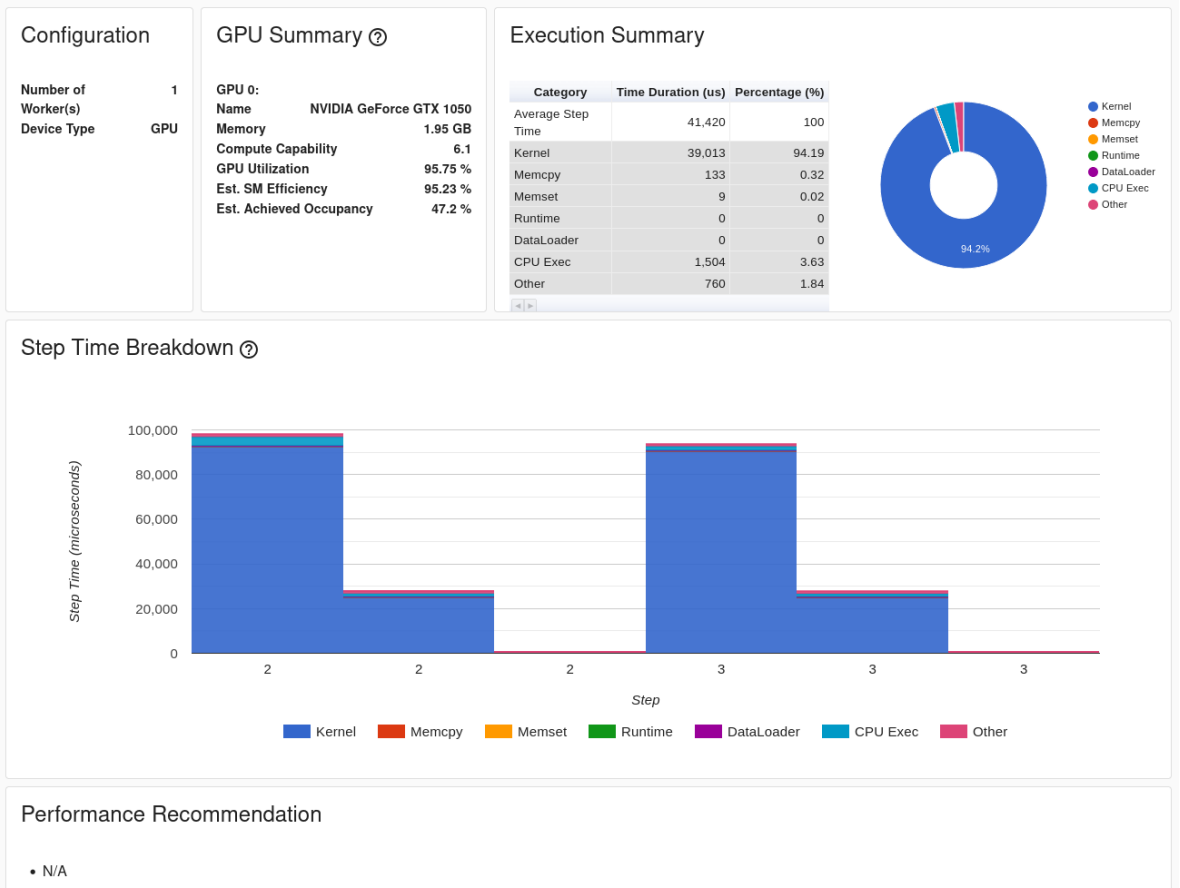

Figure 3.8: TensorBoard overview

As we can see, the bottleneck does not reside in the CPU computations or memory transfers, but in the GPU.

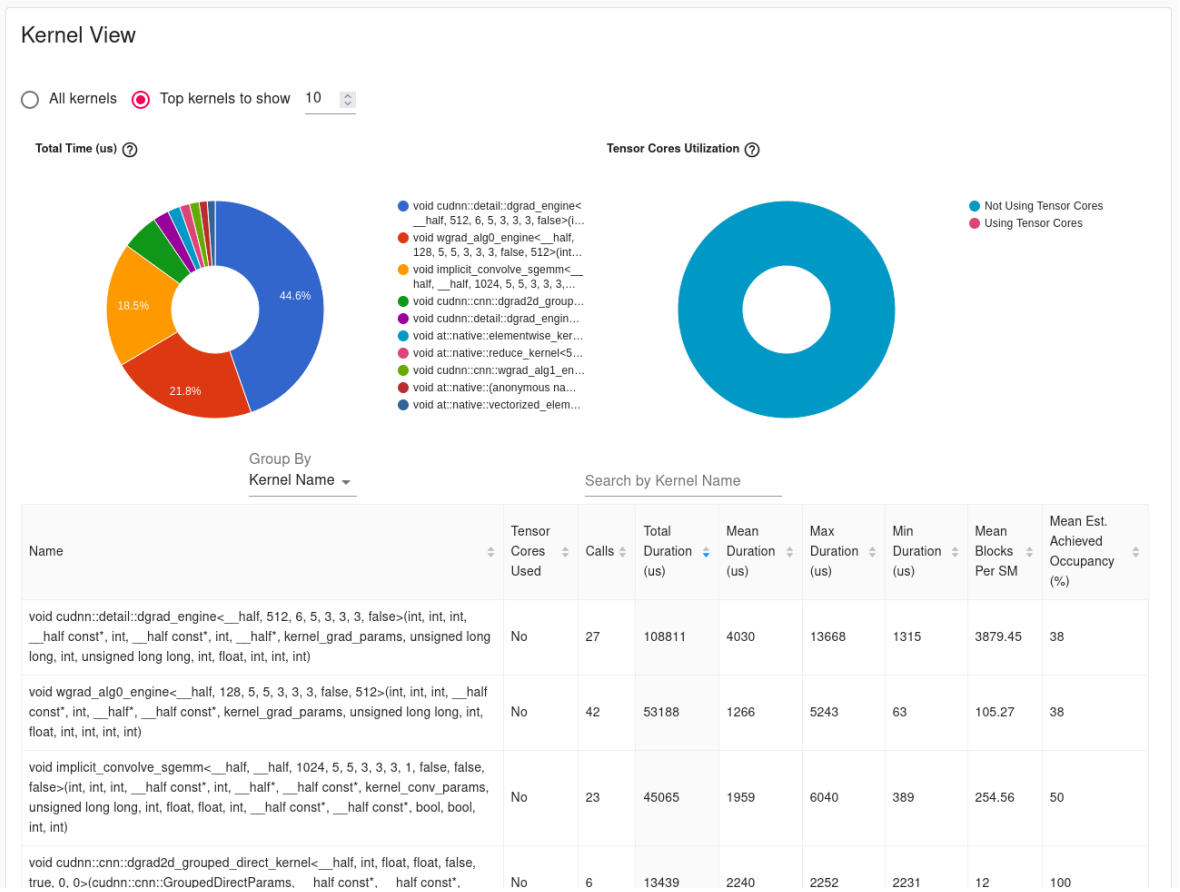

Figure 3.9: TensorBoard kernel view

GPU usage of 96% with an SM efficiency of 95% is excellent. The `DataLoader`, `Memcpy`, and CPU execution are tiny compared to 94% of time used for kernel execution. It can be concluded that training is compute-bound on convolution and not blocked by data loading or Python overhead. Unless convolution can be sped up, training time cannot be reduced much further.

Convolution cost roughly scales with:

$$\text{FLOPs} \propto B \cdot C_{\text{in}} \cdot C_{\text{out}} \cdot K \cdot L_{\text{out}}$$

Where B is batch size, $C_{\text{in/out}}$ channels, K kernel size, L_{out} output length.

The hidden size was reduced as it divides the convolution cost by a squared factor. The kernel size was also reduced as using dilated smaller kernels gives nearly the same model

performance and the TCN architecture is already using dilated kernels.

Note: Setting CUDA Deep Neural Network benchmark to True disabled the determinism of the convolutions. We consider this to be a minor issue compared to the performance gain.

3.14 Ablation Study

To take advantage of the sweep's runs, the ablation study was performed on these runs. It has the drawback of not having uniform sampling of the hyperparameters values because of the Bayesian optimization strategy but has the advantage of considerably reducing the number of runs to perform the ablation study.

Sweep data reveals how each parameter affects model performance. By counting how often each value is sampled and computing the mean and standard deviation of metrics (test accuracy, validation loss), the importance and correlation of each parameter can be determined.

3.15 Dependencies and Versions

For the dependent libraries, you should use the same version as the one used in the repository. With only one exception being the libraries that use the GPU: cuML and PyTorch.

For cuML, compatibility could not be achieved on the GPU used due to a CUDA version that was too new for the GPU. To make it work, the output of `nvidia-smi` can be checked to find the CUDA version. Then `cuml-cuXX` and `cudf-cuXX` can be installed (XX being the major version of the CUDA version). The right version of the library that works with the Compute Capability must then be identified. Note: it cannot work with another CUDA version.

3.16 Implemented but Not Evaluated

A few additional ideas were explored but not carried through to final evaluation because they either conflicted with the data regime or were not central to the main contribution:

- **French Sign Language dictionaries**: several online dictionaries were scraped, but the resulting data would have contained a very large number of classes with fewer than five samples per class, making rigorous evaluation difficult in a few-shot setting.
- **Discrete Cosine Transform (DCT) representation**: a frequency-domain encoding of temporal sequences was implemented to compare against frame-window representations. However, the implementation no longer functions correctly in the current codebase and was therefore not included in the final experiments.
- **Classical ML models**: models such as Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Random Forest (RF) were partially integrated into the training code, but GPU support was not available on the target hardware (Compute Capability 6.1), and sweeping their hyperparameters on CPU proved more than two orders of magnitude slower than the PyTorch models.

These attempts nonetheless informed design choices for the main pipeline (e.g. the focus on PyTorch-based models and skeleton window representations) but are reported here only for completeness.

Overall, this chapter has detailed the datasets, skeleton representations, model architectures, hyperparameter search strategy, and implementation choices that define the experimental pipeline for skeleton-based ISLR. Building on these design decisions, the next chapter presents the empirical results obtained with this pipeline and analyzes how different temporal architectures and configurations perform under the constraints introduced here.

4 Results

> I owe my results not solely to the methods I borrowed, but to countless hours spent debugging, adapting, and, above all, listening to what the data tried to tell me.
>
> Original

4.1 Evaluation Protocol

To ensure reproducibility, all parameters are logged in Weights and Biases. It is a framework that enables comprehensive monitoring of training and validation metrics, hyperparameters, and model architecture for every run. It also supports parameter sweeps and run comparisons.

Models are trained with a single highly configurable script that train and then evaluate the model. The script is able to train the model for a maximum of 150 epochs using Cross Entropy loss and AdamW optimizer. There is a Learning Rate Scheduler along with an Early Stopping mechanism to prevent overfitting and stop bad runs earlier. The code of this script changed a bit during the experiments, but remains very similar.

4.2 Evaluation Metrics

As it is a multi-class classification problem and the goal is to recognize signs as accurately as possible, the following metrics are used:

Test accuracy is the number one metric to evaluate the performance of the model. It is also important to ensure that some classes are not mixed up with other classes. So confusion matrix will be used to see mixing patterns between classes.

Other measures are also taken into account, to see if there are some hidden patterns in the data. Examples of such metrics are weighted F1, number of epochs, time per epoch, inference time, GPU memory usage, etc. In the next part, these metrics will be called Secondary Metrics. The ablation statistics underlying the findings below are exported to `data_analysis/models_results/csv/ablation_*.csv` in the repository (one CSV per sweep) with columns: variable, group, metric, count, mean, max, min, best, std.

4.3 Previous Training Results

In the first tests, all models achieved 100% training accuracy. The following table shows test accuracy results:

Model	**Accuracy (PyTorch)**	**Accuracy (DFT)**
SVM (l1)	0.30	0.12
SVM (2-part)	0.61	0.21
KNN (2-part)	0.22	0.11
RF	0.27	0.10

Table 4.1: Test accuracy of selected models from first tests

The 2-part models comprise four separate models, one for each body part (body, left hand, right hand, face). Results are combined using an averaging mechanism.

The multipart models show promise, likely because smaller data partitions reduce Overfitting risk. Therefore, more multi-stage models were explored in subsequent tests.

Note: The results of the first tests are not very reliable as a lot of the configuration change notably the splitting of the dataset which wasn't done correctly.

4.4 Exploratory Experiments

The second iteration of the experiments was conducted using only PyTorch models because training scikit-learn models was taking too much time on CPU and the GPU alternative cuML doesn't work on the hardware used.

The first runs where more something to get a general idea of what parameters work well, how much they influence, ...

The goal was to find the range of values the lr, batch size, max number of epochs, ... can take to get a good result. And of course, get a first impression of the models.

The Linear model was tested first and achieved less than 20% validation accuracy.

Sequential models were then tested. As expected, Recurrent Neural Network (RNN) underperformed compared to LSTM and GRU, which showed comparable performance. LSTM demonstrated a slight advantage, but GRU trained at least 3 times faster. These models achieved approximately 30% accuracy, which remains low.

Next, Transformer and Multi-Head Attention (MHA) models were tested. The Transformer model performed poorly, but the MHA model achieved approximately 50% valid accuracy.

Finally, a Temporal Convolutional Network (TCN) model was tested. The first run achieved 0.80 valid accuracy, and hyperparameter sweeps improved this to 0.82 valid accuracy. Unfortunately, the splitting wasn't exactly right, which is why the next results will have a lower accuracy.

4.5 Model computational cost

All deep learning models use a full GPU PyTorch pipeline. Depending on the model and batch size, they consume between 0.1 and 0.5 GB of GPU memory in bfloat16 precision, which remains very low.

Training time also remains low, varying from 0.2 to 8.2 seconds per epoch on the hardware used, with the following configuration:

Hardware Configuration

CPU: Intel i5-7300HQ (4 cores @ 3.5GHz)
GPU: NVIDIA GTX 1050 (2 GB) with CPU fallback
RAM: 24 GB

1. Lin: 0.2 seconds, 0.1 GB of GPU memory
2. Hierarchical Linear (hLin): 0.3 seconds, 0.1 GB of GPU memory
3. MHA: 0.5 seconds, 0.2 GB of GPU memory
4. Transformer: 1.1 seconds, 0.2 GB of GPU memory
5. GRU: 3 seconds, 0.2 GB of GPU memory
6. TCN: 5 seconds, 0.3 GB of GPU memory
7. LSTM: 7.1 seconds, 0.3 GB of GPU memory
8. Hierarchical Long Short-Term Memory (HLSTM): 8.2 seconds, 0.4 GB of GPU memory

Note: The training time and GPU memory usage are not really reliable as it depends highly on the hidden size and other parameters of the model. But this gives a general impression with a fixed hidden size.

4.6 Data Processing Protocol

Subsequent tests were conducted using only the TCN model because it proved most promising. It achieved the best accuracy, and the Curse of Dimensionality prevented comprehensive testing of all models.

The experiments are separated into three main categories:

Window augmentation compares ways to generate frames when window_size exceeds the number of frames. The implemented methods are linear, bicubic and nearest interpolation along with zero-filled which add frames of zeros at the end of the sequence. (1 sweep)

Window selection compares frame selection strategies. Multiple sweeps are needed due to dependent hyperparameters (Gaussian std, beta). (3 sweeps)

Landmarks augmentation compares frame-level augmentations. For each technique, the optimal intensity and probability that maximize performance are identified; dropout is not included. (4 sweeps)

Each of these sweeps also try multiples values for the window_size, to see if there is an optimal value for the window_size. To find the conditional hyperparameter values the sweep method used is random but for all the other sweeps the method used is Bayesian optimization.

4.7 Data Processing Results

Two sweeps were run to find the best parameters for the beta and Gaussian window selection strategies, followed by identifying the best **window selection** strategy with the fixed sub-hyperparameters.

The parallel coordinate plots in Figures 4.1–4.4 summarize sweeps performed with Weights and Biases. Each polyline corresponds to a single training run. The vertical axes represent hyperparameters or measured quantities (e.g., window size, seed, loss). Following a line from left to right shows the configuration of that run and its resulting metrics; lines are colored by performance according to the figure legend, so clusters of darker (better) lines highlight promising regions of the hyperparameter space.

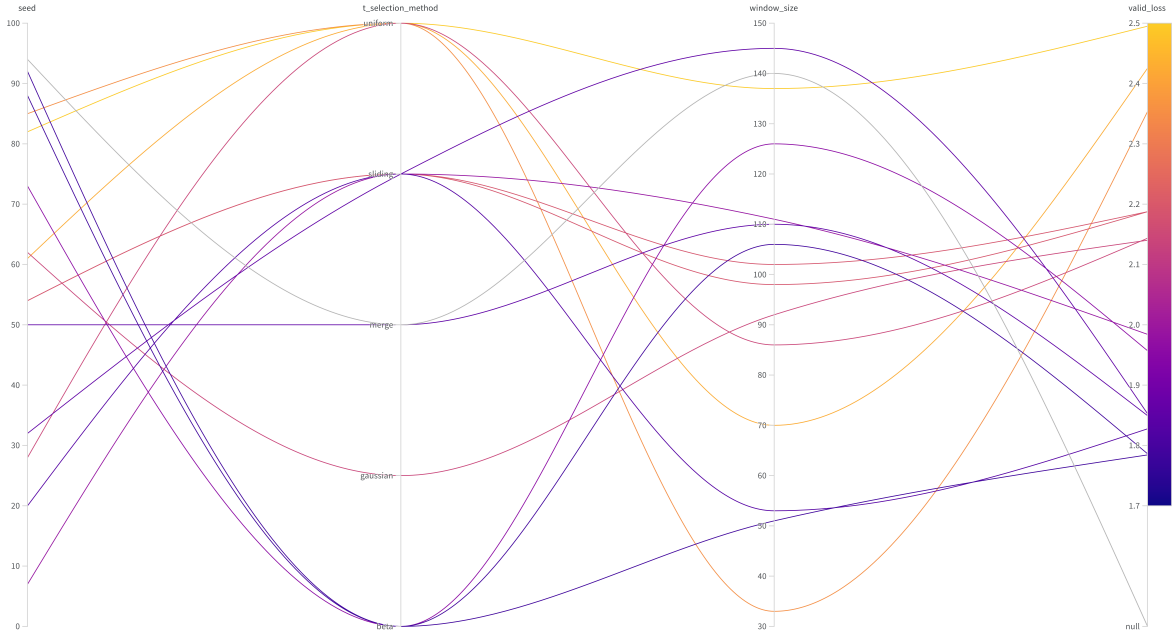

Figure 4.1: W&B parallel coordinate plots of window selection strategies sweep

Figure 4.1 suggests that uniform selection is a weak default: its runs concentrate among the lower-accuracy lines across the metric axes. The evidence for Gaussian and merge selection is limited in this sweep (few successful runs), so their ranking is less reliable. However, the available runs do not indicate a consistent advantage over the strongest-performing methods.

Window augmentation techniques:

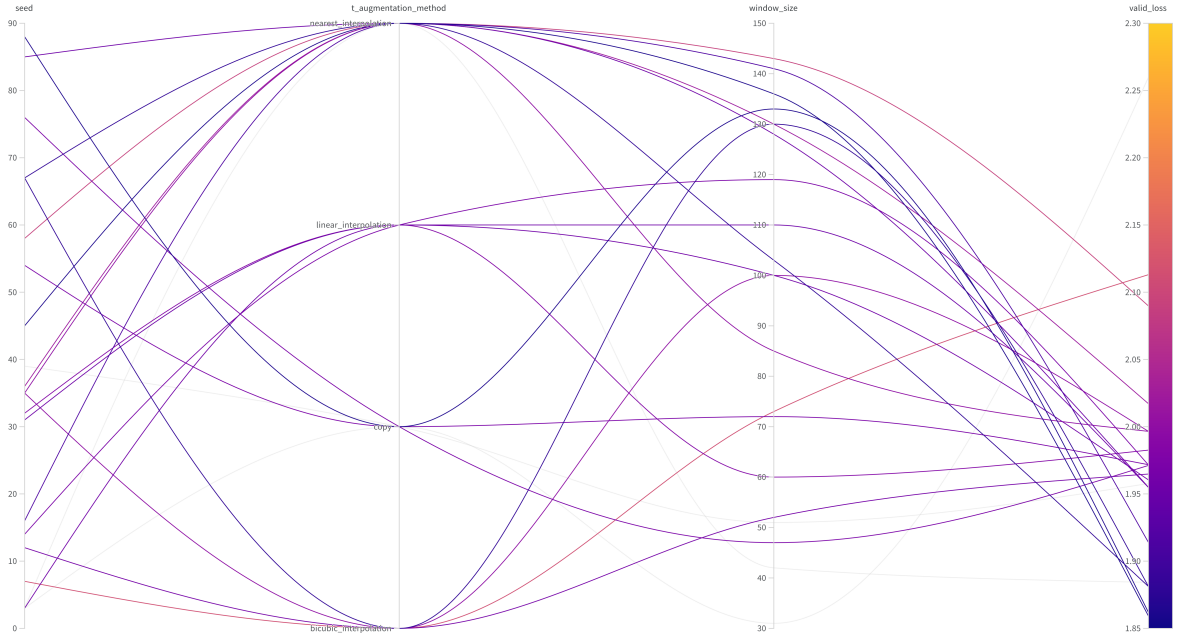

Figure 4.2: W&B parallel coordinate plots of augmentation strategies sweep

In Figure 4.2, the window_size axis aligns strongly with the highest-performing lines, indicating that window length is a primary driver of accuracy in this sweep. Consistently, the ablation statistics show a rise in mean test accuracy from 59.5% (window size 31–52) to 64.2% (130–143). By comparison, the augmentation method has a weaker visual separation, and seed-to-seed variability is of similar magnitude. Bicubic interpolation is associated with higher mean validation loss (1.99) than nearest (1.95) and linear (1.97), and is therefore deprioritized. Nearest interpolation is largely redundant with linear, so subsequent experiments focus on linear and copy augmentations.

Window selection and augmentation combination:

These plots showed that higher window size is promising. An additional sweep was performed that looks at the best window selection strategy and the best augmentation strategy at the same time.

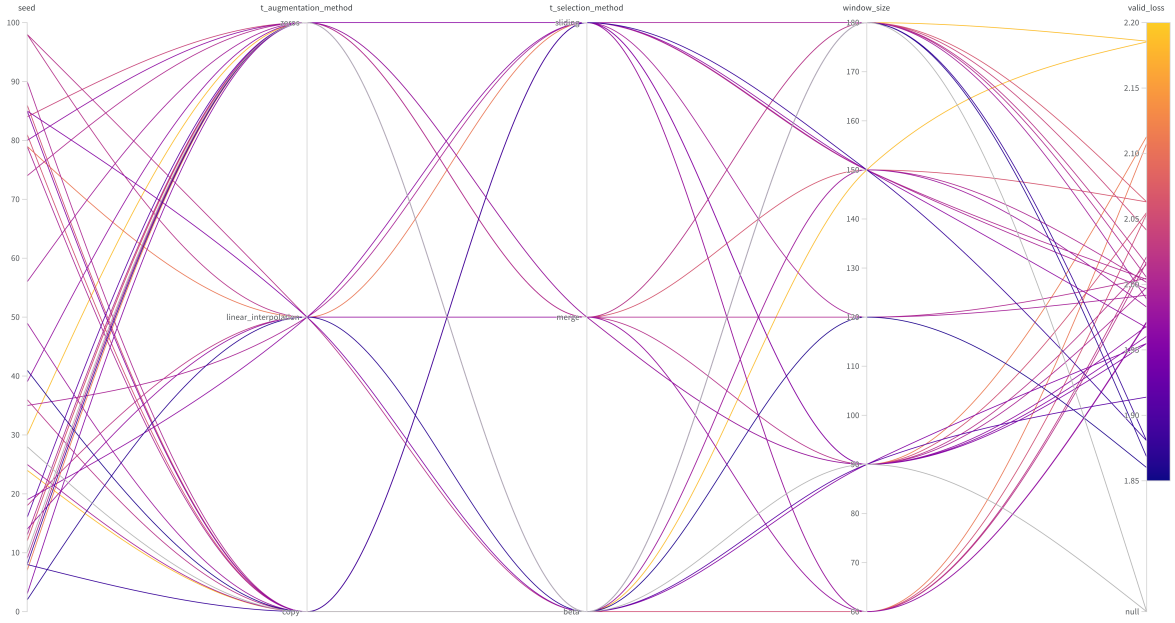

Figure 4.3: W&B parallel coordinate plots of window methods sweep

Figure 4.3 jointly varies selection and augmentation. In this sweep, mean test accuracy by window size is 62.7% (60), 62.8% (90), 60.9% (120), 62.7% (150), and 61.2% (180), which weakens the monotonic trend seen earlier and suggests that mid-range values can be competitive. Across the plot, many high-performing lines traverse multiple augmentation and selection choices, indicating that these choices are second-order compared with window size and run-to-run variability. Beta selection and copy augmentation have the highest mean accuracy (62.4% and 62.4%, respectively), but the interaction between them and window size is not consistent across bins (e.g., beta+copy is best at 120 but worst at 150), so the combination should be interpreted cautiously.

Now it's time to find the best **Landmarks augmentation** techniques.

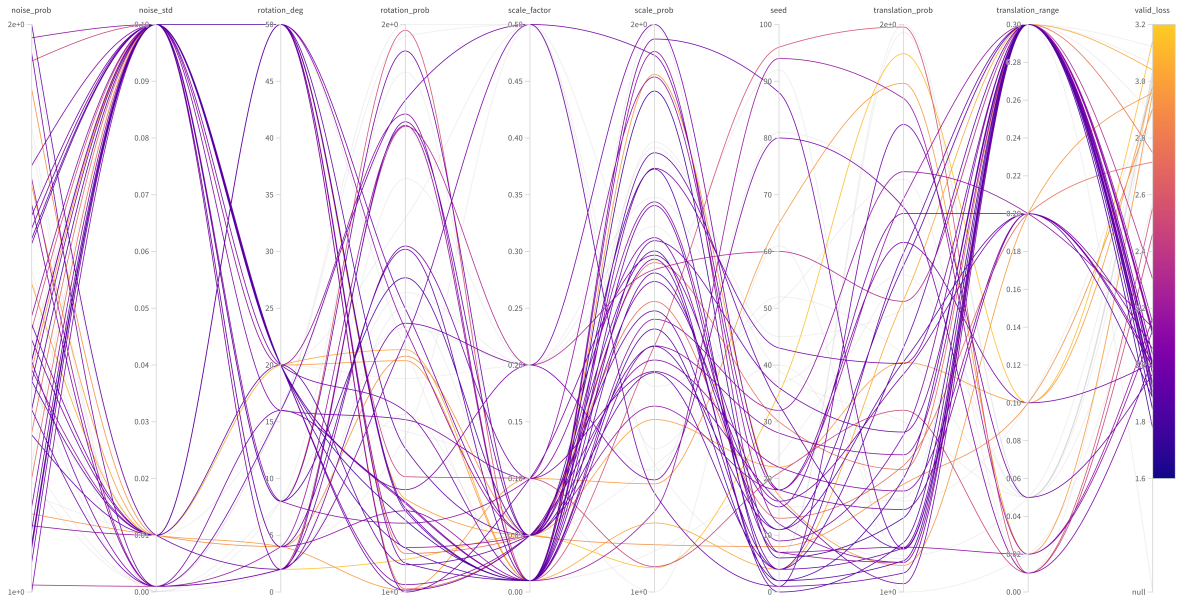

Figure 4.4: W&B parallel coordinate plots of Landmarks augmentation strategies sweep

In Figure 4.4, the performance bands overlap substantially across augmentation settings, consistent with the small variation in mean test accuracy (61.7–63.8%). Nevertheless, the plot and the ablation statistics provide weak trends that are useful for narrowing the search: stronger translations (e.g., translation range 0.3 with mean 63.5%) combined with lower translation probability (lowest bin 1.01–1.048 with mean 63.7%) appear favorable, and lower noise probability is preferable (noise_prob 1.404–1.625 has mean 61.7% vs 64.4% for 1.11–1.209). Beyond these broad tendencies, the plot does not show a clearly separable optimum, suggesting that landmark-level augmentation is beneficial but not the dominant factor in this setting.

4.8 Model Protocol

The first goal is to find by experiment the best hyperparameters values of the implemented models. One sweep per model is used to find the values of all the hyperparameters of the model, both the ones that are common to all the models and the ones that are specific to the model.

The second goal is to compare the models to each other, using the Secondary Metrics to see how they differ from each other. An extra script is used to read the sweep Secondary Metrics from W&B and compute the statistics.

Lin: Sweep over Landmarks augmentation config, hidden size, dropout, Learning Rate, and grad accumulation steps.

GRU: Chosen among sequence models because RNN performed poorly and GRU matches LSTM accuracy (within 1%) with much lower training time and GPU memory. Sweep over Landmarks augmentation config, hidden size, number of layers, dropout, Learning Rate, and grad accumulation steps.

MHA: Chosen as the foundation of the Transformer. Sweep over hidden size, embedding size, number of heads, dropout, and Learning Rate.

TCN: Sweep over hidden size, number of levels, kernel size, and Learning Rate.

All sweeps used random seeds, which affect weight initialization and batch shuffling.

Note: To find the possibles values of the variables or other information about the sweeps, please refer to the repository A.1.

4.9 Individual Model Results

Lin model:

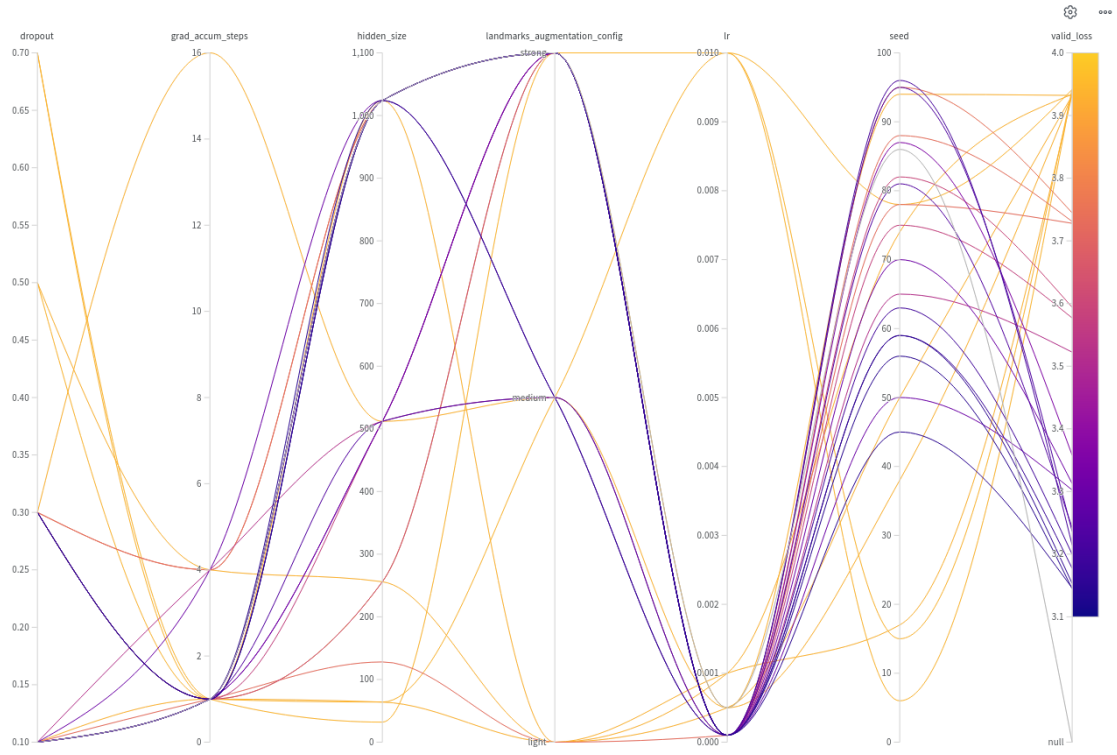

Figure 4.5: W&B parallel coordinate plot of the Lin model sweep results

Figure 4.5 shows that the linear baseline is highly sensitive to a few hyperparameters while remaining limited in absolute performance. The strongest separation appears on the dropout and learning-rate axes: mean test accuracy is 13.0% for dropout 0.1 and 12.6% for 0.3, but collapses to ≈2% for 0.5 and 0.7. For Learning Rate, 10^{-4} yields mean 16.2% compared to 3.1% for 5×10^{-4} and 2.5% for 10^{-3}. Hidden size 512 and 1024 have the best means (12.3% and 14.9%), while large gradient-accumulation values are associated with poorer runs (11.4% for 1 vs 9.2% for 4 and 4.0% for 16). The seed shows little structure in the plot, suggesting that performance is primarily driven by the optimization-related settings.

Another use of this sweep was to find the best batch size in terms of training time and GPU memory usage. The goal is to reduce the training time as much as possible and often the right batch size is the biggest power of 2 that fits in the GPU memory.

GRU model:

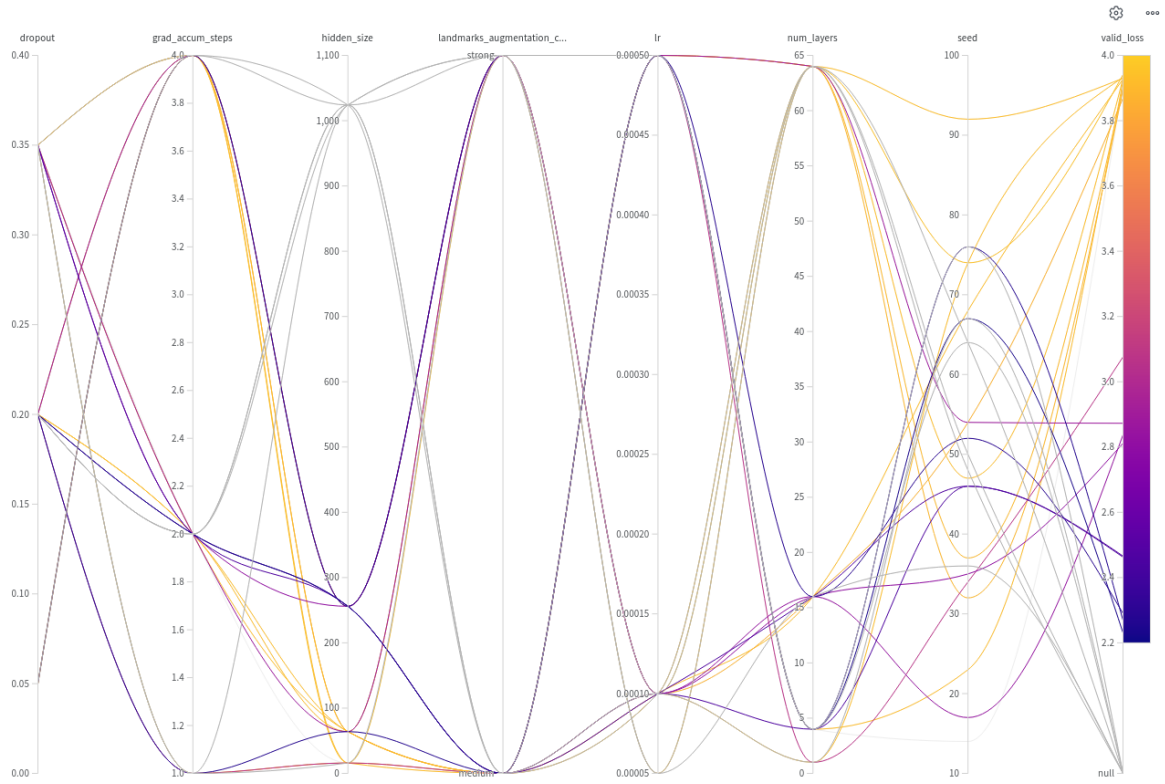

Figure 4.6: W&B parallel coordinate plot of the GRU model sweep results

Figure 4.6 indicates that the largest hidden size (1024) is not feasible under the available GPU memory constraints. Among feasible settings, hidden size 256 yields the best mean test accuracy (39.9%) in the sampled range. Gradient accumulation 1 and 2 both appear among the best-performing lines. Subsequent experiments will use 2.

Landmarks augmentation performs best at strong settings (mean 25.4%) but remains effective at medium (31.8%).

In the sweep, Learning Rate 5×10^{-4} has higher mean test accuracy (39.7%) than 10^{-4} (21.6%), though individual runs with 10^{-4} can reach similar peaks.

Another use of this sweep was to find the best effective batch size in terms of model performance. After finding the best batch size in terms of training time in the previous sweep, now the goal is to find the best effective batch size that train faster and with the best final loss.

MHA model:

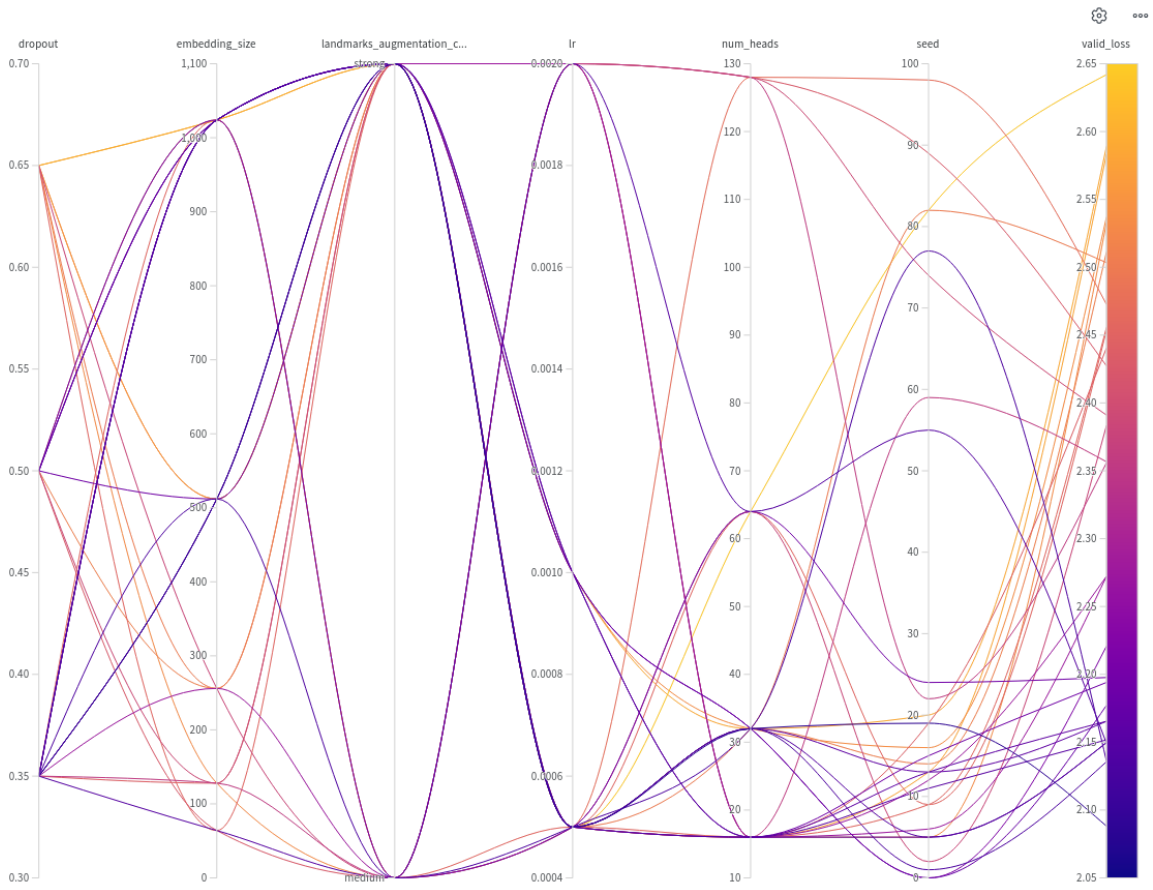

Figure 4.7: W&B parallel coordinate plot of the MHA model sweep results

For this sweep, the results where simple: bigger is better. Performance improves with model capacity within the explored range. Embedding size and hidden size show the clearest association with lower validation loss: the best runs concentrate at embedding size 1024 and hidden size 256. Intermediate values (e.g., hidden size 64 or embedding size 512) are competitive mainly when paired with the largest setting on the other axis, indicating that under-sizing either dimension can become a bottleneck.

Number of heads should be ≥ 16 but after that all values are around the same importance. Dropout value have a negligible importance (even smaller than seed).

TCN model:

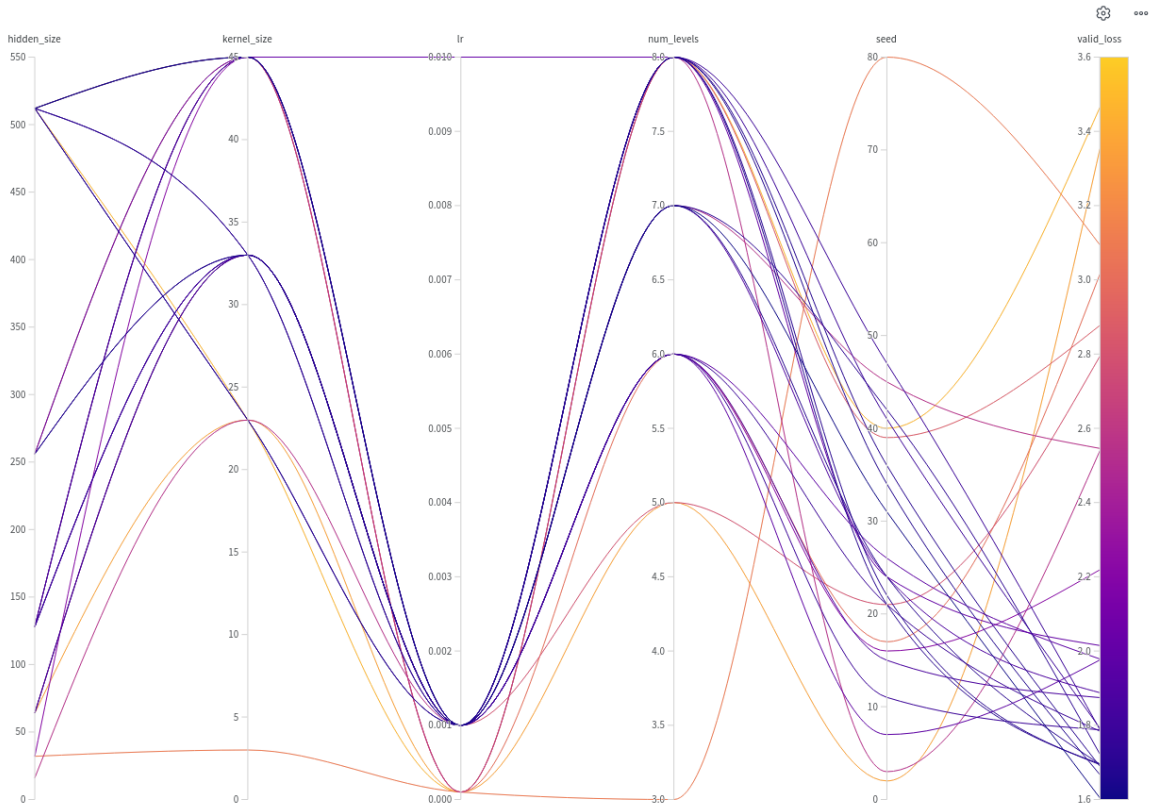

Figure 4.8: W&B parallel coordinate plot of the TCN model sweep results

Figure 4.8 suggests that the number of temporal levels should be at least 7 in the explored configurations. Beyond that, the differences between levels are comparatively small. This is consistent with the TCN's dilation values, each level take $\frac{1}{2^k}$ of the frames with k the number of the level. In this case, the last level takes $\frac{1}{2^7} = \frac{1}{128}$ of the frames.

The best Learning Rate is around 1e-4. This parameter is one of the most important. Kernel size seems better when set to high values, so 33 is used for the next experiments. Hidden size is one of the least important parameters, any value greater than 64 looks like a good choice; 128 is taken for the next experiments as it considerably reduce the training time.

4.10 Between Models Comparison

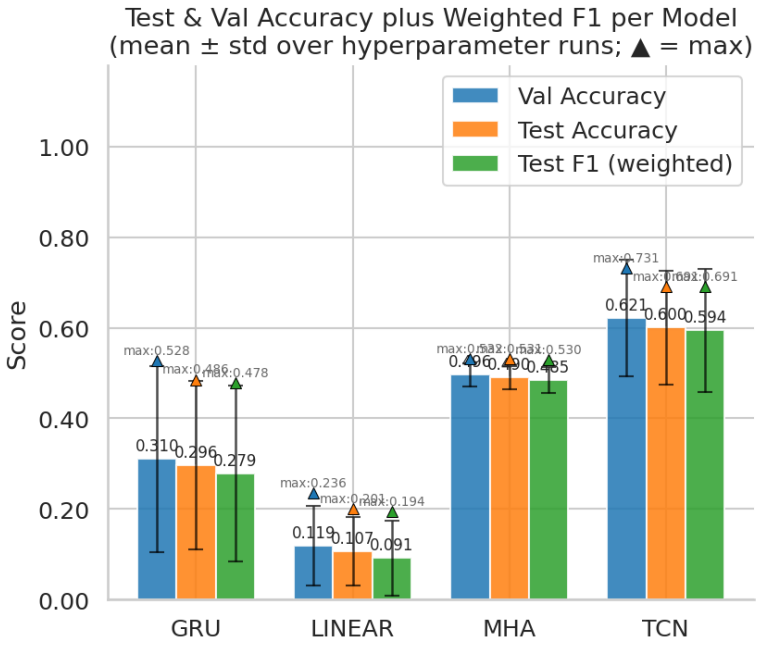

Figure 4.9: Test accuracy and weighted F1 score comparison between models

Accuracy and weighted F1 are closely aligned for each model; changes in one affect the other proportionally.

In terms of test accuracy or test weighted F1 score, the best results are achieved with the TCN models, then the MHA models, then the GRU models, then the Lin models are the worst.

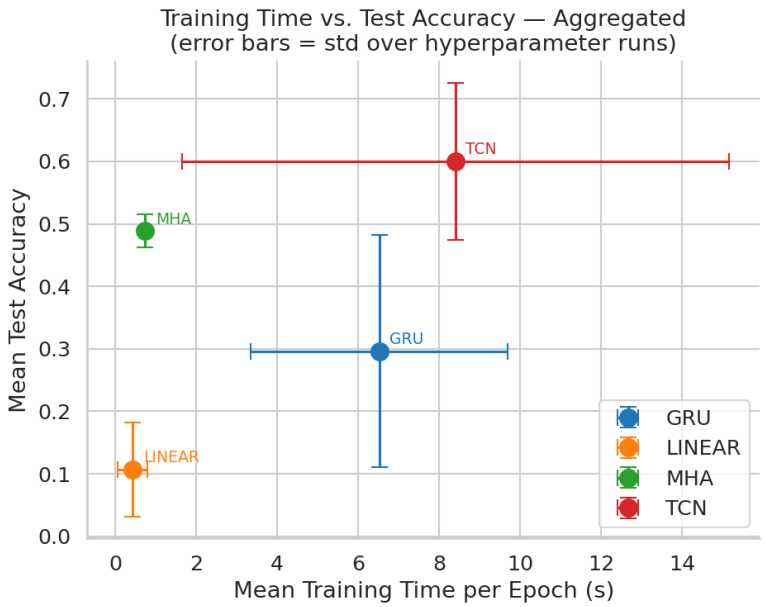

Figure 4.10: Test accuracy vs training time with standard deviation

The standard deviation is high for some models and low for others, the reason is that the sweep values range can be higher or have a greater influence on the training time or test accuracy.

Figure 4.11: Test accuracy vs training time scatter plot of all runs

The training time and test accuracy seems linearly correlated for linear models. But for GRU there are no apparent correlation. MHA have a pretty constant training time regardless of the test accuracy. And TCN is the opposite, its test accuracy doesn't seem to be related to the training time. The TCN training time is higher because the convolution cost dominate and is not really optimized on old GPUs and the training time is proportional to the hidden size which doesn't improve much the accuracy.

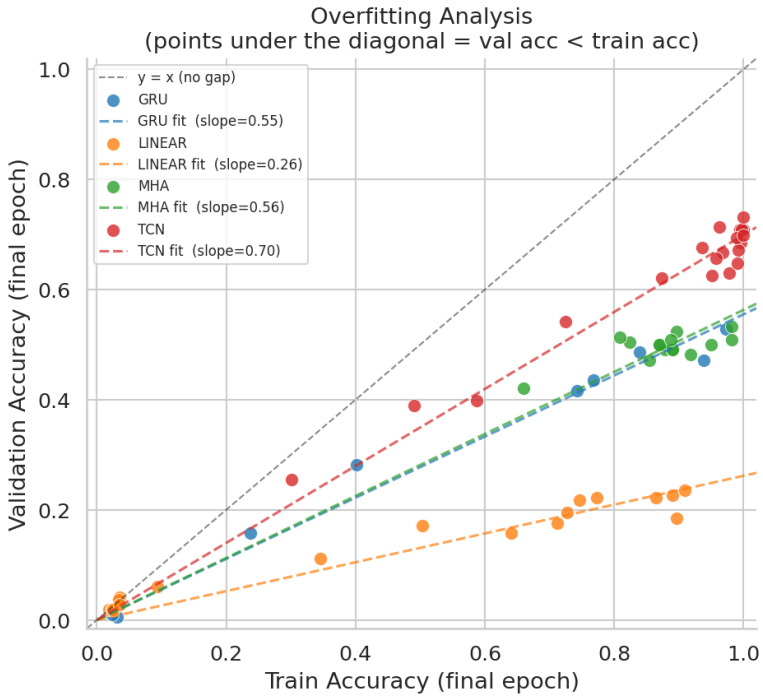

Figure 4.12: Validation vs training accuracy scatter plot between each run

An interesting observation is that each model shows its own slope as if overfitting was an intrinsic characteristic of the model and not something linked with the training time, dropout, Learning Rate or other features.

Validation accuracy increases proportionally to the training accuracy. A linear regression shows that the slope is around 0.7 for the best model (TCN) which is a bit low.

Summary of the model results:

Model	Accuracy Training Time (s)	Top-5 Accuracy Inference Time (s)	F1 Score GPU Memory (GB)
Lin	0.20577 121.41381	0.49059 **0.32989**	0.20499 1.426
GRU	0.51819 238.92424	0.77792 0.6599	0.51164 0.413
MHA	0.48306 **110.5351**	0.79423 0.36708	0.47608 **0.409**
TCN	**0.60351** 505.07749	**0.8256** 2.46449	**0.5976** 0.642

Table 4.2: Results of the best run for each model

Note: To reproduce these runs, the config files have been saved in the repository along with a script that allows to reproduce a run with the exact same config.

The linear model that performs the best is a model with a bunch of parameters. But while being the biggest model by far it is the fastest in inference. MHA for its given size is pretty fast too because its architecture is very similar to a linear model. And GRU is two times slower and TCN is even slower because of the convolution on an old GPU.

4.11 WASL protocol

Another experiment regroup multiples goals:

- Ensure no mixing patterns between classes.
- Compare the best model with the State-of-the-art models on WASL-100 instead of the hand made citizen-50.
- See the importance of the seed on the performance of the model.
- Evaluate on both signer dependent and signer independent splitting of the dataset.

Two random sweeps will be necessary (one for the signer dependent and one for the signer independent). The confusion matrix will be logged in a log file for a deep analysis. Those will be the only sweeps that have different learning rate scheduler and Early Stopping values. The only parameter that will change across runs is the seed. An extra code is used to read the sweep confusion matrix raw data from W&B and compute the stats.

The number of data element per class is small, particularly when reduced for the test set. That's why the confusion matrix is not very reliable on one run. But it can be reliable enough if averaged across runs. Other interesting metrics will be computed such as sum along the columns or sum along the rows (except the diagonal).

4.12 Between Class Analysis Results

The first sweep was on citizen_50 2d signer independent. The seed had a negligible importance on the performance despite having an impact on the batch shuffling, transformations and weights initialization.

The loss varied between 2.02 and 1.70.

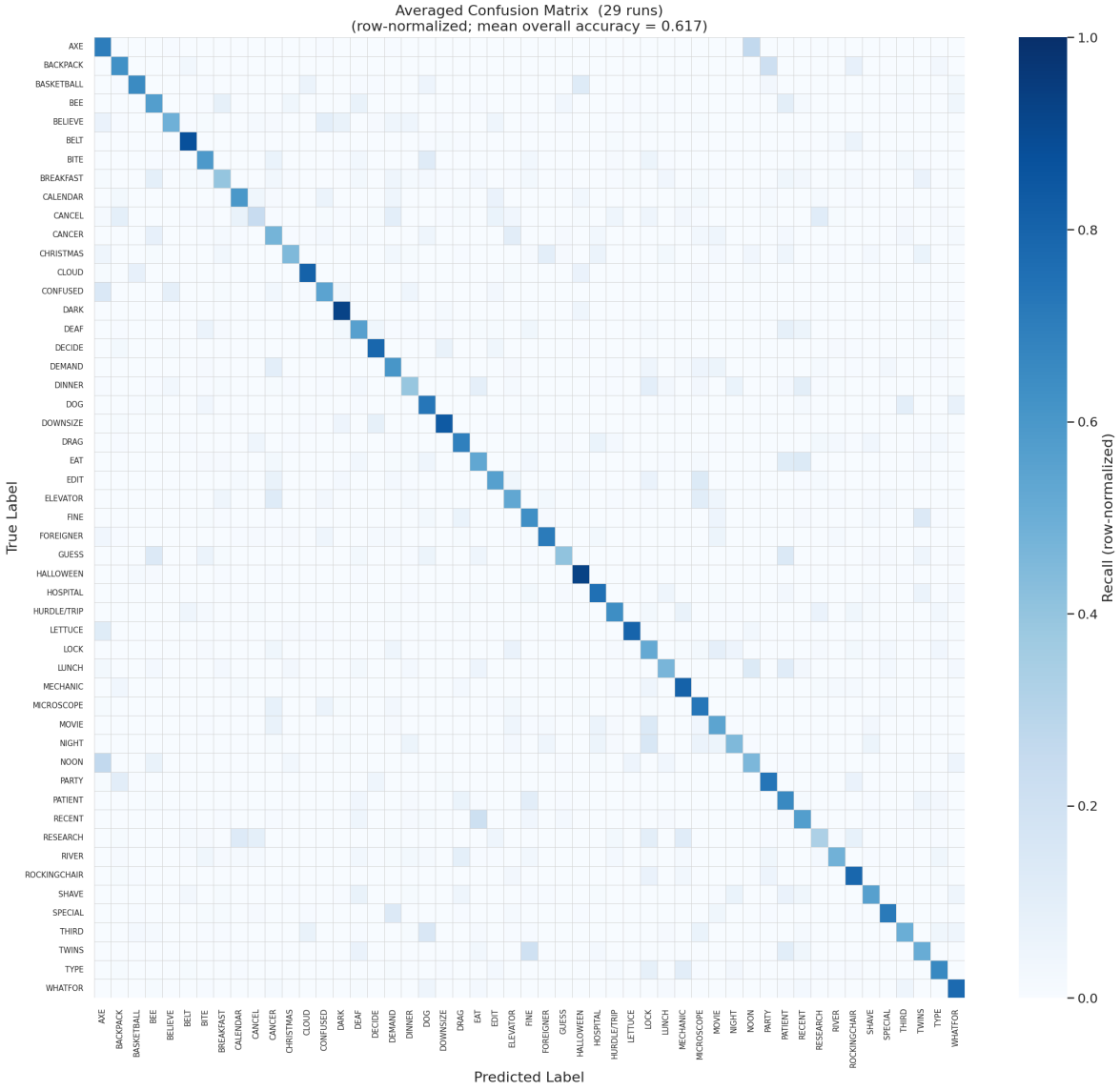

Figure 4.13: Normalized confusion matrix of all the runs

No class is systematically confused with another. The averaged recall below shows that some classes are harder to recognize than others. The confusion matrix is not really symmetric, it seems that some classes can be mixed with others while the opposite is not true. The only exception are the classes NOON and AXE.

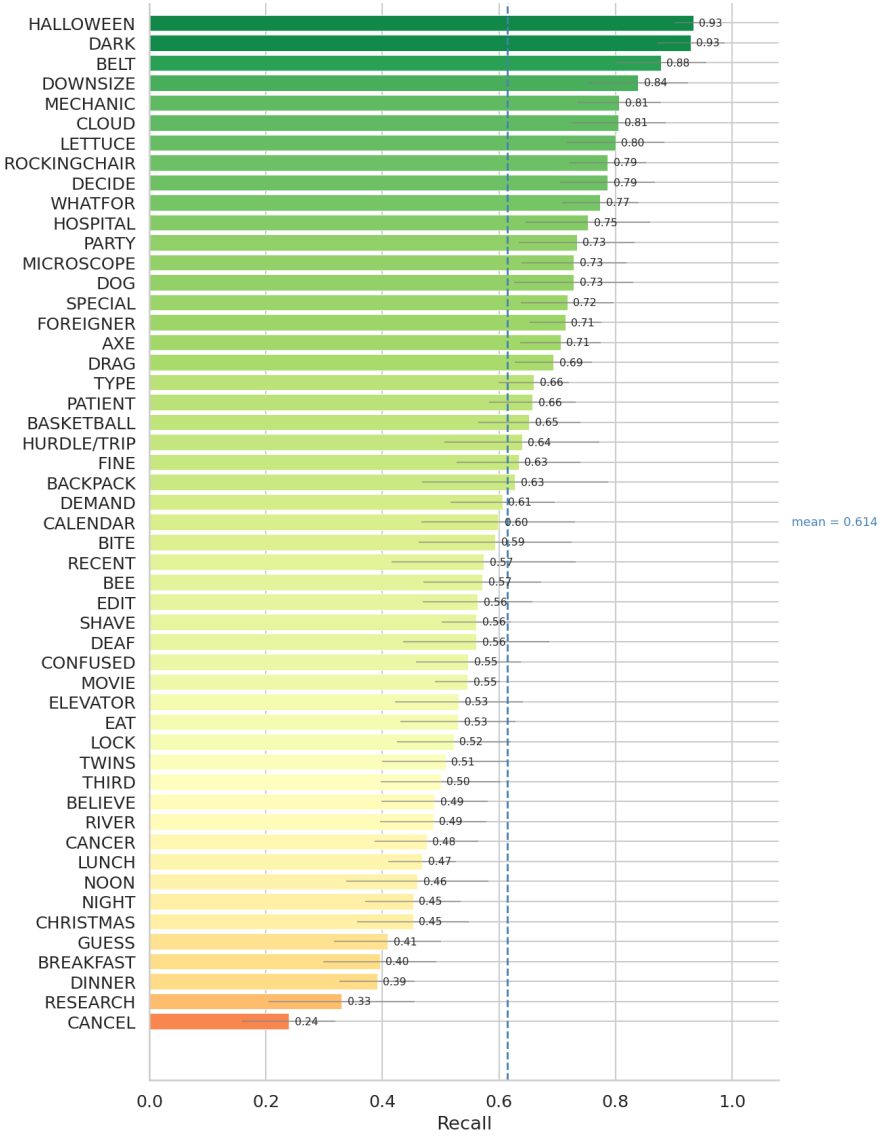

Figure 4.14: Per-class averaged recall over 29 runs

4.13 WASL Datasets Results

A sweep was run to find the best parameters for other datasets, measuring performance on WASL-100 and also to assess the impact of: number of classes, 2D vs 3D coordinates, and signer-dependent vs signer-independent splits.

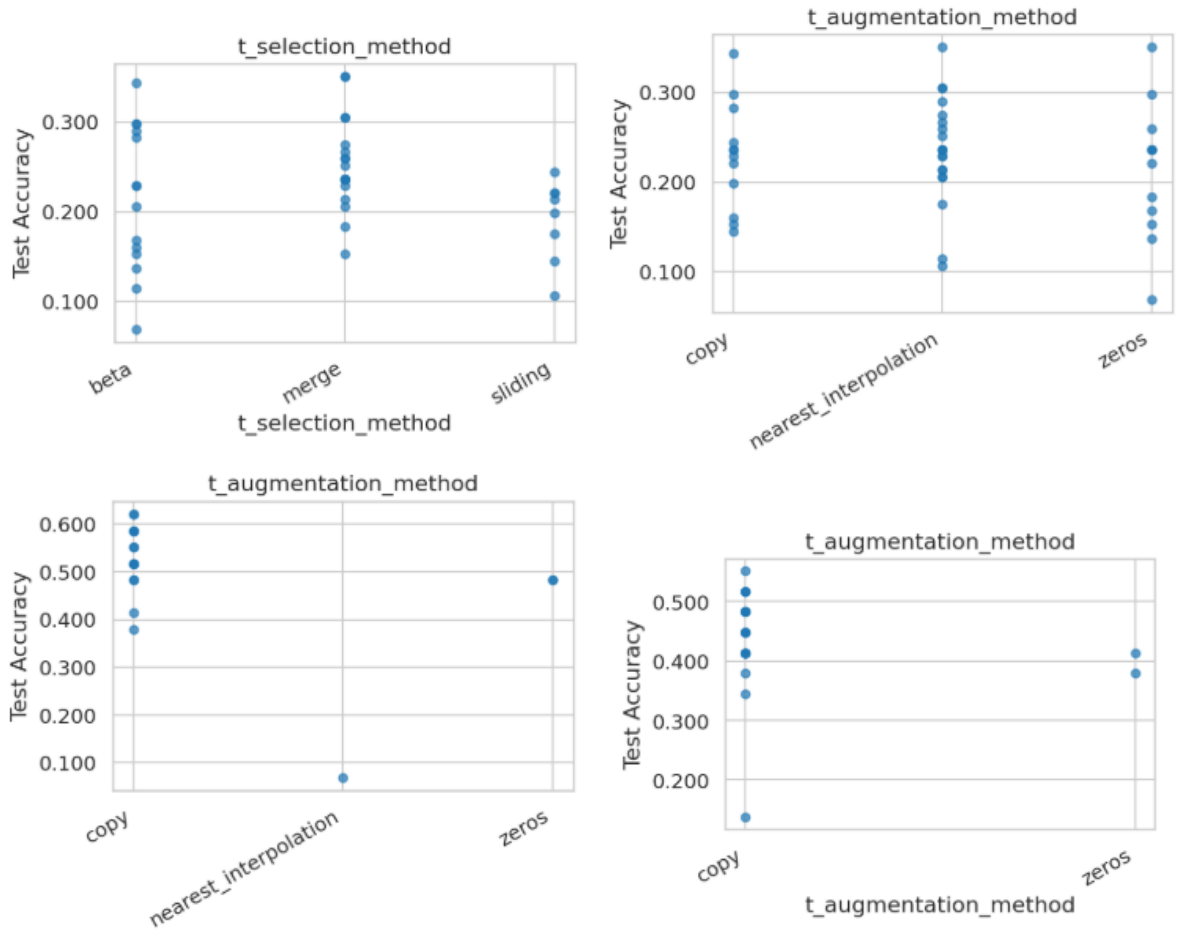

Figure 4.15: Scatter plot of ablation of some parameters of the WASL datasets

Here are the best models stats for: WASL-100 2d signer independent, WASL-100 2d signer dependent, WASL-20 2d signer independent and WASL-20 3d signer independent.

Model	*Accuracy*	*Top-5*	*Top-10*	*F1 Weighted*	*Training (s)*	*Inference (s)*	*GPU (GB)*
WASL-100 2d indep	0.206	0.519	0.649	0.164	3175.534	24.992	0.317
WASL-100 2d dep	0.365	0.625	0.750	0.296	**100.810**	**0.452**	**0.187**
WASL-20 2d indep	**0.552**	**0.862**	0.862	**0.531**	691.027	23.869	1.564
WASL-20 3d indep	0.448	0.827	**0.966**	0.379	631.296	23.807	1.548

Table 4.3: Results of the best run for each WASL dataset version

Sometimes, the val loss stagnates while the train loss is still decreasing and the val accuracy is still increasing. Unfortunately, Early Stopping (after 18 epochs) stopped the experiments before seeing if the accuracy would continue to grow. The fact that the val accuracy and loss are irregularly increasing and decreasing and the val loss and accuracy ceased to be correlated led to changing the Learning Rate Scheduler and Early Stopping conditions.

As expected, 100 classes are harder than 20, suggesting that more classes increase difficulty even though the model sees fewer data samples during its training. Signer-independent splits are also harder than signer-dependent, though the difference is smaller than expected.

Unexpectedly, the 3d version is more difficult to recognize than the 2d version. It is a bit surprising because it's just adding some data; intuitively one might think that even adding random noise wouldn't decrease the performance. But that's only true for train loss, the problem is that more data increase the tendency to overfit.

4.14 Results Summary

Taken together, the experiments reported in this chapter show that temporal convolutional networks strike the best balance between accuracy and resource usage on skeleton-based ISLR, while attention-based architectures offer competitive performance under tighter computational budgets and linear models remain useful as fast but weak baselines. The comparative analyses across datasets, signer protocols, and skeleton variants highlight both the strengths and limitations of the proposed pipeline, especially its sensitivity to domain shifts and evaluation protocols. The next chapter interprets these findings in more depth, discusses their implications for ISLR and assistive technologies, and examines the main limitations of the present study.

5 Discussion

> The important thing is not to stop questioning.
> Curiosity has its own reason for existing.
>
> Albert Einstein

This chapter presents a comprehensive analysis of the model results. The main difficulty was to cover a wide range of experiments while running them enough times to get a good estimate of the performance.

5.1 Data

As previously mentioned, the primary challenge in this thesis stems from the limited number of data samples per class.

The second challenge arises from the data's dual nature as both time series and multi-dimensional skeleton data.

For most of the experiments, the citizen-50 2d signer independent split was used. It is a deliberate choice made because the dataset is more homogeneous and the given split is easier to use, more reliable and closer to the real life cases. Also, citizen-50 while being less known, is not entirely made by professionals contrary to WASL which bring us closer to the real life cases.

5.2 Data Loader Analysis

The first results show that the DCT loader underperforms compared to the default loader. This may occur because signs encode information through both movement and position, not movement alone. Additionally, only the first 10 coefficients are retained, which may filter out important high-frequency information.

5.3 Data Processing Analysis

Initially, window augmentation and window selection were explored separately. Since they may be correlated and combining them would not require many additional runs, a single sweep was performed on both variables at the same time.

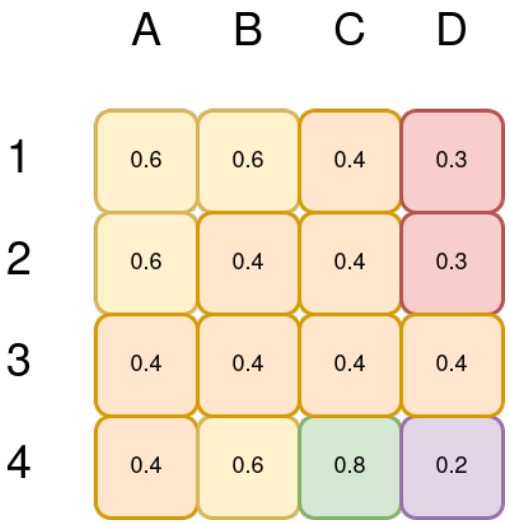

Figure 5.1: Example to illustrate a correlation between two variables

In this example, the highest accuracy depend at the same time on letters and numbers. If we set one to a fixed value, it will change the best value of the other.

Possible interdependence between the variables justify the decision to perform a joint sweep over window augmentation, selection strategies, and window size simultaneously.

For the Landmark augmentation techniques, the hypothesis was made that they are not too much correlated to the window augmentation and selection strategies.

The best data parameters were not sought at the same time as the model parameters. It also would have been interesting to compare the impact of the model parameters vs the data parameters.

The sweeps suggest that, within the tested ranges, the random seed contributes variance comparable to, and sometimes larger than, the choice of window selection or augmentation methods. But some parameters of the landmark augmentation have a higher impact like translation range, scale probability, translation probability and noise probability.

5.4 Model Analysis

We saw that on all metrics (validation accuracy, validation loss, test accuracy, top-5 accuracy, weighted F1 score), the best model is the TCN model. Then MHA and GRU are similar with a slight advantage for MHA. And Lin is the fastest model due to its simplicity.

Figure 4.12 shows that when plotting validation accuracy vs training accuracy, each model can be modeled by a linear regression. A slope close to 1 indicates nearly no difference between validation and training accuracy. A slope close to 0 indicates that the model is not learning and performs only on the training data. One might expect all models to follow the same polynomial curve with the best runs in the upper right. But overfitting appears to be an intrinsic characteristic of each model rather than something linked to other parameters.

We said in background chapter that ISLR is a few-shot problem with high-dimensional data. TCN which uses convolutions, make a better use of its parameters and lead to less overfitting as the experiment confirm. It has a slope of 0.7 which is a moderate level of overfitting. Furthermore, having multiples levels allows to capture the sign at different time scales which fit both positional and very dynamic signs.

Next are the MHA and GRU models which have a slope of around 0.5 which starts to be a serious amount of overfitting. As they are both recurrent models, it seems that they make a less good use of their parameters than causal convolutions.

Attention is mid-level in terms of performance because it allows focusing on multiples parts of the sign at the same time but lacks fitting for the temporal nature of the sign.

Contrary to GRU which completely leverages the temporal nature of the sign, but lacks explicit mechanisms for multiscale spatial context compared to TCN.

Due to the combinatorial explosion, it is not possible to combine data and model hyperparameters. The assumption had to be made that they are not too much correlated.

5.5 Class Recall Variation Analysis

AXE vs NOON:

Figure 4.13 4.13 shows that AXE and NOON are more frequently confused with each other than with other classes.

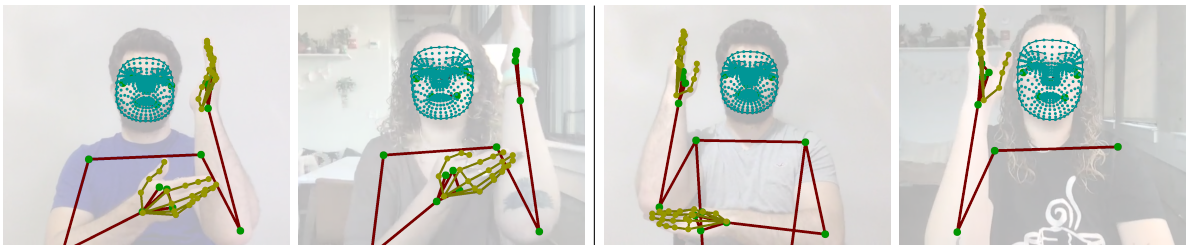

Figure 5.2: Skeletons of AXE (left) vs. NOON (right)

Inspection of the original videos suggests that the two signs share a very similar global configuration: one arm is vertical while the other is horizontal and touches the vertical arm twice. The key difference is in the height of the horizontal arm: for NOON the contact occurs below the elbow, whereas for AXE it occurs in front of it.

Recall differences interpretation:

Figure 4.14 4.14 shows substantial variability in recall across classes.

When examining recall per class, it is natural to ask whether some classes are systematically over-predicted by the model. However, the average predicted count per class can see a bit of imbalance but not for the expected classes. The most represented classes are PATIENT, LOCK then AXE with 24 to 22 predicted counts and the least represented are RIVER, CANCEL and GUESS with 10 to 8 predicted counts. The ordered distribution look around the same as the recall per class, but the classes on top are not the same.

The number of predicted counts per class does not fully explain the recall differences. To provide more intuition, several representative classes are examined below.

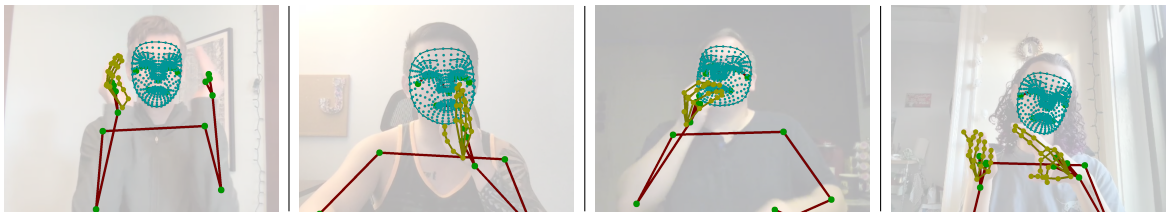

Figure 5.3: Skeleton of HALLOWEEN, BREAKFAST, DINNER and RESEARCH

For the sign HALLOWEEN, the gesture is almost always executed as a short wrist movement with very similar trajectories and body configuration. This low intra-class variability makes the class comparatively easy: the model can rely on a stable motion pattern around the wrists and elbows, and temporal misalignment between executions is small. In other words, the class is well clustered in the feature space, which explains its relatively high recall.

In contrast, BREAKFAST shows much higher intra-class variability. The global hand trajectory is similar, but the exact index-finger position, handedness (right vs. left hand). On top of that, this sign is in two parts, which requires more frames and more information to be recognized.

For DINNER, most executions share the same final pose, but the initial movements differ: some signers start with an additional small hand movement while others transition directly to the main gesture.

Finally, RESEARCH is defined by the hand shape in form of the sign "R" that slides along the palm of the other hand. While this relational cue (*one hand moving on the other*) is stable, the absolute arm configuration varies a lot: elbows can be high or low, close to or far from the torso, and sometimes part of the palm is occluded.

CANCEL class:

It remains unclear what makes the recall difficult for some classes and not for others. To make this more concrete, we focus on one illustrative example: the class CANCEL.

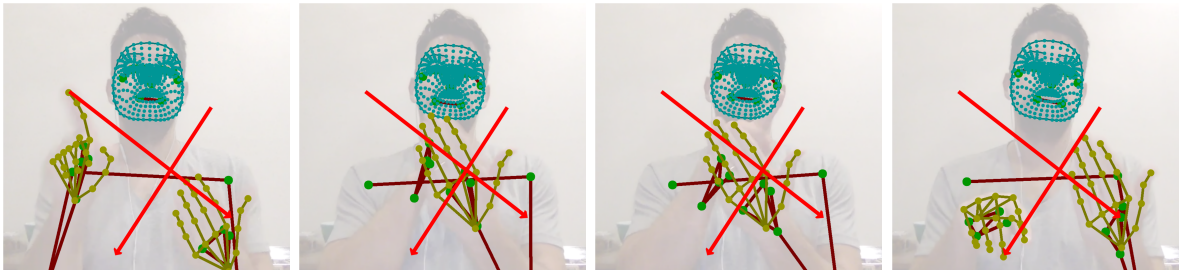

Figure 5.4: CANCEL arrow sign

First, the word has multiple valid execution forms: signing a cross starting with either the upper right diagonal or the upper left diagonal, as well as using either the right or the left hand to draw the arrow.

Second, the sign is highly dynamic, and the model may struggle to capture the landmarks when the signer moves rapidly over a short period of time.

Third, it is frequent that when behind something else, the landmarks are not detected correctly.

Finally, the chosen data or model hyperparameters (such as the window size) might be effective for most signs but suboptimal for this specific one.

The results on the WASL dataset are far lower than with the ASL-Citizen dataset. It might be due to the fact that the tests and the choices of hyperparameters where done on the ASL-Citizen dataset, but it doesn't fully explain this big gap.

5.6 Frugality and Efficiency

The code use Skorch Hugging Face integration *AccelerateMixin* to handle different precision. Along the runs, no big difference in performance was observed between float16 and bfloat16 and float32. For the training time, using another GPU would have change something, but on this GPU, only the memory usage changed.

The training fits an 8+ years old 2 GB VRAM GPU and still manage to have an acceptable training time. Inference on the full test set runs on CPU in less than a second for most models (around two seconds for convolutional models), showing that the final pipeline remains usable on modest hardware.

5.7 Sweep Analysis

The method of doing a sweep and then computing the ablation statistics is really effective given the combinatorial explosion of the number of experiments and the fact that there are already 800+ runs in this thesis. If we say that each run takes around 8 seconds per epoch $\times$ 80 epochs = 640s = 10.6 minutes. Then, the total time to run all the experiments is around 800 $\times$ 10.6 minutes $\approx$ 5.8 days. However, there are some flaws with this method:

- To find the best hyperparameters faster, the sweep had a Bayesian optimization strategy which loses the uniform sampling of the hyperparameters values and makes some ablations not statistically significant.
- The number of runs, while being high enough to get a good estimate of the performance, can be too low to get a statistically significant result.

5.8 Limitations

Methodological constraints

The Learning Rate Scheduler, Early Stopping, Early Terminate, precision, optimizer, label smoothing, etc. were chosen not because of a sweep to find out the best ones but through previous experimentation and intuition which doesn't guarantee the best ones.

Varying the signer-independent split would have been valuable. For signer-dependent splits, the method is random and seed-dependent. For signer-independent splits, stratification and one set per signer are required, which is more complex to implement, so only the provided split was used.

A constraint on window size for more efficient real-world usage would also have been useful.

Including a t-SNE visualization of the classes would have been valuable to see how separated the clusters are.

No measures of how correlated the methods are. For example, how much the choice of the window size influence the choice of the window augmentation and selection strategies.

Non-constant metrics: some papers use top-k accuracy and some others use recall@k. In this thesis, accuracy is used as a primary metric because it more directly reflects how we would evaluate the model in real life.

There are no domain-expert validation with deaf community members to criticize the results and provide feedback. Such user- and expert-centered evaluation was explicitly considered out of scope for this methods-focused work and is left to future applied studies.

Hardware constraints

Some models were not tested, notably scikit-learn models that did not run on the available GPU. Speaking of model creation, merging models to combine their strengths or designing a new model tailored to the task would have been worthwhile. The combining mechanisms for changing behavior for inference were implemented but not tested.

To increase speed of implementation, the code was developed for a specific use case first and not fully generalized. For example, to use W&B parameters, there are no variables, you will need to change the entity and project names in the code. Or the versions of some libraries or use of CUDA Deep Neural Network are chosen for the specific hardware used and might need to be tweaked for better performance on different configurations.

Data & Augmentation constraints

Some augmentation techniques were not tested, including SKiM [Lin+24] which merge frames across samples (which greatly increases data quantity) and frame-level dropout as in [Liu+25b] with DSLNet.

Another potential experiment involves applying random cropping on the first and last frames to mitigate their influence and force frame interpolation. This would help in understanding how changing the execution speed of some signs affects the overall performance.

Furthermore, replacing certain frames with an interpolation of the previous and next ones like Roh et al. [Roh+] preprocessing but even for non-missing landmarks. This could help clean the data and also generate additional similar samples. Here, the MediaPipe output is kept as-is to avoid introducing extra assumptions and to keep the pipeline focused on modeling rather than post-hoc correction of pose estimation errors.

Developing data loading methods would have been valuable, like the DCT representation of the Landmarks would allow a comparison with the window of frames. Other data loaders have been imagined too: pos2vec, raw frames and relative coordinates. The first one uses the same method as Spacy word2vec but for skeleton frames. A model is trained to embed the frames with a closer embedding for close frames and a different embedding for distant frames. Then, the classifier uses the embedding as input instead of the window of frames. The second data loader, just give the frames as they are, without forcing them to fit a specific window size, it would work only on recurrent models and might deteriorate the training time, but this approach remains worthwhile to explore. The third data loader use relative coordinates between the landmarks. The wrist coordinate is relative to the elbow which is relative to the shoulder, etc.

Reproducibility constraints

Difficulties of reproducibility: While the best precautions were taken to make the experiments reproducible, the oldest experiments are not reproducible with the latest commit. To fully reproduce the experiments, you will need to go to the commit of the experiment and use the exact same config. Reproducing those results is still possible but can be difficult and time-consuming.

In summary, this chapter has interpreted the empirical findings through the lenses of model behavior, data quality, hardware constraints, and methodological trade-offs. The concluding chapter will synthesize these insights, connects them back to the original research questions, and reflects on the broader impact and future prospects of skeleton-based ISLR.

6 Conclusion

> Every new discovery is just a beginning, not an end.
>
> Marie Curie (paraphrased)

6.1 Summary of the Work

This thesis investigated skeleton-based ISLR under severe data scarcity and consumer hardware constraints, using a modular pipeline to explore data processing, model architectures, and hyperparameters. On the data side, the work shows that raw JSON skeleton files are too large for practical loading, whereas conversion to compressed NPZ format substantially reduces storage requirements. Among the tested window augmentation methods, copy and interpolation methods achieve comparable performance. Copy is slightly preferable, likely because it preserves the original signing speed. The different window selection strategies also yield close results, with the beta-based selection being marginally more effective than the other strategies. By contrast, window size has a much stronger impact on performance: a fixed size of 120 frames emerges as a good compromise between discarding information and generating artificial frames, and is more strongly correlated with model accuracy than the specific augmentation or selection method. Landmark-level augmentation further improves performance while keeping the skeleton trajectories realistic.

On the model side, the experiments showed that overfitting behavior is primarily an intrinsic property of the architecture rather than a consequence of individual hyperparameters such as learning rate, dropout, or hidden size. Scatter plots of training versus validation accuracy reveal that each model family is well-approximated by its own linear

regression, with the slope characterizing its degree of overfitting. Temporal Convolutional Networks (TCN) achieve the highest accuracy, top-5 accuracy, and F1-score among the evaluated models. The corresponding regression slope of about 0.7 indicates moderate overfitting, less severe than for the other architectures. GRU reaches approximately 0.52 accuracy and 0.78 top-5 accuracy but exhibit stronger overfitting (slope ≈ 0.5) and higher training time than attention-based models. MHA attains around 0.48 accuracy and 0.79 top-5 accuracy with very favorable training time and GPU memory usage, making it competitive under a tight resource budget. Linear models remain around 0.20 accuracy and serve as weak baselines, but offer the fastest inference and illustrate the cost of discarding temporal modeling entirely.

The dataset-level analysis highlights the importance of domain and evaluation protocol. Performance on WASL-100 is much lower (for instance, about 0.21 accuracy on the 2D signer-independent split) than on the reduced WASL-20 2D signer-independent subset, where the best model reaches roughly 0.55 accuracy, revealing a strong dataset and domain gap. The more homogeneous, semi-non-professional signing style of citizen-50 appears easier, while WASL exposes the generalization limits of the models. As expected, signer-independent evaluation is harder than signer-dependent, but the gap is smaller than intuition alone would suggest, indicating that the models do capture some signer-invariant structure. No class is systematically confused with another, although certain pairs, such as NOON and AXE, remain challenging due to subtle spatial differences in otherwise similar motion patterns.

Finally, the thesis quantifies the frugality and efficiency of the proposed pipeline and methodology. Deep models operate with only 0.1–0.5 GB of GPU memory in bfloat16 precision, with the best TCN run using approximately 0.64 GB in float32. Training times per epoch range from 0.2 s for linear models to about 10 s on the condition that hidden size remains ≤ 128. For the largest recurrent architectures on an eight-year-old 2 GB GTX 1050 GPU. Inference for the test set completes in under one second on CPU, making near real-time feedback plausible. The sweep-based experimental protocol enabled the exploration of roughly 800 configurations, corresponding to about 5.8 days of aggregated GPU time, without manual tuning of each run. At the same time, the reliance on Bayesian optimization introduces non-uniform sampling of hyperparameters, which limits the statistical robustness of some ablation results and reflects an inherent trade-off between searching aggressively for good hyperparameters and obtaining uniformly sampled evidence about which values are truly better.

6.2 Implications for ISLR

Methodological implications

The results indicate that TCN-like temporal convolutions are a strong default choice for temporal skeleton data classification. Their advantage comes in few-shot, high-dimensional data scenarios where they can learn to exploit temporal structure.

More generally, representation choices interact with the few-shot regime: naively switching from 2D to 3D landmarks does not guarantee better performance and may increase overfitting.

Practical implications

Training models for skeleton-based ISLR can be feasible on modest hardware. In this work, the full experimentation loop ran on a 2 GB GPU with manageable training and inference times, indicating that rapid iteration is possible without large-scale compute. The remaining bottleneck is often storage and experiment management rather than raw throughput.

This pipeline can serve as a baseline for future research on skeleton ISLR by reducing implementation overhead and enabling more time to be spent on research questions and principled comparisons.

6.3 Future Work

Expanding this work

Several extensions naturally follow from this work. First, evaluating on additional datasets and more diverse subsets (more signers, backgrounds, and signing styles) would clarify how well the current conclusions transfer across domains. Second, future work could expand the set of skeleton representations and loaders, for instance by revisiting frequency-domain encodings with less aggressive truncation, or by learning compact frame embeddings before classification. Third, the modular design makes it straightforward to integrate new temporal architectures and compare them under a controlled protocol, including hybrid models that combine convolutional front-ends with lightweight attention to capture both local dynamics and longer-range structure.

Experiments and reproducibility

While the experimental campaign in this thesis is extensive, several methodological improvements would strengthen future conclusions. A first direction is to increase the statistical robustness of ablation claims by using replicated runs for key configurations (multiple seeds per setting) and reporting uncertainty estimates (e.g. confidence intervals over repeated runs). A second direction is to complement Bayesian sweeps with designs that provide more uniform coverage for ablation analysis, such as stratified random sampling, partially factorial grids over the most influential variables (model family, window size, and signer split), or two-stage protocols that first identify promising regions and then perform controlled comparisons within those regions.

In addition, explicitly testing interactions between data hyperparameters and model hyperparameters would address the current assumption of weak correlation between these groups. For example, joint experiments could quantify whether the optimal window size and augmentation intensity depend on the architecture family, or whether resource-efficient settings systematically change the ranking between models.

Finally, reproducibility can be improved by packaging the experimental artifacts needed to rerun and audit results: versioned configuration files for the best runs, fixed dataset manifests for each split, and scripts that regenerate tables and figures from logged metrics. Given the existing logging infrastructure, these steps would make it easier for future work to extend this thesis and for other researchers to validate and compare new methods under the same protocol.

A Reproducibility

The reproducibility is a long known problem in research, particularly in this underdeveloped field of ISLR. Some datasets could not be used, and some projects could not be replicated. Even previous results from this work were sometimes not reproducible. For this reason, particular attention was paid to reproducibility so that the results would be as easy as possible to reproduce.

A.1 Find the Code

For further details, the associated repository[1] can be consulted.

Figure A.1: QR code to the repository of the project

Note: A readme was added in each code folder to help understand how to use the code, along with a wiki with the same content.

[1] https://framagit.org/elixio/recherche

A.2 Install the Tools

The following tools are required:

- git
- Python 3.10.16
- The libraries in the requirements.txt files (note that the exact version was used)
- `uv` is recommended as a replacement for pip (optional)

Then clone the repository and install the dependencies.

A.3 Get the Same Data

- `ASL citizen` (**advised**): reproducible as long as `https://download.microsoft.com/download/b/8/8/b88c0bae-e6c1-43e1-8726-98cf5af36ca4/ASL_Citizen.zip` gives the same file
- `WASL` (**not advised**): not completely reproducible as some videos links might not be available anymore (some are already not working)

Further information can be found in the README present in each folder of the code.

The README files use the following structure:

- **Installation**: how to install the dependencies
- **Usage**: how to use the code
- **Examples**: how to use the code with examples and expected effect (hash of the output if applicable)
- **Tests**: how to run the tests
- **Architecture**: file structure and description of the files

For the data pipeline, the results can be reproduced by using the same commands and arguments to generate the same citizen-50 dataset.

A.4 Replicate the Experiments

The README present in each folder of the code is recommended reading; one can then familiarize oneself with the code by running some commands.

Experiments were managed with W&B. For a single run, the same hyperparameters can be used. For a sweep, the same sweep file configuration should be used.

To allow for the replication of the experiments, the config files have been saved in the repository along with a script that allows to reproduce a run with the exact same config.

Note: In a sweep, the hyperparameters values of a run cannot be guaranteed to be the same or in the same order, so results might differ slightly.

Glossary

`num_frames` Length of the sequence of a data element. The number of frames in a given skeleton in `raw_data` is the same as in the corresponding video. 23

`window_size` Number of frames to be used as input for the model, it is a constant value for all data elements of a set. 23

AdamW AdamW is an optimizer that combines the benefits of Adam and L2 regularization. 35

bfloat16 Brain floating point 16-bit format. A 16-bit floating-point format that preserves the same exponent range as float32, enabling mixed-precision training with reduced memory usage. 28

Bone A line between two landmarks, can be also be called a connection or an edge. 17

Cross Entropy Cross Entropy is a loss function used in classification problems, it is the negative log-likelihood of the true distribution given the predicted distribution. 35

CUDA CUDA is a parallel computing platform and programming model developed by NVIDIA for general-purpose computing on graphical processing units (GPUs). 25, 30, 33

CUDA Deep Neural Network It is an highly-optimized library of deep-learning primitives for NVIDIA GPUs. 28, 30, 33, 65

cuML cuML is a library for machine learning on GPUs. 30, 33, 37

Curse of Dimensionality The phenomenon where the volume of space increases so fast that the available data becomes sparse, making machine learning algorithms perform poorly in high-dimensional spaces. 38

cysimdjson cysimdjson is an optimized library for using JSON in python. 22

Dynamic Signs Signs that involve movement and which cannot be represented with a single frame. 4

Dynamic Task The task of recognizing Static Signs or Dynamic Signs from a sequence of images or skeletons. 4

Early Stopping Early Stopping is a technique that stops the training of the model when the validation loss stops improving. 35, 53, 57, 64

Early Terminate Early Terminate is like early stopping but it uses sweep data to determinate when to stop. Instead of relying only on the same run data it compare the run with the others.. 64

HDF5 Hierarchical Data Format version 5 is a binary hierarchical data format for complex datasets, used in scientific computing.. 22

Holistic When the pose estimation model takes not only the body but also the hands and the face into account. 14, 16

Hugging Face Hugging Face is a library for deep learning in Python. 63

JSON JavaScript Object Notation is a text-based data format that represents structured data as human-readable key-value pairs. 22, 67

Landmark Correspond to a joint in the human body. List of values that represent a coordinate of a point (can be 2D or 3D with or without visibility). 2, 9, 10, 15–17, 20, 21, 23, 25, 39, 42–44, 46, 59, 66

Learning Rate Learning Rate is the step size used to update the weights of the model. 44–46, 48, 51

Learning Rate Scheduler Learning Rate Scheduler is a function that updates the learning rate of the model. 35, 57, 64

MediaPipe MediaPipe is used to extract the skeleton from the video. 3, 6, 10, 14, 16, 21, 22, 26, 65

NPZ NumPy's compressed file format. 20, 22, 67

Overfitting A modeling error that occurs when a function is too closely fit to a limited set of data points, resulting in poor performance on new data. 2, 36

Parquet columnar, compressed storage format for tabular data and efficient analytics and selective column reads.. 22

PT PyTorch's file format. 22

PyTorch PyTorch is a library for deep learning in Python. 25, 30, 33, 37

scikit-learn scikit-learn is a library for machine learning in Python. 37

Secondary Metrics Metrics used as extra signal to help understand or compare the model, e.g. number of epochs, weighted F1, inference time, GPU memory usage, etc.. 36, 44

Signer-dependent Recognition systems trained and tested on the same signer, typically achieving higher accuracy due to consistent signing style. 17

Signer-independent Recognition systems tested on signers not seen during training, representing a more challenging and realistic scenario. 11, 17

SIMD Single Instruction Multiple Data is a parallel computing technique that executes multiple data operations in parallel on the same data. 22

Skorch Wrapper for PyTorch that allows to use PyTorch models with scikit-learn API. 26, 28, 63

Spacy Spacy is a library for natural language processing in Python. 66

State-of-the-art The best performance of the task at the time of the publication. 52

Static Signs Signs that can be represented with a single image or pose, requiring no temporal information for recognition. 4

Static Task The task of recognizing Static Signs from a single image or pose. 4

t-SNE t-Distributed Stochastic Neighbor Embedding is a non-linear dimensionality reduction method that tries to place points so close points stay close and far points stay far. 64

TensorBoard TensorBoard is a tool for visualizing data, profiler or models in TensorFlow and PyTorch. 30–32

Torch Profiler Profiler for PyTorch, helps to get CPU and GPU usage of functions and computations.. 30

uv uv is a faster python tool that replace pip, poetry, ... 72

W&B Weights and Biases is a tool for logging and tracking machine learning experiments. 28, 29, 40–48, 53, 65, 73

Zarr Zarr is a chunked, compressed storage format for large multidimensional arrays, used in cloud. 20, 22

Acronyms

3DGCN 3D Graph Convolutional Network. 8

ASL-Citizen American Sign Language Citizen dataset. 8, 20, 21, 24, 63

ASL-Signs American Sign Language Signs dataset. 8, 22

AVSN Adversarial Vulnerability-Seeking Networks. 10

Compute Capability Version of the GPU. Hardware information to know which operations can be performed on the GPU.. 28, 30, 33, 34

CSLR Continuous Sign Language Recognition. 4, 5, 12

DCT Discrete Cosine Transform. 34, 59, 66

DSLNet Dual-SignLanguageNet. 7–9, 65

GRU Gated Recurrent Unit. iii, 26, 37, 38, 44, 45, 49, 51, 52, 60, 61, 68

hLin Hierarchical Linear. 38

HLSTM Hierarchical Long Short-Term Memory. 38

HWGAT Holistic and Weighted Graph Attention Network. 8

I3D Inflated 3D ConvNets. 7–9

ISLR Isolated Sign Language Recognition. iii, 2–6, 12, 17, 19, 34, 57, 60, 66, 67, 69, 71

JMDA Joint Mixing Data Augmentation. 11

KNN K-Nearest Neighbors. 34, 36

Lin Linear. 26, 38, 44, 49, 52, 60

LSA64 Argentinian Sign Language dataset. 7, 14, 23

LSTM Long Short-Term Memory. 26, 37, 38, 44

MHA Multi-Head Attention. iii, 26, 37, 38, 44, 46, 49, 51, 52, 60, 68

MoE Mixture of Experts. 29

MS-ASL Microsoft American Sign Language dataset. 8, 21

MSNN Multi-Stream Neural Network. 9

PGF Prior-Guided Fusion. 9

RF Random Forest. 34, 36

RNN Recurrent Neural Network. 37, 44

SignBart Sign Language Bart. 7, 8, 10, 14

SignBERT Sign Language BERT. 7, 9

SKiM Skeleton-Based Isolated Sign Language Recognition with Part Mixing. 10, 65

SL-GCN Sign Language GCN. 8

SLR Sign Language Recognition. 1, 4–6, 10

SLT Sign Language Translation. 5

SPOTER Sign POse-based TransformER. 7–10

ST-GCN Spatial-Temporal GCN. 7–9

SVM Support Vector Machine. 34, 36

TCN Temporal Convolutional Network. iii, 26, 37, 38, 44, 47, 49, 51, 52, 60, 61, 68, 69

Transformer Transformer architecture. 26, 38, 44

Uni-Sign Unified Sign Language Recognition. 9

WASL Word-Level American Sign Language dataset. 7, 20, 21, 24, 52, 55, 56, 58, 63, 68

Bibliography

[BH22] Matyas Bohacek and Marek Hruz. "Sign Pose-based Transformer for Word-level Sign Language Recognition". en. In: *2022 IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW)*. Waikoloa, HI, USA: IEEE, Jan. 2022, pp. 182–191. ISBN: 978-1-6654-5824-5. DOI: 10.1109/WACVW54805.2022.00024. URL: https://ieeexplore.ieee.org/document/9707552/ (visited on 03/06/2026) (cit. on pp. 9, 10).

[Cao+21] Zhe Cao et al. "OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.1 (Jan. 2021), pp. 172–186. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2019.2929257. URL: https://ieeexplore.ieee.org/document/8765346 (visited on 09/05/2025) (cit. on pp. 6, 14).

[Des+] Aashaka Desai et al. "ASL Citizen: A Community-Sourced Dataset for Advancing Isolated Sign Language Recognition". en. In: () (cit. on p. 8).

[Hu+21] Hezhen Hu et al. "SignBERT: Pre-Training of Hand-Model-Aware Representation for Sign Language Recognition". en. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. Montreal, QC, Canada: IEEE, Oct. 2021, pp. 11067–11076. ISBN: 978-1-6654-2812-5. DOI: 10.1109/ICCV48922.2021.01090. URL: https://ieeexplore.ieee.org/document/9709967/ (visited on 03/06/2026) (cit. on p. 9).

[Joz] Hamid Reza Vaezi Joze. "MS-ASL: A Large-Scale Data Set and Benchmark for Understanding American Sign Language". en. In: () (cit. on p. 8).

[Ke+13] Shian-Ru Ke et al. "A Review on Video-Based Human Activity Recognition". en. In: *Computers* 2.2 (June 2013), pp. 88–131. ISSN: 2073-431X. DOI: 10.3390/computers2020088. URL: https://www.mdpi.com/2073-431X/2/2/88 (visited on 09/05/2025) (cit. on p. 6).

[Lai+23] David Laines et al. "Isolated Sign Language Recognition based on Tree Structure Skeleton Images". en. In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Vancouver, BC, Canada: IEEE, June 2023, pp. 276–284. ISBN: 979-8-3503-0249-3. DOI: 10.1109/CVPRW59228.2023.00033. URL: https://ieeexplore.ieee.org/document/10208922/ (visited on 09/07/2025) (cit. on p. 6).

[Li+20] Dongxu Li et al. "Word-level Deep Sign Language Recognition from Video: A New Large-scale Dataset and Methods Comparison". en. In: *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*. Snowmass Village, CO, USA: IEEE, Mar. 2020, pp. 1448–1458. DOI: 10.1109/wacv45572.2020.9093512. URL: https://ieeexplore.ieee.org/document/9093512/ (visited on 07/26/2025) (cit. on pp. 7, 9).

[Li+25] Zecheng Li et al. "UNI-SIGN: TOWARD UNIFIED SIGN LANGUAGE UN- DERSTANDING AT SCALE". en. In: (2025) (cit. on pp. 5, 9).

[Lin+24] Kezhou Lin et al. "SKIM: Skeleton-Based Isolated Sign Language Recognition With Part Mixing". In: *IEEE Transactions on Multimedia* 26 (2024), pp. 4271–4280. ISSN: 1941-0077. DOI: 10.1109/TMM.2023.3321502. URL: https://ieeexplore.ieee.org/document/10464382 (visited on 07/26/2025) (cit. on pp. 10, 65).

[Liu+25a] Liangjin Liu et al. *Skeleton-based sign language recognition using a dual-stream spatio-temporal dynamic graph convolutional network*. en. arXiv:2509.08661 [cs]. Sept. 2025. DOI: 10.48550/arXiv.2509.08661. URL: http://arxiv.org/abs/2509.08661 (visited on 03/06/2026) (cit. on p. 9).

[Liu+25b] Mengyuan Liu et al. *3D Skeleton-Based Action Recognition: A Review*. en. arXiv:2506.00915 [cs]. June 2025. DOI: 10.48550/arXiv.2506.00915. URL: http://arxiv.org/abs/2506.00915 (visited on 10/12/2025) (cit. on p. 65).

[Lug+19] Camillo Lugaresi et al. *MediaPipe: A Framework for Building Perception Pipelines*. en. arXiv:1906.08172 [cs]. June 2019. DOI: 10.48550/arXiv.1906.08172. URL: http://arxiv.org/abs/1906.08172 (visited on 09/05/2025) (cit. on pp. 6, 14, 16, 22).

[Mar+24] Mizuki Maruyama et al. "Word-Level Sign Language Recognition With Multi-Stream Neural Networks Focusing on Local Regions and Skeletal Information". In: *IEEE Access* 12 (2024), pp. 167333–167346. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2024.3494878. URL: https://ieeexplore.ieee.org/document/10749796/ (visited on 09/08/2025) (cit. on p. 9).

[NJ25] Yuriya Nakamura and Lei Jing. "Skeleton-Based Data Augmentation for Sign Language Recognition Using Adversarial Learning". In: *IEEE Access* 13 (2025), pp. 15290–15300. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2024.3481254. URL: https://ieeexplore.ieee.org/document/10718297 (visited on 07/26/2025) (cit. on p. 10).

[NT25] Tinh Nguyen and Minh Khue Phan Tran. *SignBart – New approach with the skeleton sequence for Isolated Sign language Recognition*. en. arXiv:2506.21592 [cs]. June 2025. DOI: 10.48550/arXiv.2506.21592. URL: http://arxiv.org/abs/2506.21592 (visited on 09/07/2025) (cit. on pp. 8, 10, 14).

[Roh+] Kyunggeun Roh et al. "Preprocessing Mediapipe Keypoints with Keypoint Reconstruction and Anchors for Isolated Sign Language Recognition". en. In: () (cit. on pp. 10, 65).

[Ron+23] Franco Ronchetti et al. *LSA64: An Argentinian Sign Language Dataset*. en. arXiv:2310.17429 [cs]. Oct. 2023. DOI: 10.48550/arXiv.2310.17429. URL: http://arxiv.org/abs/2310.17429 (visited on 03/06/2026) (cit. on p. 7).

[WZA22] Quanyu Wang, Kaixiang Zhang, and Manjotho Ali Asghar. "Skeleton-Based ST-GCN for Human Action Recognition With Extended Skeleton Graph and Partitioning Strategy". In: *IEEE Access* 10 (2022), pp. 41403–41410. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2022.3164711. URL: https://ieeexplore.ieee.org/document/9749063/ (visited on 01/27/2026) (cit. on p. 9).

[XW25] Linhua Xiang and Zengfu Wang. "Joint Mixing Data Augmentation for Skeleton-Based Action Recognition". en. In: *ACM Transactions on Multimedia Computing, Communications, and Applications* 21.4 (Apr. 2025), pp. 1–24. ISSN: 1551-6857, 1551-6865. DOI: 10.1145/3700878. URL: https://dl.acm.org/doi/10.1145/3700878 (visited on 07/26/2025) (cit. on p. 11).

[ZJ24] Yanqiong Zhang and Xianwei Jiang. "Recent Advances on Deep Learning for Sign Language Recognition". en. In: *Computer Modeling in Engineering & Sciences* 139.3 (2024), pp. 2399–2450. ISSN: 1526-1506. DOI: 10.32604/cmes.2023.045731. URL: https://www.techscience.com/CMES/v139n3/55626 (visited on 07/26/2025) (cit. on p. 4).

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich diese Master thesis selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe und alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, als solche gekennzeichnet sind, sowie, dass ich die Master thesis in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt habe.

Lyon, May 24, 2026

Gaspard Serpinet