UNIVERSITÀ DEGLI STUDI DI CATANIA
Dipartimento di Matematica e Informatica
Corso di Laurea Triennale in Informatica

Antonio Finocchiaro

Human Body Pose Estimation Missing Keypoints Reconstruction through Interpolations

Report on Multimedia and Laboratory project

Professors:
Dario Allegra
Filippo Stanco

Academic Year 2023 - 2024

Contents

1 Introduction **1**

2 Prerequisites **3**
2.1 Interpolation Methods . . . 3
2.2 Distribution Evaluation Metrics . . . 7
2.3 Performance Metrics in Classification . . . 9
2.4 Models . . . 10
2.5 Python and Related Libraries . . . 12

3 Dataset **14**
3.1 Video Dataset . . . 14
3.2 Keypoints Dataset . . . 16

4 Proposed Method **18**
4.1 Interpolations Performance on Reference Data . . . 18
4.1.1 Ground Truth Dataset Creation . . . 18
4.1.2 Missing Sequence Generation . . . 19
4.2 Multiclass Classifier . . . 19
4.2.1 Multilayer Perceptron Architecture . . . 19

5 Analyses and Results **21**
5.1 Results of Interpolations Performance on Reference Data . . . 21
5.1.1 Reconstructed Joints Visual Inspections . . . 22
5.1.2 Methods Distributions Trends Analysis . . . 24
5.2 Multilayer Perceptron Analyses and Results . . . 27
5.2.1 Hyperparameters Tuning . . . 27
5.2.2 Test Results . . . 29

6 Conclusions **31**

Chapter 1

Introduction

In this work, we focus on Human Pose Estimation (HPE), a field dedicated to determining the pose of a human body by estimating the 2D or 3D spatial position of its joints. HPE finds widespread use in sports contexts for understanding movements. Specifically, we apply HPE to classify Calisthenics isometric elements. For this purpose, we utilize OpenPose as the chosen pose estimator, configured specifically for 2D pose estimation.

The discipline of Calisthenics comprises various branches, including endurance, strength, and skills. In recent years, each of these branches has experienced significant growth in competitive arenas. Notably, the skills branch holds a particularly influential position on the international stage due to the demanding strength requirements of the performed poses. Calisthenics encompasses a wide range of skills and their variations. However, for this specific project, a carefully selected subset of skills has been chosen.

This work offers a comprehensive overview of estimating missing joint spatial information through six interpolation methods. Missing values can be due to various factors such as lighting conditions and background influences. Each interpolation will be thoroughly examined and utilized to estimate the missing values.

The primary task is to recognize and classify Calisthenics skills from video footage. This work's applications include monitoring athletes' skill execution during competitions and serving as a training tool to simulate a judge's role. To accomplish the project's ultimate goal, several sequential steps must be completed to successfully address various subtasks.

The process begins with the selection of skills to recognize and the creation of a video dataset. Subsequently, a pre-trained pose estimator is utilized to extract spatial information about the athletes' joints in the videos. Following this, a dataset is constructed from the frame-level body pose tracking results. A subset of this dataset is then used as reference data to evaluate the performance of different interpolation methods. The original keypoint dataset is filled with various interpolations, and all these datasets are utilized to train and test a multiclass classifier designed to detect poses.

Chapter 2

Prerequisites

In this section we present the mathematical processes, the model and the tools involved through this work.

2.1 Interpolation Methods

In this project, our main focus is on interpolation, it is like a way to connect the dots when we have some data points, but there are gaps in between. We are using this technique to estimate what is happening with an athlete's body joints when we do not have all the data. Sometimes, the pose estimator that we use to detect these joints does not work perfectly, leaving us with missing data at certain points in time or positions. So, interpolation comes to the rescue. It helps us smooth out these gaps and create a complete picture of what is happening over time.

The interpolations used in this work are the following:

- Nearest Neighbor
- Linear
- Inverse Distance Weight
- Akima
- Pchip
- Spline

Nearest Neighbor

Nearest Neighbor (Nearest) interpolation is one of the simplest interpolation techniques. It works by assigning the value of the nearest data point to the location we want to interpolate.

Formally, given a set of data points (x_i, y_i), where $i = 1, 2, ..., n$ represents the known data points, and we want to estimate the value y at a specific location x, Nearest Neighbor Interpolation can be defined as:

$$\text{Nearest Neighbor}(x) = y_k, \text{ where } k = \arg\min_i |x - x_i| \tag{2.1}$$

where:
x: The location where we want to estimate the value.
x_i: The x-coordinates of the known data points.
y_i: The corresponding y-values.
k: The index of the nearest data point to x, found by minimizing the absolute difference between x and x_i.
y_k: The value at the nearest data point, which is used as the interpolated value at x.

In practice, this will result in a step function.

Linear

Linear interpolation is a straightforward method to estimate values between two known data points. It assumes that the relationship between data points is approximately linear. In other words, it draws a straight line between two adjacent data points and calculates the value at a desired position along that line.

Suppose we have two data points, (x_0, y_0) and (x_1, y_1), and we want to estimate a value y at a position x that falls between x_0 and x_1. Linear interpolation can be expressed as:

$$y = y_0 + \frac{(x - x_0)}{(x_1 - x_0)} \cdot (y_1 - y_0) \tag{2.2}$$

Here is what each variable represents:
- y: The estimated value at the position x within the range $[x_0, x_1]$.
- x_0: The x-coordinate of the first data point.
- y_0: The y-coordinate of the first data point.
- x_1: The x-coordinate of the second data point.
- y_1: The y-coordinate of the second data point.
- x: The position where we want to estimate the value y.

Inverse Distance Weight

Inverse Distance Weight (IDW) interpolation is a method used to estimate values at unmeasured locations based on values at nearby measured locations. It assumes that values at a given location are influenced by the values at other locations, with the influence decreasing as the distance between the locations increases.

In mathematical terms, the formula for IDW can be expressed as follows:

The interpolated value at a location (x, y) can be denoted as z_0, and it is calculated as a weighted average of the known values at nearby locations. The weight assigned to each known value is inversely proportional to the distance between the unknown location and the known location. The general formula for IDW interpolation is:

$$z_0 = \frac{\sum_{i=1}^{n} \frac{z_i}{d_i^p}}{\sum_{i=1}^{n} \frac{1}{d_i^p}} \tag{2.3}$$

where:
z_0 is the interpolated value at the location (x, y).
z_i is the known value at the ith location.
d_i is the distance between the interpolated location (x, y) and the ith known location.
p is a user-defined positive power parameter that controls the influence of distance. Common values for p are 1 (inverse linear distance) and 2 (inverse squared distance).
n is the total number of known locations used in the interpolation.

This formula essentially calculates a weighted average of the known values, with the weights based on the inverse of distances raised to power p. The larger the distance, the smaller the weight, and vice versa. The power parameter p controls the rate at which the influence of distance diminishes.

Akima

Akima interpolation is a method for estimating values between data points using piecewise polynomials. It provides a smooth and continuous approximation of a function based on the given data points.

Akima interpolation is particularly useful when dealing with unevenly spaced data points.

For each interval $[x_i, x_{i+1})$ of an input data set of nodes x and values v, piecewise cubic Hermite interpolation finds a cubic polynomial which not only interpolates the given data values v_i and v_{i+1} at the interval's nodes x_i and x_{i+1}, but also has specific derivatives d_i and d_{i+1} at x_i and x_{i+1}.

The key to cubic Hermite interpolation is the choice of derivatives d_i.

Let $\delta_i = \frac{v_{i+1}-v_i}{x_{i+1}-x_i}$ be the slope of the interval $[x_i, x_{i+1})$. Akima's derivative at x_i is defined as:

$$d_i = \frac{|\delta_{i+1} - \delta_i|\delta_{i-1} + |\delta_i - \delta_{i-1}|\delta_{i+1}}{|\delta_{i+1} - \delta_i| + |\delta_i - \delta_{i-1}|},$$

and represents a weighted average between the slopes δ_{i-1} and δ_i of the intervals $[x_{i-1}, x_i)$ and $[x_i, x_{i+1})$:

$$d_i = \frac{w_1}{w_1 + w_2}\delta_{i-1} + \frac{w_2}{w_1 + w_2}\delta_i,$$

where $w_1 = |\delta_{i+1} - \delta_i|$, $w_2 = |\delta_i - \delta_{i-1}|$.

It is worth noting that Akima's derivative at x_i is computed locally from the five points x_{i-2}, x_{i-1}, x_i, x_{i+1}, and x_{i+2}. For the end points x_1 and x_n, it requires the slopes δ_{-1}, δ_0 and δ_n, δ_{n+1}. Since these slopes are not available in the input data, Akima proposed using quadratic extrapolation to compute them as $\delta_0 = 2\delta_1 - \delta_2$, $\delta_{-1} = 2\delta_0 - \delta_1$ and $\delta_n = 2\delta_{n-1} - \delta_{n-2}$, $\delta_{n+1} = 2\delta_n - \delta_{n-1}$.

Compared to the spline algorithm, the Akima algorithm produces fewer undulations and is better suited to deal with quick changes between flat regions. Compared to the Pchip algorithm, the Akima algorithm is not as aggressively flattened and is therefore still able to deal with oscillatory data.

Pchip

Pchip (Piecewise Cubic Hermite Interpolating Polynomial) interpolation is a method for approximating a curve that passes through a set of given data points. It is particularly useful when we want a smooth and continuous interpolation, preserving the shape and monotonicity of the data.

Pchip interpolates using a piecewise cubic polynomial $P(x)$ with these properties:

On each subinterval $x_k \leq x \leq x_{k+1}$, the polynomial $P(x)$ is a cubic Hermite interpolating polynomial for the given data points with specified derivatives (slopes) at the interpolation points.

$P(x)$ interpolates y, that is, $P(x_j) = y_j$, and the first derivative $\frac{dP}{dx}$ is continuous.

The second derivative $\frac{d^2P}{dx^2}$ is probably not continuous, so jumps at the x_j are possible. The cubic interpolant $P(x)$ is shape-preserving. The slopes at the x_j are chosen in such a way that $P(x)$ preserves the shape of the data and respects monotonicity.

Therefore, on intervals where the data is monotonic, so is $P(x)$, and at points where the data has a local extremum, so does $P(x)$.

Spline

In Spline interpolation, we aim to approximate a function or a curve, $f(x)$, using piecewise defined cubic polynomials. These cubic polynomials, known as "spline functions," are combined to create a smooth and continuous curve. The spline passes through given data points (x_i, y_i) for $i = 0, 1, 2, \ldots, n$.

A cubic spline consists of n cubic polynomials, each defined on a subinterval $[x_i, x_{i+1}]$, where i ranges from 0 to $n-1$. The spline is represented as:

$$S(x) = \begin{cases} S_0(x) & \text{if } x_0 \leq x \leq x_1 \\ S_1(x) & \text{if } x_1 \leq x \leq x_2 \\ \vdots \\ S_{n-1}(x) & \text{if } x_{n-1} \leq x \leq x_n \end{cases}$$

Each $S_i(x)$ is a cubic polynomial, typically expressed in the form:

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

The coefficients a_i, b_i, c_i, and d_i are determined by imposing various conditions to ensure continuity and smoothness at the data points.

2.2 Distribution Evaluation Metrics

In this section, we provide an overview about the evaluation metics used to compare the ground truth joints distribution with the interpolated versions.

Euclidean Distance

The Euclidean distance is a fundamental distance metric used to measure the straight-line distance between two points in Euclidean space. In mathematical notation, the Euclidean distance between two points, say A and B, with coordinates (x_1, y_1, z_1) and (x_2, y_2, z_2), respectively, in three-dimensional space is calculated as:

$$d(A, B) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

In this formula: $d(A, B)$ represents the Euclidean distance between points A and B. x_1, y_1, and z_1 are the coordinates of point A. x_2, y_2, and z_2 are the coordinates of point B.

The formula essentially computes the square root of the sum of squared differences in coordinates for each dimension (in this case, x, y, and z). This yields the straight-line distance between the two points in a three-dimensional space.

Root Mean Square Error

The Root Mean Square Error (RMSE) is one of the most commonly used measures for evaluating the quality of predictions. It provides a comprehensive assessment of the accuracy of predictions by quantifying the average deviation between predicted values and actual values in a dataset.

$$\text{RMSE} = \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}}{n}$$

Where:

- n is the number of data points.
- y_i is the actual value of the dependent variable for the i^{th} observation.
- $\hat{y}_i$ is the predicted value of the dependent variable for the i^{th} observation.
- The term $(y_i - \hat{y}_i)^2$ represents the squared difference between the actual and predicted values for each observation.

Kullback-Leibler Divergence

The Kullback-Leibler (KL) divergence is a measure of how one probability distribution diverges from a second, expected probability distribution. It quantifies the difference between two probability distributions P and Q. In essence, KL divergence measures the information lost when Q is used to approximate P.

$$D_{KL}(P \parallel Q) = \sum_i P(i) \log\left(\frac{P(i)}{Q(i)}\right)$$

Where:

- $D_{KL}(P \parallel Q)$ denotes the KL divergence between distributions P and Q.
- $P(i)$ and $Q(i)$ represent the probabilities of event i occurring in distributions P and Q respectively.

- The summation is taken over all events in the sample space.
- The term $\frac{P(i)}{Q(i)}$ represents the ratio of the probability of event i in P to that in Q, providing a measure of how much more likely an event is under distribution P compared to distribution Q.
- The logarithm is applied to this ratio to emphasize the information gain or loss when approximating P with Q. Positive values indicate that Q overestimates P, while negative values indicate underestimation.

Interpretation: A KL divergence of 0 indicates that the two distributions P and Q are identical. A positive KL divergence implies that Q is a poor approximation of P, with the magnitude of divergence indicating the extent of the difference. However, KL divergence is not symmetric, meaning $D_{KL}(P \parallel Q) \neq D_{KL}(Q \parallel P)$. Thus, it is essential to consider the ordering of the distributions when interpreting the result.

2.3 Performance Metrics in Classification

When evaluating the performance of a classification model, several metrics are commonly used to assess its effectiveness. These metrics provide insights into different aspects of the model's performance.

Accuracy

Accuracy measures the proportion of correctly classified instances among all instances:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

Precision

Precision measures the proportion of true positive predictions among all positive predictions made by the model:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Recall

Recall, also known as sensitivity or true positive rate, measures the proportion of true positive predictions among all actual positive instances:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

F1 Score

The F1 score is the harmonic mean of precision and recall. It provides a balance between precision and recall:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

2.4 Models

OpenPose

OpenPose is an open-source library for multi-person 2D pose detection, including body, foot, hand, and facial keypoint. Tt is developed by CMU Perceptual Computing Lab. OpenPose uses a bottom-up representation of association scores via *Part Affinity Fields* (PAF) to learn to associate body parts with individuals in the image.

There are several models used in OpenPose, the one employed in this work is BODY_25B.

BODY_25B The *BODY_25B* model is an updated version of the *BODY_25* model with higher accuracy. This model includes some extra PAF (Part Affinity Field) channels and additional body keypoints, specifically the MPII head and neck keypoints. We opted to this model in our project due to its superior performance in accurately tracking body animation.

Fig. 2.1 showcases different body representations between BODY_25 and BODY_25B.

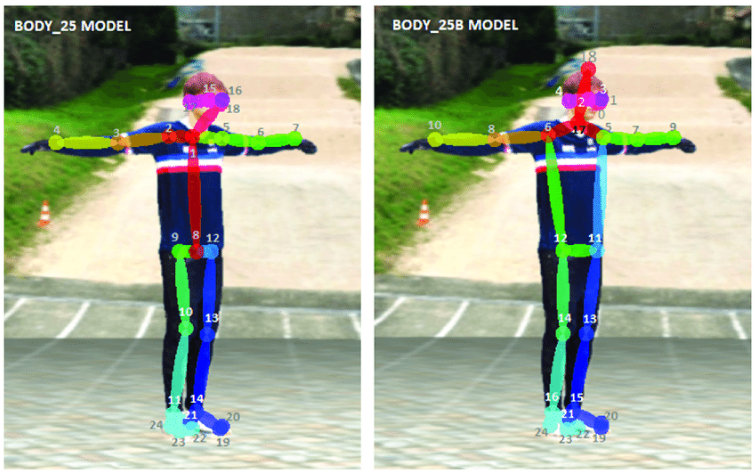

Figure 2.1: Unlike the *BODY_25* model, the neck and middle hip keypoints are not artificially created in the *BODY_25B* model. Instead, the neck keypoint is determined by the middle point of the shoulders, and the middle hip keypoint is determined by the middle point of the hips.[1]

Multilayer Perceptron

The Multilayer Perceptron (MLP) is an advanced type of artificial neural network that enhances the simple perceptron model by incorporating multiple layers of interconnected neurons in a feedforward network structure. MLPs can solve various problems such as classification, regression, and prediction, thanks to their multiple layers of neurons and non-linear activation functions. Unlike simple perceptrons, MLPs can handle complex and non-linear problems.

An MLP typically consists of an input layer, one or more hidden layers, and an output layer. The input layer receives data, with each neuron representing a single input feature. Hidden layers perform most of the computation, with neurons in each layer receiving input from the previous layer and producing an output based on weighted sums of inputs. During training, weights between neurons are adjusted to improve network performance.

Activation functions like Tanh, ReLU, and Sigmoid introduce non-linearity into the model, enabling the network to learn complex patterns and relationships in the data. The output layer produces the final network output, such as class predictions in a classification task.

Training an MLP involves backpropagation, where input-output pairs are used to adjust weights based on the error between predicted and true outputs. This iterative process continues until the error is minimized, achieving a satisfactory level of accuracy.

Optimizers are crucial in this process, adjusting network parameters during training to minimize error. They determine how weights are updated based on calculated errors, guiding the network toward better performance.

2.5 Python and Related Libraries

Python is a high-level, object-oriented, and interpreted programming language (version used: 3.10). Python offers developers many resources to create sophisticated and high-performance applications. Lots of its libraries have been used in this project, including Scipy, PyTorch, Pandas in particular.

Scipy SciPy is an open-source scientific computing library for Python. It builds upon the capabilities of NumPy, another fundamental Python library, by providing additional functionality for a wide range of scientific and engineering applications. Some of the involved modules are: *scipy.interpolate*, *scipy.special*, *scipy.stats*. The *interpolate* module provides a wide range of interpolation techniques for working with data. These techniques are essential for estimating values between known data points approximating functions from limited data. The module includes functions for linear, nearest, spline and others interpolation.

PyTorch PyTorch is an open source library primarily developed by Facebook AI Research. It offers a wide range of features for building machine learning models, including deep neural networks, optimization algorithms, loss functions, data visualization tools, and much more. PyTorch provides functionality for processing multidimensional arrays, similar to what is provided by Numpy, but with a more comprehensive and powerful library.

Pandas is a data manipulation library that offers a variety of powerful tools to quickly and efficiently analyze, clean, and transform data. It provides functionalities such as merging, reshaping, indexing, slicing, and grouping, which are essential for data manipulation. Its primary data structure, DataFrame, is a flexible and robust data structure that allows users to work with data in a tabular form, similar to a spreadsheet. In our project, Pandas has been extensively used to work with CSV files and to manipulate data in some algorithms.

Numpy is a numerical computing library for Python. It provides a multi-dimensional array object, along with a set of mathematical functions to operate on these arrays. It is widely used in scientific and data-intensive computing because of its ability to perform complex mathematical operations efficiently and effectively. It provides a high-performance array computing functionality and tools for working with them, such as linear algebra or random number generation.

Matplotlib Matplotlib is a Python library used for creating static, animated, and interactive visualizations. It offers a range of customization options to make graphs, histograms, scatterplots, and other visualizations.

Chapter 3

Dataset

In this chapter, we will discuss about the data we are working on.

To compile a numerical dataset detailing the keypoints of the human body, a video dataset has been established. The first step to address this task is to define a set of Calisthenics skills to be detected.

The chosen calisthenics skills for this task are the following:

- Back Lever (BL)
- Front Lever(FL)
- Human Flag (FLAG)
- Iron Cross (IC)
- Maltese (MAL)
- One Arm Front Lever (OAFL)
- One Arm Handstand (OAHS)
- Planche (PL)
- V-sit (VSIT)

3.1 Video Dataset

The video dataset comprises 839 clips sourced from popular social media platforms like YouTube and Instagram. Each video has been processed using *Kdenlive* software, ensuring a standardized resolution of 960x540 pixels at 24 frames per second (fps).

Table 3.1: Occurrences of each skill in our dataset in terms of number of videos, seconds and frames.

Skill	**Videos**	**Seconds**	**Frames**
Back Lever (BL)	88	574.66	13792
Front Lever (FL)	108	443.08	10634
Human Flag (FLAG)	75	633.16	15196
Iron Cross (IC)	77	513.08	12314
Maltese (MAL)	98	392.70	9425
One Arm Front Lever (OAFL)	80	363.50	8724
One Arm Handstand (OAHS)	94	606.45	14555
Planche (PL)	103	485.25	11646
V-sit (VSIT)	116	814.87	19557
Total	839	4826.79	115843

The Table 3.1 shows the video and frames occurrences of each class.

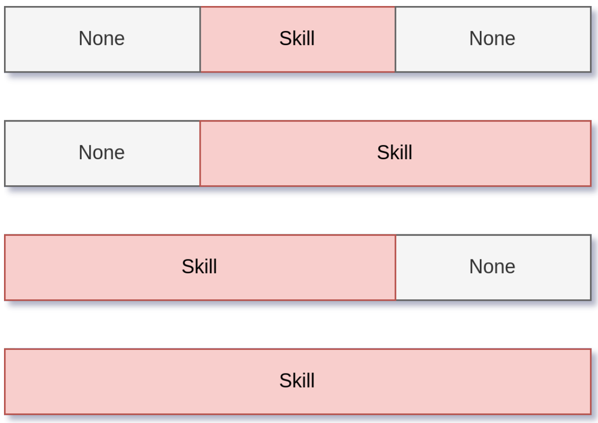

Figure 3.1: Video segments structure.

Each video in the dataset displays only one skill, with optional periods of 'NONE' before or after the skill, as shown in Fig. 3.1. On average, a video featuring a skill lasts about 5.83 seconds. The longest video lasts 27.83 seconds, while the shortest is just 0.83 seconds.

3.2 Keypoints Dataset

To facilitate research on calisthenics skill recognition using body pose analysis, we employed OpenPose to extract body poses from each frame.

Regarding the inner architecture of OpenPose, we fixed the 'net_resolution' parameter at 208, selected after several visual inspections. The 'number_people_max' flag was set to 1 to prioritize detecting the most prominent human in scenes with multiple subjects. While this setting does not guarantee that the recognized person is the athlete in multi-person scenes, optimal system performance is achieved when the video contains only the athlete.

As discussed in Section 2.4, the model used is MODEL_25B, capable of extracting information about 25 joints from a frame. For each joint, we obtain the X coordinate, Y coordinate, and a confidence score, with all three values ranging from 0 to 1, properly set through the 'pose_keypoints_2d' flag.

The detected joints are as follows:

1. **Nose**
2. **Left Eye**
3. **Right Eye**
4. **Left Ear**
5. **Right Ear**
6. **Left Shoulder**
7. **Right Shoulder**
8. **Left Elbow**
9. **Right Elbow**
10. **Left Wrist**
11. **Right Wrist**
12. **Left Hip**
13. **Right Hip**
14. **Left Knee**
15. **Right Knee**
16. **Left Ankle**
17. **Right Ankle**
18. **Upper Neck**
19. **Head top**
20. **Left Big Toe**
21. **Left Small Toe**
22. **Left Heel**
23. **Right Big Toe**
24. **Right Small Toe**
25. **Right Heel**

Table 3.2: Frame classes occurrences in the keypoints dataset.

Skills	BL	FL	FLAG	IC	MAL	NONE	OAFL	OAHS	PL	VSIT
Frame Occurrences	10236	7561	6964	9791	7309	23150	6389	7950	8694	13092

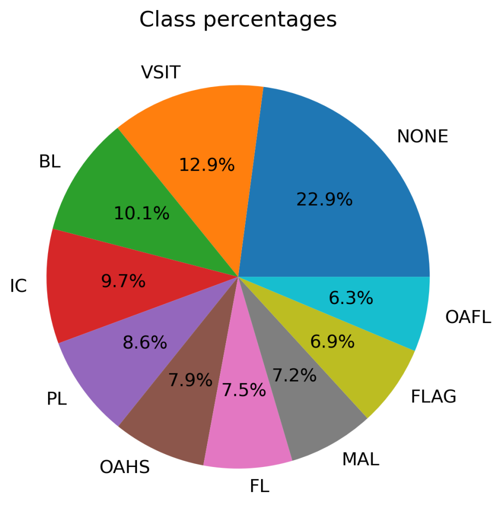

Figure 3.2: Frame classes percentage in the keypoints dataset.

The table in Table 3.2 illustrates the occurrences of classes within the dataset on a per-frame basis. Additionally, Fig. 3.2 presents a pie chart showing the percentage distribution of these occurrences. Notably, the 'NONE' class frames cover nearly a quarter of the entire dataset. This predominance is attributed to the significantly higher number of poses to discriminate as absence of movement compared to those showcasing specific skills.

The dataset was randomly splitted, with 80% of the videos allocated to the training split and the remaining 20% to the test split.

Chapter 4

Proposed Method

In this chapter we delve into the methods used to solve the classification task. First, we create a reference dataset to see how interpolations perform. The results will lead us to study the behaviors of the distribution of frames obtained. To solve classification task, a Multilayer Perceptron is then designed, trained and tested with several interpolated datasets.

4.1 Interpolations Performance on Reference Data

In this section we discuss the method used to enstablish a performance rank between the various interpolations.

4.1.1 Ground Truth Dataset Creation

The various interpolations discussed in Chapter 2 have been tested on a reduced-size dataset, that contains ground truth values. The initial dataset, with missing values, comprises 78 columns that represent the positions of 25 joints of the human body. For each joint, the dataset records the X coordinate, Y coordinate, and the associated confidence value. Additionally, the dataset includes three columns dedicated to video name, frame number, and skill id, providing essential information for the classification phase. This dataset contains 101137 rows, with a total amount of 7585200 values (last three columns excluded).

From this dataset, a cut was made to obtain sequences of known values to use as reference for the estimations. For this purpose, a code has been written. It iterates all the rows of the dataset, storing attributes such as previous/current frame, previous/current video, and constructing sequences of sequential rows with all known values. All the sequences have to contain at least 5 observations and the frames have to be contiguous. The dataset obtained has 10027 rows and a total of 751950 values (last three columns excluded).

It is used as ground truth to compare all the interpolated dataset, ranking them through several evaluation metrics discussed in 2.2.

4.1.2 Missing Sequence Generation

Once the reduced dataset has been obtained, a script has been developed to augment the dataset by inserting zero sequences based on specific criteria. These sequences are applied to each joint group categorized by video name, while ensuring that the initial and final frame values remain unaltered. This condition is imposed to prevent the presence of zero values in the interpolated dataset since they are not comprised between two known points. For example, this helps avoid scenarios where there are zero values occur at the beginning or end of a sequence. The dataset contains sequence with up to 5 contiguous zeros. The total values set to zeros are: 556293, being the 73.98% of the entire dataset. The results of the accuracy reached with the various interpolations will be discussed in Chapter 5.

4.2 Multiclass Classifier

After comparing all interpolations on reference data to analyze their behavior, it is worthwhile to examine how the reconstructed data impacts a multiclass classifier.

Initially, we constructed six additional datasets, each interpolated using a different method, from the entire keypoints dataset. The common specifications of the keypoints dataset are discussed and illustrated in Section 3.2.

4.2.1 Multilayer Perceptron Architecture

From the extracted subset of the original dataset, we obtained an idea about how interpolations work on these kind of distribution of continuous data. To solve the detection task, a Multilayer Perceptron has been designed.

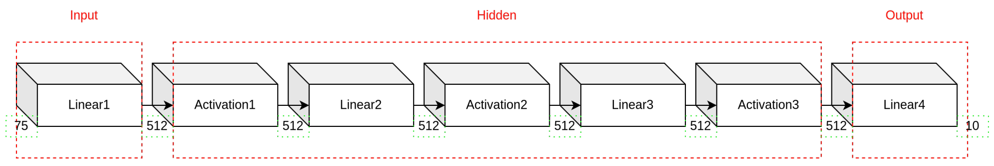

Figure 4.1: Multilayer Perceptron architecture.

As illustrated in Fig. 4.1, the input size is 75, representing the number of values extracted by OpenPose. The hidden layers are structured into three blocks, each comprising an activation layer followed by a linear layer, with 512 neurons in each layer. The network's output size is 10, encompassing the distinct skills along with the 'NONE' frames, allowing for comprehensive prediction capabilities.

For the activation function, the LeakyReLU was specifically chosen due to its ability to alleviate the vanishing gradient problem, facilitating more effective learning. The optimizer selected for training is Adam. A learning rate of 0.0001 was set to ensure stable convergence during training. The Cross-entropy loss function was employed to measure the disparity between predicted and actual distributions, optimizing the network's performance in classification tasks.

Throughout the training phase, the model was subjected to intensive optimization across 500 epochs, with each epoch iterating through the entire dataset in batches of size 512. This approach not only ensures comprehensive learning but also mitigates issues like overfitting and poor generalization. All hyperparameters, including learning rate, batch size, and network architecture, were carefully fine-tuned using cross-validation techniques with 5 folds on the original dataset containing zeros.

Chapter 5

Analyses and Results

In this chapter, we look at the analyses carried out and the subsequent results obtained. The initial focus lies on comparing the performance of interpolations within the ”toy dataset”.

5.1 Results of Interpolations Performance on Reference Data

The interpolations and the evaluation metrics considered are the ones listed in Chapter 2.

Table 5.1: Interpolation comparisons in relation with ground truth.

Method	Euclidean Distance	RMSE	KL divergence
Akima	39.879	0.0664	0.0740
IDW	22.385	0.0373	0.0022
Linear	**18.279**	**0.0304**	**0.0015**
Nearest	21.951	0.0365	0.0023
Pchip	18.453	0.0307	**0.0015**
Spline	30.899	0.0514	0.0109

Table 5.1 presents a comparison of interpolation methods in relation to their performance about the ground truth. The table evaluates the effectiveness of different interpolation techniques by measuring their Euclidean Distance, Root Mean Square Error (RMSE), and Kullback-Leibler (KL) divergence when compared to the ground truth data.

It is important to note that the Kullback-Leibler divergence is typically applied between probability distributions whose elements sum up to

1. Therefore, for its usage, a normalization step is required, wherein each element is divided by the total number of elements in the distribution.

Among these methods, Linear interpolation stands out with the lowest Euclidean Distance of 18.279, indicating its ability to closely approximate the ground truth data. Additionally, it demonstrates the lowest RMSE value of 0.0304 and the smallest KL divergence of 0.0015, further underlining, togheter with Pchip that has close values, its effectiveness in accurately representing the original data points.

Comparatively, other methods such as Akima, Nearest Neighbor, and Spline interpolation exhibit higher values across all three metrics, suggesting a relatively poorer fit to the ground truth data.

These findings offer analytical insights into the performance of different interpolation techniques, particularly in approximating joint positions. However, to further validate and comprehensively assess the performance of these interpolation techniques, we will conduct additional experiments utilizing a neural network-based approach to classify poses.

5.1.1 Reconstructed Joints Visual Inspections

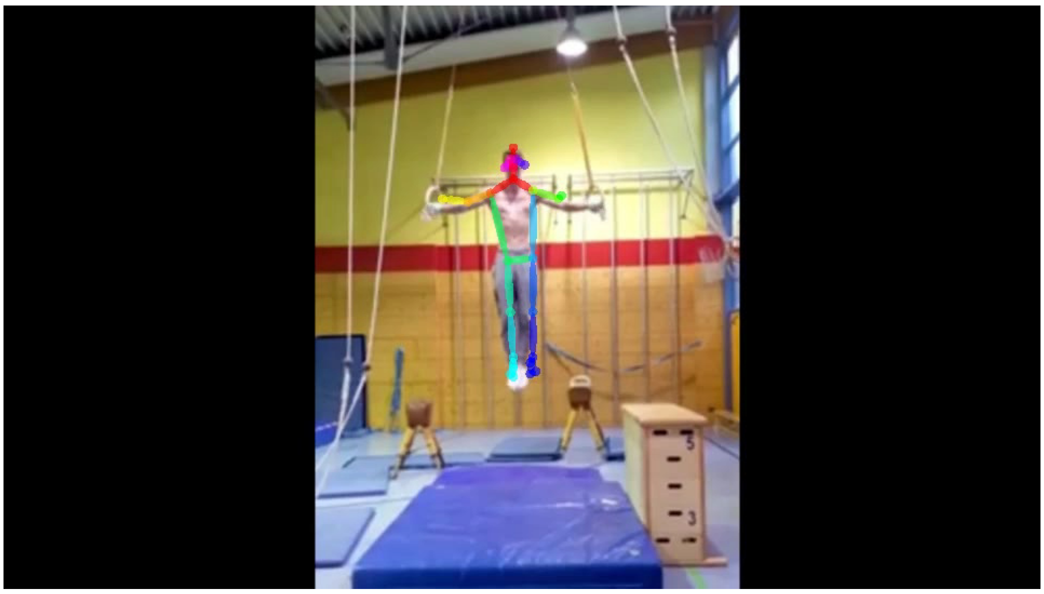

Figure 5.1: Frame 72 of 'ic36' video showing an Iron Cross.

Now, we examine a frame within the ground truth dataset where all joints are known, allowing for a visual comparison between the ground truth and the reconstructed body joints produced by different interpolation methods.

The frame considered is the 72th of the video 'ic36' where the athlete clearly performs an Iron Cross without occlusions.

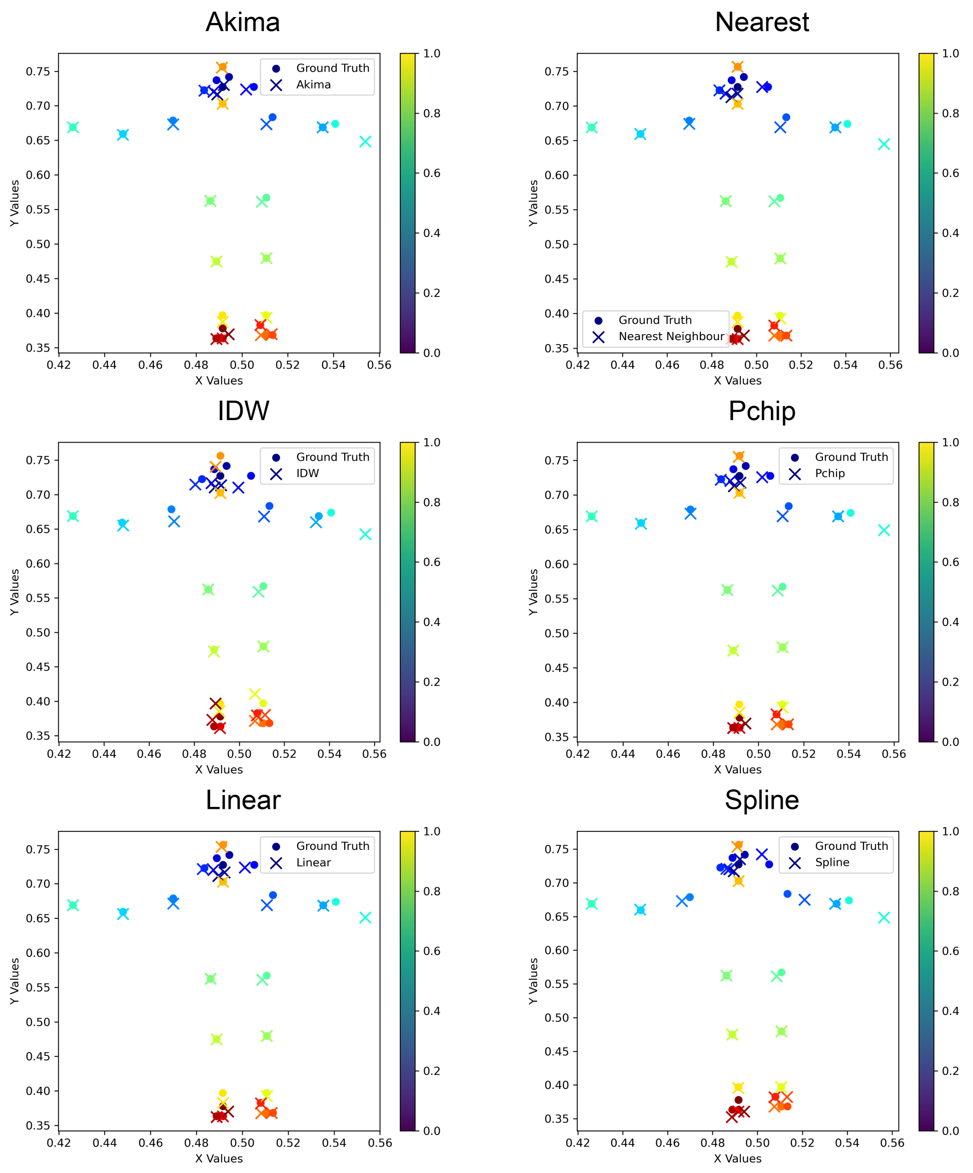

Figure 5.2: Interpolation methods visual joints reconstruction. The circles are the ground truth joints, the cross points are the interpolated ones.

Observing Figure 5.2, it becomes evident that each plot illustrates distinct variations in estimations, based on the theoretical mechanisms underlying each method.

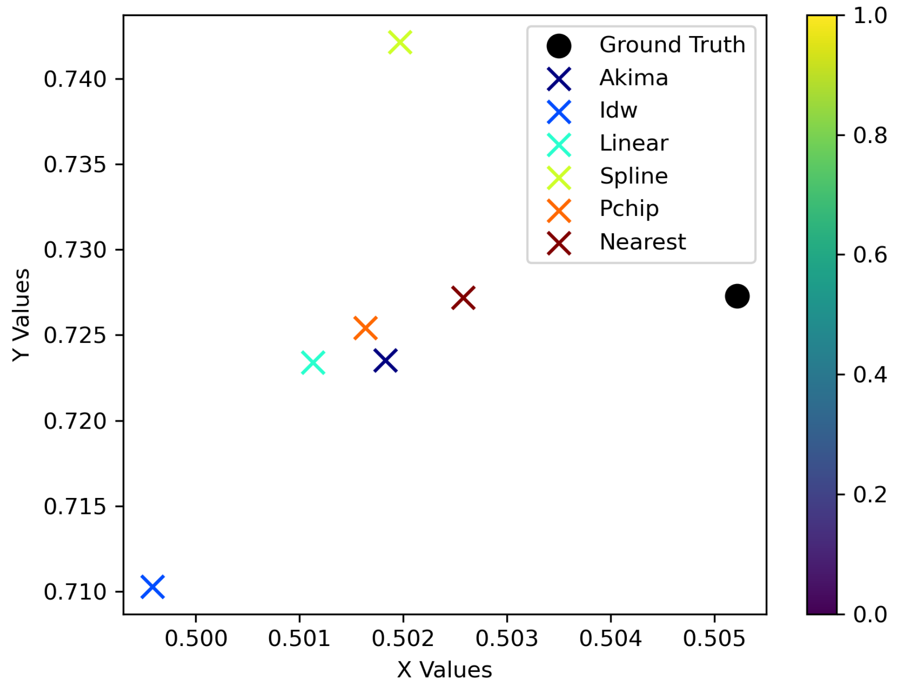

Figure 5.3: Interpolation methods visual joints reconstruction on Left Ear X and Y coordinates.

For a detailed examination of the positional discrepancies among different interpolation methods, a closer inspection can be achieved by zooming in and comparing all interpolations collectively. Fig. 5.3 illustrates the disparity in joint positions for 'LEarX' and 'LEarY' within the same frame as depicted in Fig. 5.2.

5.1.2 Methods Distributions Trends Analysis

Now, we will consider all the dataset reconstructed with each method as a distribution. Considering the high quantity of points, we will consider only the distribution of LEarX and LEarY.

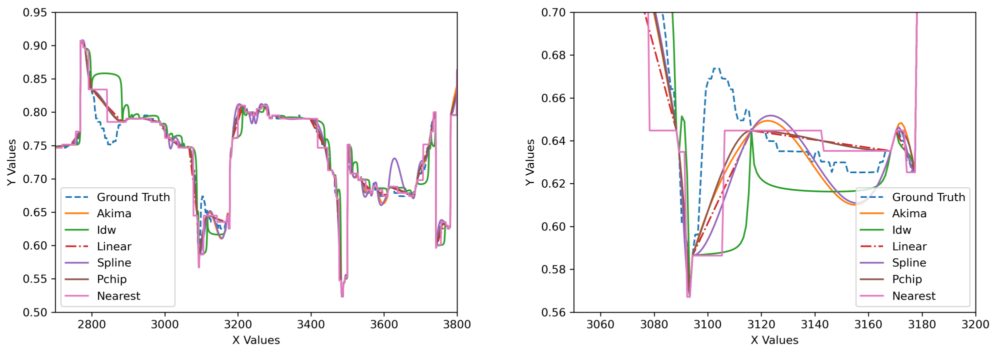

Figure 5.4: Distribution of X and Y coordinates of Left Ear. On the left, the overview of the entire distribution counting 10026 values, on the right, a portion of it.

Illustrated in Figure 5.4, the Nearest method exhibits the poorest behavior, characterized by a step-like trend, which aligns with the inherent mechanism of the method. On the other hand, Akima, Spline, and Pchip display similar behaviors with smooth curves. The Linear method approximates values without fully capturing both the upward and downward peaks. Lastly, IDW shows steep peaks followed by extended hills.

Next, we explore two additional statistical techniques for analyzing distributions: the ECDF (Empirical Cumulative Distribution Function) and KDE (Kernel Density Estimation). In both methods, the distributions are normalized to ensure their values sum to 1. This normalization is essential as these techniques operate within a probabilistic framework.

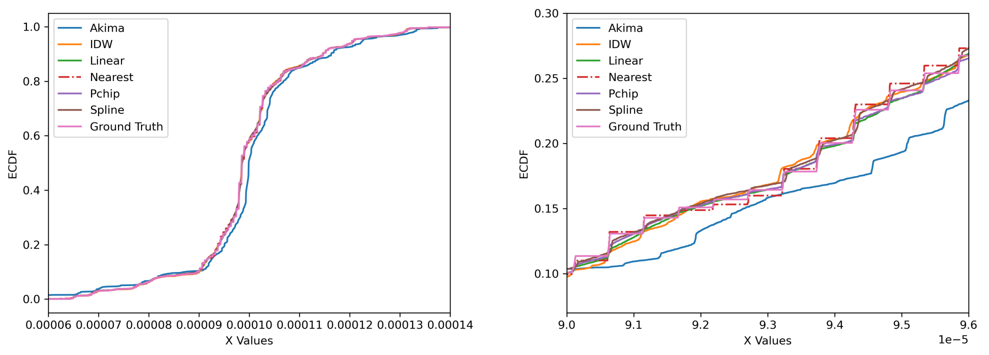

Figure 5.5: Empirical Cumulative Distribution Function of X coordinate of Left Ear. On the left, the overview of the entire distribution counting 10026 values, on the right, a portion of it.

As illustrated in Fig. 5.5, on the left side, we have a plot that represents the entire ECDF, on the right side we have a zoomed-in version of the ECDF, focusing on a specific segment ranging approximately from 0.10 to 0.30.

From the plots, it can be observed that the Akima and Pchip interpolations appear to provide smoother transitions compared to others like Nearest, which shows a step-like pattern.

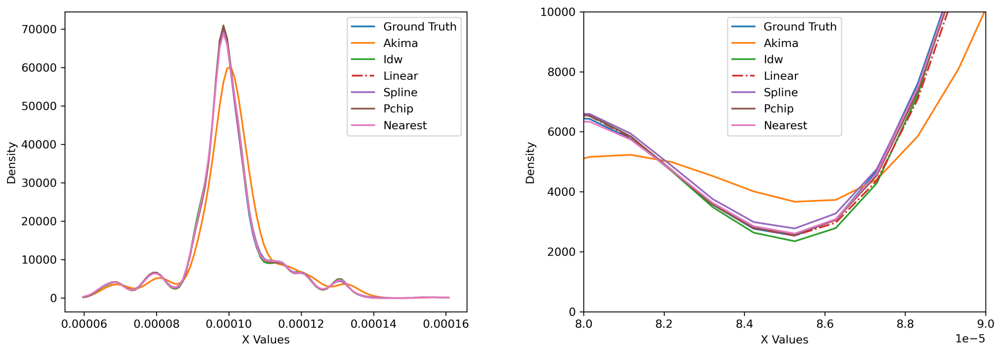

Figure 5.6: Kernel Density Estimation of X coordinate of Left Ear. On the left, the overview of the entire distribution counting 10026 values, on the right, a portion of it.

Even in Fig. 5.6 there are two plots, where the right one is a zoomed-in version of the left one. The left plot represents the entire KDE, with a sharp peak around X=0.00010 where all interpolation methods and the ground truth converge. The Ground Truth has a smooth curve while other interpolations have varying degrees of smoothness. Akima and Pchip appear to be smoother compared to Linear and Nearest interpolations which show abrupt changes. There's consistency among all methods at the peak but divergence as we move away from it.

The zoomed-in version of the KDE on the right, focuses on a specific segment near X=8.4e-5. All curves except Akima follow a similar trend but with slight variations in curvature and steepness. The Akima interpolation shows a distinct characteristic with an abrupt change indicating it might not be ideal for capturing certain transitions.

From both plots, it can be observed that the trends of interpolations vary significantly. Smoother interpolations like Akima, Pchip or Spline provide more fluid transitions. In contrast, simpler methods like Linear might not capture subtle variations effectively especially visible in the zoomed-in plot on the right.

5.2 Multilayer Perceptron Analyses and Results

In this section, we delve into the comprehensive analysis conducted to meticulously determine the optimal configuration for our neural network, alongside providing insights into the testing outcomes of the classifier.

5.2.1 Hyperparameters Tuning

To find the most effective configuration, we tuned the hyperparameters using the original dataset, ensuring no interpolations were applied. Leveraging the technique of Cross-Validation, we evaluated the performance across various hyperparameter settings. The optimizers subjected to comparison encompassed:

- Adam
- Adagrad
- RMSProp
- SGD

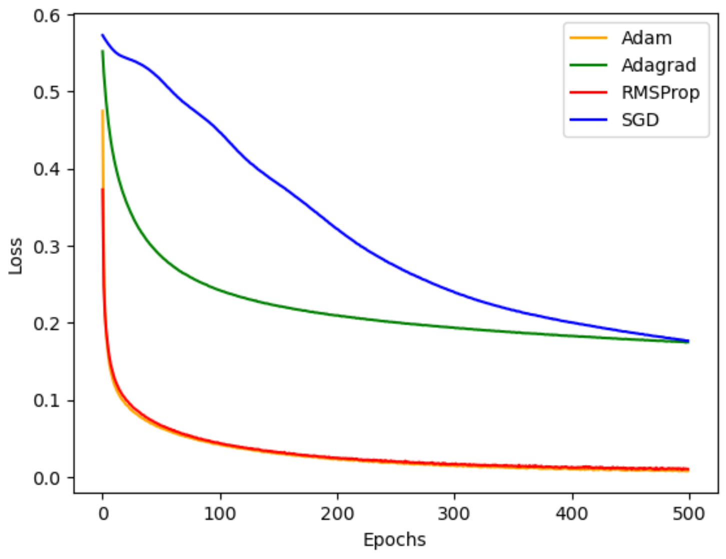

Figure 5.7: Comparison of optimizer convergence over epochs.

Each optimizer was examined to study its efficacy in helping convergence towards optimal solutions. The convergence behavior of these optimizers is illustrated in Figure 5.7.

Simultaneously, we conducted a thorough investigation into various activation functions to ascertain their impact on network performance. The activation functions under scrutiny encompassed:

- LeakyReLU
- ReLU
- Sigmoid
- Tanh
- SiLU

By validating the performance under diverse activation functions, we aimed to recognize the optimal function that encourage the information propagation and model convergence.

Table 5.2: Activation functions testing results comparison.

Activation Function	TR Loss	Test Accuracy	Recall	Precision	F1 Score
LeakyReLU	0.008	**76.17%**	0.763	**0.784**	**0.767**
ReLU	0.008	76.15%	0.762	0.782	0.765
Sigmoid	0.110	76.17%	**0.765**	0.768	0.764
Tanh	**0.007**	74.49%	0.747	0.779	0.754
SiLU	0.029	74.69%	0.747	0.777	0.752

The results of the activation function training have been obtained using Adam as optimizer.

5.2.2 Test Results

Table 5.3: Testing results of the model trained and tested with various interpolated datasets.

Methods	Recall	Precision	F1 Score	Training Loss	Accuracy
Akima	0.760	0.787	0.765	**0.007**	76.023%
IDW	0.762	**0.792**	0.766	0.009	76.201%
Linear	0.742	0.774	0.747	**0.007**	74.210%
Nearest	0.729	0.777	0.737	**0.007**	72.941%
Pchip	0.742	0.769	0.746	**0.007**	74.264%
Spline	0.723	0.756	0.730	**0.007**	72.366%
Zero	**0.763**	0.784	**0.767**	0.008	**76.388%**

Once that the architecture of the classifier has been defined, the optimal configuration has been used to train and test the model with the various datasets. The results are written in Table 5.3.

Table 5.4: F1 Score per class calculated for each Interpolated Dataset

		Interpolated Datasets						
	F1 Score	Akima	IDW	Linear	Nearest	Pchip	Spline	Zero
Skills	BL	0.823	0.795	0.816	0.804	0.825	0.758	**0.836**
	FL	0.752	0.741	0.719	0.714	0.743	0.700	**0.767**
	FLAG	0.857	0.846	0.853	0.853	0.838	0.819	**0.861**
	IC	0.913	0.922	0.912	0.910	0.926	**0.934**	0.920
	MAL	0.760	0.711	0.705	0.692	0.687	0.686	**0.776**
	NONE	0.430	0.426	0.420	0.403	0.418	0.406	**0.445**
	OAFL	0.797	**0.789**	0.756	0.752	0.733	0.743	0.788
	OAHS	0.733	**0.767**	0.731	0.709	0.711	0.709	0.689
	PL	0.813	0.798	0.778	0.769	0.772	0.752	**0.832**
	VSIT	0.886	**0.906**	0.877	0.865	0.872	0.849	0.889

The table 5.4 presents the F1 scores per class calculated for each interpolated dataset, providing insights into the performance of different interpolation methods in the context of classification tasks across various skill classes.

From both tables 5.3 and 5.4 emerges that the dataset filled with zeros (Zero) consistently demonstrates superior performance across multiple skill categories compared to datasets with interpolated values. This phenomenon may be attributed to several factors:

- Preservation of Original Information: Interpolation techniques introduce artificial values to replace missing data points, potentially altering the original distribution and characteristics of the dataset.
- Simplicity and Robustness: Zero interpolation is a straightforward and robust method that imposes minimal assumptions on the missing data distribution. Its simplicity may prevent overfitting and generalization issues that could arise from more complex interpolation techniques.

Akima interpolation stands out for its competitive performance across multiple metrics, being one of the most balanced methods, especially regarding the skills' F1 Score.

Moving on to IDW interpolation, this method demonstrates notable strengths, particularly in precision, where it outperforms other interpolation techniques. While its recall and F1 score are slightly lower compared to Akima, IDW still maintains competitive performance across various metrics. The marginally higher training loss compared to Akima suggests relatively slower convergence during model training. It has the highest F1 Score values in 'OAFL,' 'OAHS,' and 'VSIT' skills.

Linear interpolation, despite its simplicity, shows respectable performance across most metrics. However, it falls slightly behind Akima and IDW in terms of recall, precision, F1 Score, and accuracy.

Nearest neighbor interpolation, while delivering decent performance, falls short compared to Akima, IDW, and Linear interpolation methods. It has the second-lowest accuracy, and the overall performance on F1 Score per class is not satisfying.

Pchip interpolation exhibits performance similar to linear interpolation, with comparable scores across most metrics. It achieves moderate recall, precision, F1 score, and a quite lower outcome in terms of accuracy.

Lastly, Spline interpolation performs slightly below average compared to other methods, with lower scores in recall, precision, and F1 score. It has the lowest accuracy value amongst the methods. Despite this, it achieves the highest F1 Score value for the 'IC' skill.

In conclusion, the winning method, IDW, exhibits evaluation metric values closely similar to those of the original dataset without interpolated values.

Chapter 6

Conclusions

In this study, we investigated the spatial reconstruction of missing joints in the human body using various interpolation methods. The process started with the construction of a video dataset showing Calisthenics skills, from which we extracted body joint coordinates. Subsequently, we utilized the keypoints dataset as a reference and extracted a subset based on specific criteria. Our selection focused on sequences containing more than five contiguous non-zero elements within the dataset.

We then delved into evaluating the effectiveness of different interpolation techniques on these sequences, compared to ground truth values. Furthermore, we developed a Multilayer Perceptron and assessed its performance by training and testing it on different interpolated datasets.

Analysis of the results revealed a discrepancy between the performance of the interpolations compared to ground truth data and the testing results of the classifier. In the first phase, despite its simplicity, Linear Interpolation demonstrated the best results along with Pchip interpolation.

Meanwhile, during the testing phase of the multiclass classifier, the Inverse Distance Weight showed the second-best results after the dataset with no interpolations.

We provided a motivation behind these outcomes, emphasizing that the presence of missing values enhances the robustness of the network and helps prevent overfitting.

It's worth noting that this project serves as a side extension of the paper titled "Calisthenics Skills Temporal Segmentation" which is expected to be presented at VISAPP24.

The code for the main parts of the project is available at: `https://github.com/antof27/HBPE-Missing-Joints-Reconstruction`.