# Tracking body parts of Olympic swimmers Sciences 2024

Ihssane Youbi, Stefania Curila, and Matthieu Moutot

Institut National des Sciences Appliquées de Lyon, France {ihssane.youbi, stefania.curila, matthieu.moutot}@insa-lyon.fr

**Abstract.** This P-SAT is part of Sciences 2024 (https://sciences2024.polytechnique.fr), a multidisciplinary research project linking high-performance sports to science in order to increase the number of medals at the 2024 Olympic Games in Paris. Our goal is to improve an existing system that detects the position of several key body parts of swimmers extracted from swimming competition videos. In this paper, we discuss the state of the art in Human Pose Estimation and we present the strength and limitations of the existing system, as well as the solutions we explored in order to improve it.

**Keywords:** Human Pose Estimation · CNN-LSTM Neural Networks · Data Labelling · Computer Vision.

## 1 Introduction

### 1.1 Objectives

The goal of this PSAT is to discuss and explore ways to improve the work done by Jordan Cosio, a former PFE intern at LIRIS [1]. J.Cosio used videos of national and international competitions in order to detect swimmers and the position of their body parts.

The main objective is to analyze and measure the movements of the swimmers in order to give them advice on their race strategy using these videos. Each video is broken down into a sequence of images in order to identify specific key body parts for each swimmer. Each image is annotated with the position of relevant body parts: the head, the pelvis, and the hands.

This step was complicated because of the nature of the videos. The videos are low quality and taken from the bleachers. Therefore, we do not have an underwater complete vision of the swimmers. Consequently, it was necessary to identify different swimmers and select the most visible swimmers considering the point of view. Only the visible body parts were exploited, the rest of the body was ignored because it was not visible. In addition, the hands were not always visible while the swimmers crawled and the rest of the body parts were not easy to identify because of the noise created by the water splashes.

During the initial project, J.Cosio developed neural network-based models to extract the positions, postures, and movements of swimmers throughout the race.

The predicted data would then be used to reconstruct the swimmer's movements. Our continuation of J.Cosio's work was motivated by the margin of improvement in the accuracy of the final system, mainly due to the complexity of the task and the poor quality of the training data.

### 1.2 Our contribution

First, we began by searching the scientific and technical state of the art about swimming video analytics and Human Pose Estimation, a general problem in Computer Vision where the goal is to detect the position and orientation of a person or an object. We will briefly explain various methods already developed that are close to the current problem.

Secondly, we invested a lot of our time in understanding the work of the former LIRIS intern. We tried to become familiar with the different methods used for data extraction and the deep learning models developed so that we would be able to handle them and make relevant changes to improve them. The models developed by J.Cosio will be presented in this report: their architecture, the choice of hyperparameters, and the methods used for data processing.

Then, we explored several tracks to improve the model and added more training data to increase its robustness. We analyzed the results obtained by the models and evaluated the performance of the initial model as well as the new improvements proposed.

## 2 State of the art

We will now go into detail about the scientific approach followed along this project and the methods used by J. Cosio.

### 2.1 Scientific work

**Human Pose Estimation** is a compelling field and the basis for the development of applications for medicine, computer vision, sports, etc. It uses Machine Learning algorithms to analyze data from image sequences and predict the position of specific segments of the human body, such as the head, shoulders, arms, wrists, thighs, calves, and ankles, known as key points which describe the pose of a person. An approximate body position determined by a set of coordinates for each named segment is obtained and used for motion tracking, animation, and action recognition. The set of coordinates can be viewed as a skeleton superimposed on the original image. Algorithms work well when the human body data captured in the image sequence has clear elements, but fail when the elements are hidden, or not visible enough [2] [3].

**Neural networks for Human Pose Estimation** Neural networks are a subset of Machine Learning and are focused on deep learning algorithms. These

networks imitate the way biological neurons send signals to each other, their structure is inspired by the human brain. Neural networks use training data to learn and improve their accuracy. Once these learning algorithms have been sufficiently refined, they are useful tools for classifying data in computer science, artificial intelligence, image processing, etc.

An effective method used to solve human pose problems is to use Convolutional Neural Networks (CNN) to analyze image sequence data and predict the positions of key points on the human body. CNNs exploit the principles of linear algebra, in particular matrix multiplication, to identify patterns in an image and are generally used for image recognition, pattern recognition, and computer vision. They are trained with a set of images and corresponding annotations of specific segments of the human body (location of joints and upper and lower limbs). They can work with complex data if they receive enough training data, and learn to recognize patterns in the image sequence corresponding to key points.

CNNs have made significant progress in action recognition and in Human Pose Estimation from image sequences. Actual methods for Human Pose Estimation are based on the use of CNN [4]. The initiators were Alexander Toshev and Christian Szegedy, who proposed to use Deep Learning to identify joints in the human body. The estimation of their position is treated as a regression problem based on CNN [5]. Yeonho Kim, Daijin Kim use CNN to estimate human pose from the projection of the depth and ridge data in a distance transform map [6]. The proposed projection reduces the tri-dimensional information loss, the ridge data avoids joint drift and the CNN increases localization accuracy. Manisha Patel and Nilesh Kalani compare in [7] various deep learning models for pose estimation of a single person and multiple persons.

Algorithms at a higher level involve multi-stage architectures with multiple CNNs to analyze data at different scales and levels of detail. Affinity fields are used to model relationships between specific segments of the human body and improve the accuracy of predictions or position estimation of these key points. They are designed to handle the large variations in scale and position that are common in swimming videos.

**Open Pose** OpenPose is a human position detection library that uses the CNN as its main architecture. As mentioned in the Human Pose Estimation part, a skeleton is a set of connected key points that can describe the orientation of a person. Early algorithms estimated the position of a single person, whereas OpenPose offers the possibility to estimate the position of several swimmers in the same images in the videos we work on. Each relevant connection between two extracted keypoints is mapped to form a pair, but not all combinations of extracted keypoints result in relevant pairs.

First, OpenPose extracts features using the first layers of the network. The extracted features are inserted into two parallel divisions of convolutional network layers. The first predicts a set of confidence maps while the second predicts

a Part Affinity Field (PAF) that creates a degree of association between the parts.

Later steps are used to correct the predictions made by the branches. Using confidence maps, pairwise graphs are made. PAF discards weaker links from these graphs. The predictions are concatenated to form a skeleton based on the number of people analyzed. Several stages of CNNs are used to refine the prediction.

Zhe Cao et al. propose a non-parametric representation as a PAF to learn to pair body parts with people in the image, and prove that a refinement of just the PAF, instead of a refinement of the PAF and body part location, leads to increased accuracy [8]. Pin-Ling Liu and Chien-Chi Chang propose combining image data and depth data to find the locations of body key points. Also, using RGB-D cameras and OpenPose provides a motion tracking method to identify key points locations [9]. Sarah Mroz et al. have done a comparative study between a new pose estimation model that can run on a smartphone and the OpenPose model to see if they offer viable body key points for virtual motion evaluation [10]. With OpenPose as a reference, they calculated the Pearson correlation and root mean square error between the key point trajectories of both solutions. It turned out that BlazePose is not viable at this time for relevant evaluations because it had several cases where key points deviated from the anatomical centers of the joints compared to what happened with the classical solution.

**Deep Learning CNN-LSTM** The CNN-LSTM architecture involves using Convolutional Neural Network (CNN) layers for feature extraction on input data combined with Long Short-Term Memory layers to support sequence prediction with spatial inputs, like images or videos.

A LSTM (Long Short-Term Memory) is a type of Recurrent Neural Network (RNN) architecture that is able to capture long-term dependencies in sequential data. LSTMs are designed to handle the problem of exploding gradients and vanishing gradients that can occur in traditional RNNs. They do this by introducing a memory cell, gates (input, forget and output gates) that control the flow of information into and out of the cell, and a hidden state that is passed from one-time step to the next. This allows LSTMs to selectively retain or discard information as needed, and to use the information stored in the memory cell to make predictions about future time steps. LSTMs have been used in a wide range of applications including natural language processing, speech recognition, and time series forecasting.

**Raccourcir cette partie** Abubakr H. Ombabi et al. propose a novel deep learning model for Arabic language sentiment analysis based on one-layer CNN architecture for local feature extraction, and two layers LSTM to maintain long-term dependencies [11]. The feature maps learned by CNN and LSTM are passed to a SVM classifier to generate the final classification. Mohamad Shahbazi and Hamid Aghajan propose a deep learning-based model for prediction of epileptic seizures using multichannel EEG signals [12]. Multichannel images are first

constructed by applying short-time Fourier transform (STFT) to EEG signals. A CNN-LSTM neural network is trained on the STFTs in order to capture the spectral, spatial, and temporal features within and between the EEG segments and classify them.

However, a CNN-LSTM architecture has not been used for analyzing swimming videos yet. J. Cosio began working on the subject by proposing different models analyzing swimming videos that we will be presenting and discussing in the following section.

### 2.2 J. Cosio's work

**Data Labelling** The videos used for the project are filmed during professional swimming competitions. Most of the time, they show a view from the top of the eight swimming lanes. Two cameras were placed in the top 2 corners to capture the entire pool. The selected stroke for this project is the crawl which presents a strong cyclic aspect compared to the other swimming styles. The swimmers' arms alternate between a movement from the pelvis to the front of their head. Also, the body parts are more often out of the water than in the water.

J. Cosio tried two ways of image labeling. Initially, he considered labeling all the swimmers. However, due to the position of the camera, the first swimmers appear taller than those at the bottom because of the perspective effect. It is therefore almost impossible for a human to differentiate the different parts of the swimmers beyond the third lane. A first compromise was made by only labeling only the first three swimmers from the bottom. He used 3 labels: head, pelvis, and hand. However, these labels do not allow us to differentiate the swimmers of the same image, nor the left hand from the right hand for one exact swimmer.

After some analysis, he realized that the labels were not adapted to the problem because it is essential to dissociate the swimmers in order to apply a temporal model and to separate the hands of each swimmer since they might be both visible. Therefore, another way of labeling has been chosen.

The selected labeling method uses a framerate of 15 fps on the first 3 swimmers. Four key points were selected for each swimmer: the head, the right hand, the left hand, and the pelvis for a total of 12 different labels (4 labels per swimmer, 3 swimmers per video). He labeled 4 videos using this method. The splashes generated by the movement of the arms make it impossible to detect the head and the pelvis on several consecutive images. Thus, the past and future positions of the swimmer were used to identify them.

**Interpolation** The goal of the Interpolation is to generate a temporal model that addresses the phenomenon of cyclic occultation of swimmers' arms. The out-of-water hand is labeled so that its position is known, while the underwater hand is unlabeled and an arbitrary state identified by a zero is assumed.

A model function for the arms, called a discontinuous sine, was created by J. Cosio from a sine whose negative values were set to zero. Since even professional athletes are not so precise as to reproduce the same motion every cycle, random variability was induced to make each parabola unique. A limited number

of points have been generated from this function, which is assumed to represent what is happening in reality. The points representing the labels applied to a human have an induced variability that corresponds to reality, using a normal distribution. The variability of the points is higher when the function is set to zero to simulate the uncertainty associated with the position of a hand underwater.

The model is implemented by a simple deep network architecture created in Pytorch. First, a recurrent LSTM cell is created and trained. Then, the same procedure is used, but with a recurrent neural network (RNN). The results show that even with little data and simple architecture, it is possible to accurately predict swimmer hand positions using only the past. It should be noted that for non-real-time video recordings, future swimmer positions could also be exploited.

**Data Augmentation** Due to the complexity of the problem, J. Cosio decided to automatically label the unlabeled images using regressions. He used a hashing method to associate the labeled images with the entire set of images. To perform regression on the hands, it is necessary to isolate the pulse of each hand only when the hand is out of the water (coordinates are zero if the hand is in the water). The discontinuous sinus function (done through interpolation) helps isolate the position of the hands when the regression will be performed.

**Model 1 - Transfer learning using Inception v3** Efficient use of transfer learning involves pre-training a ConvNet on a very large dataset (ImageNet) and using it either as initialization or as a fixed feature extractor for the task of interest. Inception v3, based on the contribution of Christian Szegedy [13], is an image recognition model that has an accuracy of over 78.1% on the ImageNet dataset.

The method performs transfer learning on the Inception v3 model, locking the weights of all model layers except the last 4 fully connected layers. It takes over the ConvNet pre-instruction so that the filters learned by the model are as accurate as possible. J. Cosio uses a network whose last 4 layers are trained with mini-images containing swimmers extracted using Nicolas' model. A final layer containing eight outputs to predict the x,y coordinates of the four key points (head, right hand and left hand, pelvis) is added. The loss function chosen is the Mean Square Error (MSE) because in the case of a multi-point regression in a 2D frame, it seems to be relevant. The optimizer used is Adam.

The images are of different sizes, they are resized to 200x100 pixels and then normalized. A train-test method is performed with 85% of the data dedicated to the training of the model. This is a high proportion, but not much data is available. A training step of $10^{-5}$ is chosen as well as a batch size of 16 for 100 epochs.

**Model 2 - CNN** To get much better filters than the pre-trained models, J. Cosio decided to implement a simpler CNN because the results of the previous model were not as good as expected.

This model contains two convolution sequences followed by Max Pooling. Twenty filters are extracted from the first layer and fifty from the second layer. Then the output filters are flattened and bound to two layers of fully connected neurons. The last layer has 8 output neurons corresponding to the predicted coordinate values. A ReLu function is used between each layer in order to add non-linearity to the model.

The images are resized to 200x100, there is no normalization, and the batch size is 32 images for 601 images in the training set and 151 images in the test set. The learning step is set to $10^{-5}$.

**Model 3 - CNN-LSTM** The third model adds the dimension of temporality. The complexity of creating the model lies in the interconnection between CNN and LSTM. A 2D convolution layer under Pytorch considers a four-dimensional input and keeps the same format on the output. A fully connected layer is placed between the convolution and temporal layers. The LSTM, on the other hand, accepts a three-dimensional input. However, we can't reconstruct the sequence because we have lost the information during the pass in the convolution layers. To avoid this problem, J. Cosio designed data with dimension 5 and through iterations and stacking into a new list generates data in a three-dimensional format that will be accepted by the temporal layers.

85% of the data is used for training, the learning step varies from $10^{-4}$ to $10^{-5}$ and the batch size is 4.

### 2.3 Strength and limitations of the existing system

To quantitatively assess the performance of the models, we use the MSE metric that evaluates the distance between the actual label coordinates of a validation sequence and the predicted coordinates. For the three models used with augmented data, the following table is obtained:

**Table 1.** Models performance with MSE metric

| **Model** | Inception | CNN | CNN-LSTM |
|---|---|---|---|
| **MSE** ($10^{-2}$) | 4.7 | 3.4 | 0.72 |

The first aspect that stands out is the difference in scale between the scores of the Inception, CNN, and CNN-LSTM models. Inception is the worst performer which is certainly related to the fact that the pre-trained filters are not sufficiently adapted to this problem. The CNN performance compared to the Inception suggests that a simple model adapted to the problem proves to be more efficient than a partly pre-trained model that is not well adapted to it. However, we can see that the CNN-LSTM performs almost five times better than the CNN: the addition of two complex layers of the LSTM takes into account the time dependence and allows such improvements.

On the qualitative side, we can see in the following extract of a validation sequence that the CNN-LSTM does a decent job at identifying the body parts of the swimmers, except for the hands:

**Fig. 1.** Extract of a validation sequence with CNN-LSTM predictions

Overall, the model managed to detect the head and pelvis correctly even though it was inaccurate in some cases because of the noise. A quick analysis of the prediction samples of the other models shows that most models are able to consistently detect the head (green) and pelvis (purple) of swimmers. This is due to the presence of twice as much data for the pelvis and head than for the hands, which are underwater half the time. Inception and CNN do not take the time aspect into account and this can be seen in the inconsistency of the hand prediction. Temporal models perform much better considering the swimming context.

Most of the time, one of the two arms is underwater, not visible, and therefore not labeled. The solution used until then was to set default coordinates of (0,0) to the hands that are not visible. The problem during the predictions was that the gradient is back propagated on these coordinates even though they are useless. Moreover, the transition from normal coordinates to null coordinates in a sequence of images is often abrupt and thus has a significant impact on the loss. This explains why we can see some labels at (0,0) in Fig.1.

To counteract this, J. Cosio removed the zero coordinates present in the training set and the corresponding coordinates in the model prediction. Thus, the null labels are not taken into account during the backpropagation and the calculation of the loss function. This prevents unnecessary noise and greatly improves predictions and convergence speed.

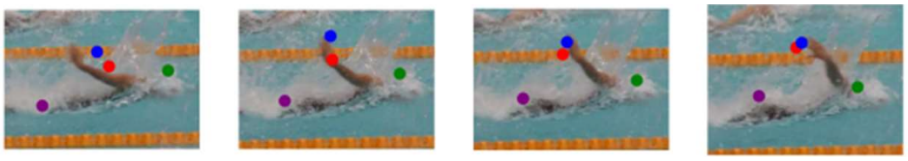

**Fig. 2.** Extract of a validation sequence with CNN-LSTM predictions when (0,0) labels are ignored

However, as we can see in Fig.2, the two points corresponding to the right and left hands are often superimposed on one single hand. This comes from the fact the hands are rarely both visible, but the model can't have a variable number of outputs according to what is observed and predictable. To solve this new problem, J.Cosio added a visibility variable to each one of the input labels and created a new 16-output model called CNN-LSTM-is-visible, whose results are displayed in the following Fig.3:

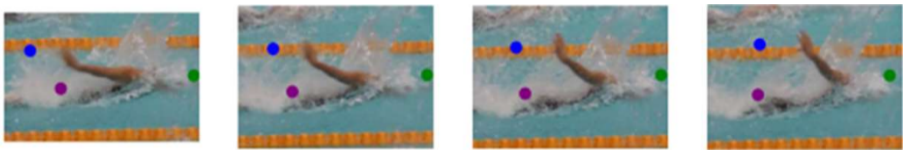

**Fig. 3.** Extract of a validation sequence with CNN-LSTM-is-visible

When the label visibility is taken into account, the model overcomes the superimposition of the two labels on a single hand, but points tend to be declared "visible" too early and too late compared to reality, causing false predictions. Moreover, the MSE scores are a little bit worse, with an MSE of 0.019. This is because when the label visibility is taken into account, the model can no longer "cheat" by bringing two points on what it considers to be a hand in the case of CNN-LSTM.

To validate the best performing model, the recording of a swimmer in real condition was used here: https://ibb.co/kML218J. The employed model in the video is CNN-LSTM with visibility and augmented data. Predictions are by no means consistent and, while accurate on some images, they are not accurate over time. The accuracy is increased for the head and pelvis but remains unusable in practice. Since the frame rate is the first feature that can cause instability in predictions, an attempt was made to use only labeled data. The predictions are not improved, they show chaotic and inaccurate behavior. It can be deduced that the chosen video frequency is the first parameter to be questioned. The model is trained on a certain fps value and when it is tested on another video of higher quality the results are not as accurate. This limitation is due to the low quantity of data used to train the model.

## 3 Model improvement attempts

Once we understood how J. Cosio's models were working and became more familiar with their strengths and drawbacks, we decided to try three different ways to improve the solution. We focused our efforts on improving the CNN-LSTM models as they were the most efficient. First, we analyzed how the performance of the regular CNN-LSTM was impacted by different hyperparameter values. Then, we modified the architecture of the CNN-LSTM in different ways, and

developed a new version that addresses the matter of hand visibility. Finally, we attempted to label another video so that the models would have more training data.

Our objective is to improve the performance of the CNN-LSTM which is a MSE on the validation set of 0.00722 (with augmented data) and 0.0111 (without augmented data).

### 3.1 Impact of hyperparameters on performance

A hyperparameter in a neural network is a variable that defines the architecture of the network and is set before the training process begins. Modifying a hyperparameter can improve the performance of the model since it can adjust the architecture of the model to better fit the data. By changing the hyperparameters, the model can find the optimal values for the weights and biases in the network, which can lead to improved accuracy and performance. Additionally, hyperparameter tuning can also help reduce overfitting, allowing the model to generalize better and produce more accurate results.

**Epochs** First, we decided to check how the number of epochs was impacting the performance. An epoch is a single iteration through all of the training data. During an epoch, the network tunes the weights and biases of the model in order to minimize the loss function. The number of epochs determines how many times the model will go through the training data, which can have an effect on the accuracy of the model. Analyzing the impact of the number of epochs on performance enables us to find the best trade-off between the accuracy of the prediction and the training time / computational resources needed.

As we can see in the following Fig.4, the validation MSE reached a minimum of 0.0115 for 75 epochs, but the performance barely improves after 25 epochs with an MSE of 0.01. Thus, we can test the next hyperparameter variations with our model trained on 25 epochs as it is a good trade-off between performance and shorter training time.

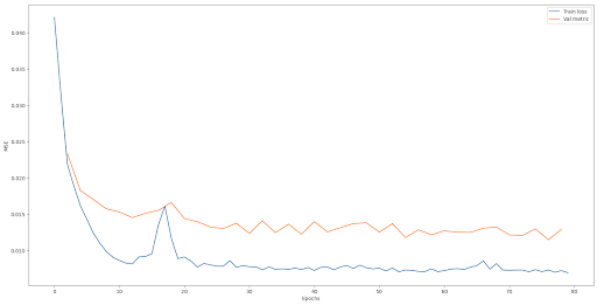

**Fig. 4.** Evolution of the CNN-LSTM loss as a function of the number of epochs

**Learning Rate** Then, we tried several values for the learning rate. In a neural network, the learning rate is a hyperparameter that determines the step size of the gradient descent optimization algorithm. It determines how quickly the weights and biases of the model can be adjusted in order to minimize the loss function. If the learning rate is too high, the model may overshoot the optimal weights and biases, while if it is too low, the model may take too long to converge.

J. Cosio initially trained his model with a learning rate of 0.0001, so we tried different orders of magnitude to see if we can get any improvement. For each learning rate value, we gathered the best MSE validation score out of the 25 epochs in Fig.5:

**Table 2.** Best validation MSE score obtained for different learning rate values

| Learning rate value | Best Validation MSE score |
|---|---|
| $10^{-5}$ | 0.017 |
| $5 * 10^{-5}$ | 0.015 |
| $10^{-4}$ **(default)** | 0.011 |
| $5 * 10^{-4}$ | 0.062 |
| $10^{-3}$ | 0.026 |
| $5 * 10^{-3}$ | 0.0152 |
| 0.01 | 0.171 |

As we can see, the initial learning rate value of $10^{-4}$ selected by J. Cosio is already the one leading to the best performance.

**SGD Optimizer** We also wanted to see if using the SGD optimizer instead of the Adam optimizer had any positive effect on performance. The Stochastic Gradient Descent (SGD) optimizer is a type of optimization algorithm used in neural networks. It is an iterative method that adjusts the weights and biases of the model based on the gradient of the loss function with respect to the weights and biases. The SGD optimizer is used to minimize the loss function and can be used with a variety of different learning rates and momentum values. The Adam optimizer is a variant of the SGD that is designed to update weights and biases more efficiently than other algorithms and utilizes adaptive learning rates, which helps the model converge faster and more accurately.

The performance of the model using the SGD optimizer linearly decreases as the number of epochs increases, but it is considerably worse than the model using Adam's optimizer (see Fig. 7 in Appendix). The validation MSE score decreases very slowly and only reaches 0.285 after 100 epochs.

**Sequence Size** Finally, we repeated the process with the length of the sequences. A sequence is a series of input values and images that are fed into the model in order to predict a target output. It is used to train the model and is composed of multiple time steps that contain different features. The model

will use the sequence to learn patterns in the data. The initial CNN-LSTM was using sequences of 40 images, so we ran it with sequence sizes between 10 and 90 to assess the impact of the sequence size on the performance. We gathered the results in Tab.3:

**Table 3.** Best validation MSE scores obtained for different sequence sizes

| Sequence Size | Run 1 - Best Validation MSE score | Run 2 | Run 3 |
|---|---|---|---|
| 10 | 0.0066 | 0.0072 | 0.0061 |
| 20 | 0.0129 | 0.0063 | 0.0072 |
| 30 | 0.0282 | 0.0092 | 0.0097 |
| **40 (default)** | **0.0882** | **0.0087** | **0.0072** |
| 50 | 0.0079 | 0.0429 | 0.0077 |
| 60 | 0.0085 | 0.0089 | 0.0083 |
| 70 | 0.0119 | 0.0379 | 0.0681 |
| 80 | 0.0103 | 0.0254 | 0.0122 |
| 90 | 0.0170 | 0.0506 | 0.015 |

After several runs, the results turned out to be quite volatile which does not enable us to draw any definitive conclusion regarding the impact of the sequence size on the performance. By looking at the general trend of the results, it seems that the length of 40 images initially selected by J. Cosio was optimal but that smaller sequences tend to perform well too.

### 3.2 Different CNN-LSTM architectures

The regular CNN-LSTM architecture consists of the following:

- 2 convolution layers
- 1 fully connected hidden layer
- 2 LSTM layers
- 1 fully connected layer to the output classes

We attempted to make changes to this base architecture to see if it leads to better results.

The first change was to add an extra convolution layer after the two first ones, hoping that this would help the model in detecting local features of the input images. Below is the evolution of the performance of this model for 25 epochs.

As we can see, we managed to obtain slightly better results, with a validation MSE score of 0.0065 for 22 epochs.

The second change was to add another fully connected layer. This would add complexity to our model which could be beneficial or not. Unfortunately, this extra layer does not enable the model to perform better, the best validation MSE that it could achieve was 0.024.

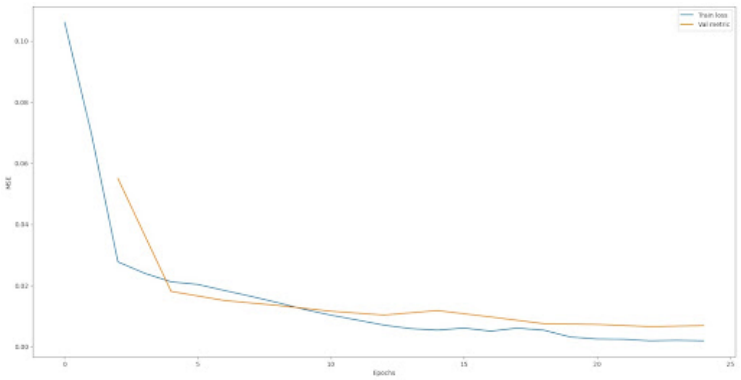

**Fig. 5.** Loss vs. 25 epochs for a CNN-LSTM with an extra convolution layer

The last modification was to add a third LSTM layer that would potentially enable the model to better capture the long-term dependencies in the sequences of images. Here again, the performance worsens and never gets below 0.05.

### 3.3 Three-label CNN-LSTM

To tackle the hand visibility problem discussed in part 2.5, we implemented another version of the CNN-LSTM that only takes three input labels and produces 3 outputs: head, pelvis and hand. By losing the distinction between the right and left hand, we could make a simpler model that leverages the fact that most of the time, only one hand is visible. In order to achieve this, we went through the files containing the labels for each video and identified three possible cases:

- only one of the hands of the swimmer was not visible
- both hands of the swimmer were not visible
- both hands were visible (only a few instances)

For the first case, we directly removed the label corresponding to the non-visible hand. For the two last cases, we decided to automatically remove the left hand of the swimmer. Indeed, all the videos were recorded when swimmers went from the left side of the pool to the right side of the pool. Consequently, the right side of the swimmers' bodies was more visible than their left side. Their right hands are more likely to be visible than their left hands since the body and the noise from the water would hide it more often.

Then, we modified the initial CNN-LSTM code so that it would take 3 labels per swimmer as an input and produce 3 output labels, and ran it with the same hyperparameters. We obtained the following results:

The 'numerical' performance has improved compared to the initial CNN-LSTM with a validation MSE score of 0.0036 at its lowest for 66 epochs. However, we look at the prediction made on some validation sequences, we can see that our 3 outputs model produces disappointing results. Indeed, as we can see on

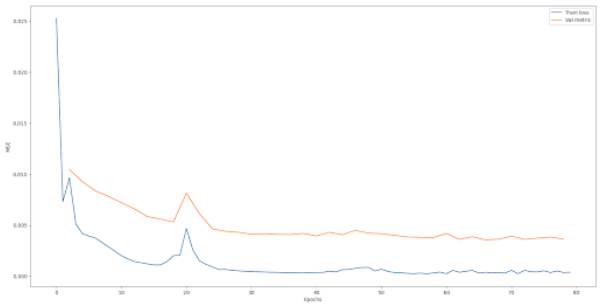

**Fig. 6.** Loss vs. 25 epochs for the three-label CNN-LSTM

the extract of a validation sequence below, our model is lacking accuracy when it comes to detecting the exact position of the visible hand.

Then, a second problem arises when one hand is becoming invisible and the other hand is reappearing. Indeed, the coordinates of the hand vary abruptly and our model fails at understanding that the visible hand would now be the one close to the pelvis and not the one close to the head anymore (cf- Fig.8, 9 in Appendix).

### 3.4 Addition of new labeled data

The relevance of the results depends not only on the model but also on the quality and quantity of data it is trained with. In order to increase the amount of training data of the model it was necessary to exploit a new video. Before embarking on video labeling, we explored several ways to optimize the data extraction from the video. We chose a video that resembles the ones used before with the same perspective and positioning of the camera. The video is 18 seconds long and has been labeled in 15 fps, using the most efficient labeling model explained in the previous sections. We obtained a total of 3000 labels.

Before giving the data to the model, several steps are followed: the video is reduced to multiple frames, labeled, augmented by interpolation then images for each swimmer are cropped.

The choice of labels was relevant and didn't need any modification. As in the previous videos, only the three first swimmers were clearly visible. We placed 4 labels per swimmer, by encompassing each body part with a rectangular frame. We took care to always position the head and pelvis labels even if they were not visible based on the previous and following pictures. As for the hands, we decided to position only one hand at a time. It was necessary to distinguish the three swimmers with the labels in order to proceed to the interpolation in the next steps. The labeling was done with a 15 fps and the interpolation allowed a data augmentation: we went from 15 fps labeled images manually to 25 fps labeled images where the labels were positioned automatically. Once all the labels are positioned, each swimmer is extracted individually in a small image. And this

collection of images is given as an input to the model. A way to improve the data treatment is to find the best way to generate images of a unique size. Considering the perspective, the size of the swimmer depends on the position of their lane. While cropping the images, we have to take into consideration the scale of the swimmer while resizing the images. The scaling problem is also present for the images of the same swimmer. Due to the perspective of the camera and its motion as well as the positioning of the swimmer, the cropping of the swimmer can create images of very different sizes. The scale must be respected in order for the key points to have a smooth position from one frame to another over time. This scaling will then be used for a recomputation of the relative coordinates of the labels for each image. This factor varies a lot from one video to another and a simple scaling rule that can be applied to all videos could not be determined. A further problem can be created when the images are resized: the body of the swimmers could be distorted and the positioning is therefore incorrect.

Unfortunately, after all the data processing, we didn't manage the integration of the new video to J. Cosio's models. We dealt with several technical challenges in the interpolation step and couldn't go further in the remaining time. The training with this new data could have led to interesting improvements.

## 4 Conclusion

To summarize, we presented in this paper the state of the art of Human Pose Estimation, the solution developed by J.Cosio, and the attempts we made at improving it. The biggest challenge we had to deal with was becoming familiar with the existing solution. We first had to understand the theory behind his vision, choices and methods. Then, we worked on the source code of his solution, so we had to analyze its structure and understand how it works in order to manipulate the data processing scripts and models. Once the previous work was understood, we managed to modify the existing models with the hyperparameters and to integrate the new Three-labels CNN-LSTM model. We ran tests to improve the quality of the data given to the model and labeled a new video but we did not manage to integrate it into the existing models within our time constraint, mostly due to the complexity and length of the data processing steps. In theory, the different models developed are adapted to our use case. Because of the bad quality and the low quantity of data, the models have probably not delivered their best performance. Further work on the data would most likely lead to better results.

To further improve the solution, other leads are worth considering. For instance, it might be interesting to develop an autoencoder, a type of unsupervised Neural Network used to learn efficient data codings. An autoencoder would enable the model to learn more compact representations of data for dimensionality reduction. It is based on a two-stage process: an encoder that compresses the data into lower dimensional code, and a decoder that reconstructs the data from the code.

Moreover, to complete our Three-labels CNN-LSTM, it would be beneficial to add a classifier that would define whether the only hand labels correspond to the left hand or the right hand of the swimmer. This could help solve the issue that arises when one hand goes underwater and the other becomes visible.

## References

1. Laboratoire d'InfoRmatique en Image et Systèmes d'information, https://liris.cnrs.fr/
2. T.L. Munea, Y.Z. Jembre, H.T. Weldegebriel, L. Chen, C. Huang, C. Yang: "The Progress of Human Pose Estimation: A Survey and Taxonomy of Models Applied in 2D Human Pose Estimation". In: IEEE Access, Volume: 8, pp. 133330 – 133348, 20 July 2020. https://doi.org/10.1109/ACCESS.2020.3010248
3. N. Sarafianos, B. Boteanu, B. Ionescu, I.A. Kakadiaris: "3D Human pose estimation: A review of the literature and analysis of covariates". In: Computer Vision and Image Understanding, Volume 152, November 2016, Pages 1-20. https://doi.org/10.1016/j.cviu.2016.09.002
4. A. Singh, S. Agarwal, P. Nagrath, A. Saxena, N. Thakur: "Human Pose Estimation Using Convolutional Neural Networks". In: 2019 Amity International Conference on Artificial Intelligence (AICAI), 4-6 Feb. 2019. https://doi.org/10.1109/AICAI45948.2019
5. A. Toshev, C. Szegedy: "DeepPose: Human Pose Estimation via Deep Neural Networks," Dec. 2013. https://doi.org/10.1109/CVPR.2014.214
6. Y. Kim, D. Kim: "A CNN-based 3D human pose estimation based on projection of depth and ridge data". In: Pattern Recognition, Publisher: Elsevier, October 2020
7. M. Patel, N. Kalani: "A survey on Pose Estimation using Deep Convolutional Neural Networks". In: 2nd International Conference on Machine Learning, Security and Cloud Computing (ICMLSC 2020), IOP Conference Series: Materials Science and Engineering, Volume 1042, 18th-19th December 2020, Hyderabad, India. https://doi.org/10.1088/1757-899X/1042/1/012008
8. Z. Cao, G. Hidalgo, T. Simon, S. Wei, Y. Sheikh: "OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields". In: IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, arXiv:1812.08008v2 [cs.CV] 30 May 2019
9. P. Liu and C. Chang: "Simple method integrating OpenPose and RGB-D camera for identifying 3D body landmark locations in various postures". In: International Journal of Industrial Ergonomics, Volume 91, September 2022. https://doi.org/10.1016/j.ergon.2022.103354
10. S. Mroz, N. Baddour, C. McGuirk, P. Juneau, A. Tu, K. Cheung, E. Lemaire: "Comparing the Quality of Human Pose Estimation with BlazePose or OpenPose". In: 2021 4th International Conference on Bio-Engineering for Smart Technologies (BioSMART), 08-10 December 2021. https://doi.org/10.1109/BioSMART54244.2021.9677850
11. Abubakr H. Ombabi, Wael Ouarda, Adel M. Alimi: "Deep learning CNN–LSTM framework for Arabic sentiment analysis using textual information shared in social networks". In: Soc. Netw. Anal. Min. 10, 53 (2020). https://doi.org/10.1007/s13278-020-00668-1

12. M. Shahbazi and H. Aghajan: "A generalizable model for seizure prediction based on Deep Learning using CNN-LSTM architecture". In: 2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP), 26-29 November 2018. https://doi.org/10.1109/GlobalSIP.2018.8646505
13. C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna: "Rethinking the Inception Architecture for Computer Vision," Dec. 2015. https://doi.org/10.48550/arXiv.1512.00567

## 5 Appendix

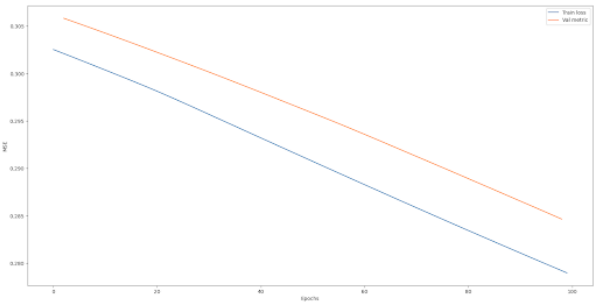

**Fig. 7.** Loss vs. 100 epochs for the CNN-LSTM with an SGD optimizer

**Fig. 8.** Extract of a validation sequence for the 3-label CNN-LSTM

**Fig. 9.** Extract of a validation sequence for the 3-label CNN-LSTM