Human Action Recognition

Classifying Human Actions using Skeleton Representations

Github Link

Guillaume Loranchet, Anthony Kobanda, Flavien Vidal
École Polytechnique
IP Paris, Palaiseau, France
{guillaume.loranchet, anthony.kobanda, flavien.vidal} @ polytechnique.edu

Referents : Dora Csillag, Pierre Perrault {dora.csillag, Pierre.PERRAULT} @ idemia.com

February 3, 2023

Abstract

The recognition of human actions is a topic in the field of computer vision, which includes various subtopics such as pedestrian detection, segmentation or classification. In order to recognise an action, a human being would first locate the person, analyse the position of the body parts, understand how they move and interact together, and finally classify the action accordingly. Through this class project, we propose to replicate this process using only body part location information. The input data is a sequence of key points coordinates, either two-dimensional or three-dimensional, of human skeleton representations, with the resulting output being the the action performed.

Index Terms *— Human Action Recognition, Skeleton, Key Points, Deep Learning, Computer Vision, LSTM.*

1. Introduction

The purpose of the project is to develop tools and methods, accompanied by meaningful metrics, to perform **human action recognition**, or **HAR**.

Over the last decade, new deep learning approaches have led to significant improvements in many areas such as computer vision, speech and language processing, and even game solving. In computer vision, in particular, progresses have been achieved in tasks such as semantic segmentation, object recognition and object detection. An explanation for such advances, besides the increasing amount of data available and improved GPUs, has been the introduction of Convolutional Neural Networks, or CNNs, allowing to capture the spatial locality assumption of the data structures and images features [1]. Consequently, the HAR task has moved from the study of manually-crafted video representations to the one of neural networks. It has become an attractive research area among the wide range of computer vision applications, and is a central task in **video understanding** which aims to understand human actions from video or image sequences, and to assign a label to each of them. This task has many real-world applications [6], such as the surveillance of public places, video captioning, sport analysis. However, the more complex the actions or applications are, the more robust and consistent the methods developed or techniques deployed need to be.

The challenges of the human action recognition task are related to the difficulty of defining the movement of the different body parts, but are also due to many real-world artifacts such as camera movements, dynamic backgrounds or recording noises. Although we could prevent some of them using high end devices or more computational power, not all of them are related to material deficiency. Different approaches have been introduced in the literature, but they are insufficient to fully cover the challenges of the topic. As an example, some of them would consider multiple points of view [5] but are unable to make up for occlusions. These studies have explored various modalities of feature representation. Indeed, the action recognition task can be studied using RGB images, infra-red data, optical streams or even **human skeleton representation**. These methods have their own advantages depending on the specific applications and, as a result, many research studies have attempted to investigate different approaches.

Action recognition based on skeleton representations

has received particular attention due to its action-focused nature and its compactness. Indeed, such structured data allow to avoid having to consider data which are much larger in size and require more processing time and optimised processes. When performing live prediction, this computational and, *a fortiori*, temporal problem must be addressed. Skeleton representations are **arrays of human joint coordinates** extracted from an image using pose estimation methods, allowing to capture essential information related to the action performed, while filtering noises, background variations or even changes in clothing.

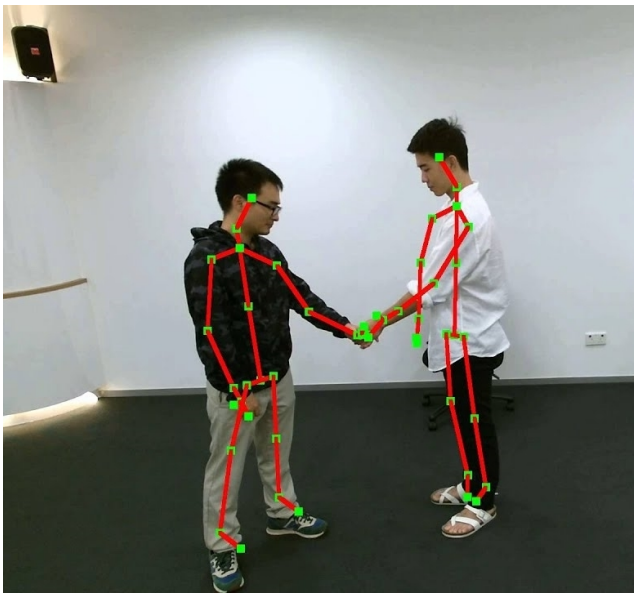


Figure 1. Human skeleton joints from an RGB image [8].

As a consequence, we decided to work on skeleton data based recognition methods, rather than directly using video data which require both more pre-processing and heavier computation. As we will see later while presentating related work, such methods have proven their worth, particularly in terms of their generalizability and resistance to photometric or scenic variability. To compare the performance of these methods, public data sets with various actions under several conditions and constraints were recorded.

In the first part, we will present an overview of recent advances in deep learning based methods for human action recognition based focusing on the one using skeleton data. In a second part, we will provide an analysis of the dataset used to build our models which will be presented in a third part. We will then analyse the results obtained from the models implemented and proceed to the conclusion.

2. Related Work

2.1. Graph Neural Networks

A graph is structure composed of vertices and connected by edges. We can distinguish two types of graphs graph networks: directed graphs, which associate a direction to the edges, and undirected graphs, which do not. Nowadays, a huge amount of information is represented through graphs such as the Google's Knowledge Graph, social media, and even simply chemical molecular structures.

Graph network structures are an emerging topic in deep learning [11], and due to the graph structure of the skeletal representations, research have been led in order to perform the action recognition task using such networks. This is the case of the Directed Graph Neural Network or DGNN [9], developed, trained and tested on the NTU RGB+D dataset [8]. In this model, the joints and bones are represented as the vertexes and the edges within a directed acyclic[1] graph, which is fed into the DGNN to extract features for action recognition. It is interesting to note that this model managed to achieve almost 95% accuracy, which is more than encouraging to think about other possible uses of GNNs.

Pierre: Say it will not be explored in the report

2.2. PoseC3D

PoseC3D present a different method for human action recognition which relies on a 3D heatmap stack [2], robust to small perturbations of the skeleton pose, and which allows to use optical flow and to take into account multiple individuals in a single frame. To develop their model, the authors of the paper aforementioned used 2D poses instead of 3D ones, as they established it helps to achieve better results. Such a method raises questions about how relevant it is to retrieve spatial information for two-dimensional data. A possible explanation to using 2D poses is their ability to carry most of the information while being simple to annotate particularly for multiperson scenes.

For research purposes we decided to test the model with the trained parameters available online. In order to do this we used a modified version of the NTU RGB+D 60 dataset to fit the 2D input of PoseC3D. In this setting, we tested 500 samples with an average duration of 22 seconds per sample. We obtained a top 1 accuracy of approximately 93%, a top 5 accuracy of 99%, and an approximate mean accuracy of 93% per class. Nevertheless, we disregard this method in our research since it uses heat-maps, which has an extended use of GPUs computation power.

[1] An directed acyclic graph is a directed graph without any loops.

3. NTU RGB+D Action Recognition Dataset

3.1. Structure of a skeleton

Here are the 25 joints of the skeletal representation:

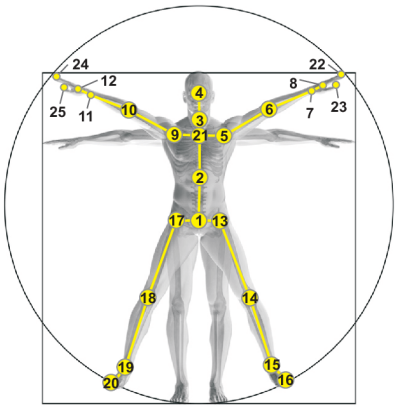


Figure 2. Configuration of 25 body joints in the dataset.

3.2. Presentation of the dataset

The dataset used in our project and presented here is **NTU RGB+D Action Recognition Dataset** [8], which counts 56,880 skeletal data and 114,480 video samples. Three Kinect V2 cameras (with a 30 frame-rate) were used so as to simultaneously capture different views from the same action. The cameras were located at the same height with however different horizontal angles: $-45°$, $0°$, $+45°$. The height and the distance of the three cameras change depending on the setup as shows the *Table* 1.

setup	h (m)	d (m)	**setup**	h (m)	d (m)
1	1.7	3.	**10**	1.8	3.0
2	1.7	2.5	**11**	1.9	3.0
3	1.4	2.5	**12**	2.0	3.0
4	1.2	3.0	**13**	2.1	3.0
5	1.2	3.0	**14**	2.2	3.0
6	0.8	3.5	**15**	2.3	3.5
7	0.5	4.5	**16**	2.7	3.5
8	1.4	3.5	**17**	2.5	3.0
9	0.8	2			

Table 1. Height and distance of the cameras in each setup.

Each file in the dataset has a *SsssCcccPpppRrrrAaaa* title:

- **sss** is the setup number (between 001 and 017).
- **ccc** is the camera ID (between 001 and 003).
- **ppp** is the performer ID (between 001 and 040).
- **rrr** is the replication number (001 or 002).
- **aaa** is the action class label (between 001 and 060.

3.3. Action classes

The dataset contains 60 action classes listed below:

A1. drink water	A31. pointing to sth with finger
A2. eat meal/snack	A32. taking a selfie
A3. brushing teeth	A33. check time (from watch)
A4. brushing hair	A34. rub two hands together
A5. drop	A35. nod head/bow
A6. pickup	A36. shake head
A7. throw	A37. wipe face
A8. sitting down	A38. salute
A9. standing up from sitting	A39. put the palms together
A10. clapping	A40. cross hands in front (say stop)
A11. reading	A41. sneeze/cough
A12. writing	A42. staggering
A13. tear up paper	A43. falling
A14. wear jacket	A44. touch head (headache)
A15. take off jacket	A45. touch chest (heart pain)
A16. wear a shoe	A46. touch back (backache)
A17. take off a shoe	A47. touch neck (neckache)
A18. wear on glasses	A48. nausea or vomiting condition
A19. take off glasses	A49. use a fan (hand/paper)
A20. put on a hat/cap	A50. punching/slapping others
A21. take off a hat/cap	A51. kicking others
A22. cheer up	A52. pushing others
A23. hand waving	A53. pat on back of others
A24. kicking sth	A54. point finger at the others
A25. reach into pocket	A55. hugging others
A26. hopping (1 foot jump)	A56. giving sth to others
A27. jump up	A57. touch others pocket
A28. phone call/answer	A58. handshaking
A29. playing with phone	A59. walking towards each other
A30. typing on a keyboard	A60. walking apart from each other

Figure 3. The actions available in the dataset.

3.4. Analysis and visualization of the dataset

In this project we only consider theses actions:

- Class 0 : Pickup (A6)
- Class 1 : Throw (A7)
- Class 2 : Sitting-down (A8)
- Class 3 : Standing-up (from sitting position) (A9)
- Class 4 : Take-off jacket (A15)
- Class 5 : Reach into pocket (A25)
- Class 6 : Pointing to something with finger (A31)
- Class 7 : Check time (from watch) (A33)
- Class 8 : Falling (A43)

Moreover, if a sequence of skeletons contains more than one skeleton per frame, we will only consider the first one.

3.4.1 Number of frames

The *Table* 3 summarises the distribution of the number of frames according to the action classe considered.

class	class size	mean	std	min	median	max
0	943	81	14	54	81	131
1	944	64	13	37	62	136
2	941	74	14	46	74	119
3	936	64	12	39	63	120
4	945	141	34	66	138	277
5	946	97	23	54	94	202
6	944	56	12	32	54	110
7	946	64	14	36	62	128
8	946	64	12	40	62	139

Table 2. Number of frames per class.

Here are a few remarks from those different values:

- Let's note that Each action class has an equivalent number of sample (1% of difference between the smaller and the bigger class).
- The mean and the median number of frames are almost equal, for each of the action class.
- Some actions have a very different average frame count (in some cases doubling). Nevertheless this is normal since some actions require more or less movements, and therefore do not have the same duration.

3.4.2 Occupation of the space

In this part we will see, for each action class, how the space is used (in the 3D dataset) and where on the screen the action are most likely to be located (in the 2D dataset).

From *Figure* 12, we can see that the actions take place in the centre of the frames. This is most likely due to the fact that the experiments centred on the camera aperture. It can be assumed that a model trained using this data could unable to analyse off-centre elements. A solution to this problem is either to manually centre the frame, or to change the space reference by considering one directly on the skeleton (for example taking 0 at the level of the pelvis).
From the histograms in *Figure* 13, we can see very similar position distribution is somewhat alike among the different action class. However, once again, the actions are centered around 0 for the X and Y coordinates, and located around a certain depth band for the Z coordinate. Similar observations as above could be made on these results.

3.4.3 Visualization

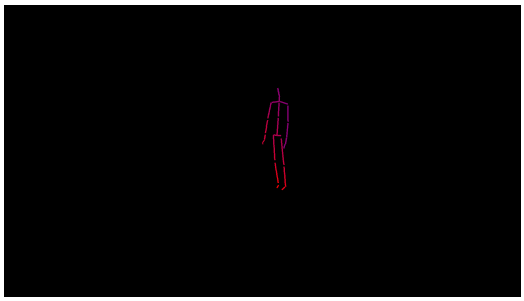


Figure 4. Animation[2] of a skeleton for the class 1 (throw).

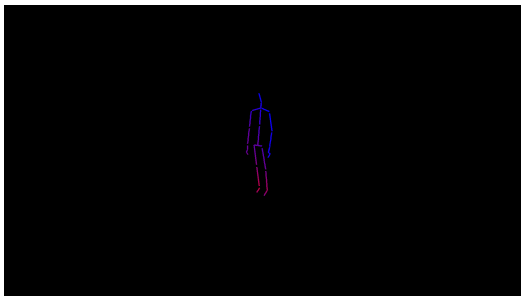


Figure 5. Animation of a skeleton for the class 2 (sit down).

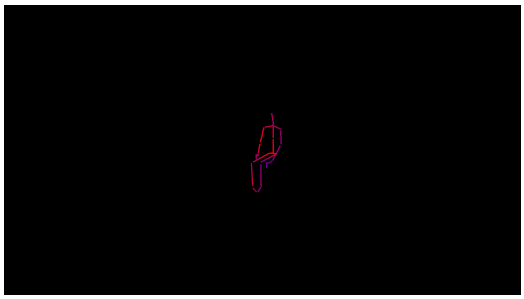


Figure 6. Animation of a skeleton for the class 3 (stand up).

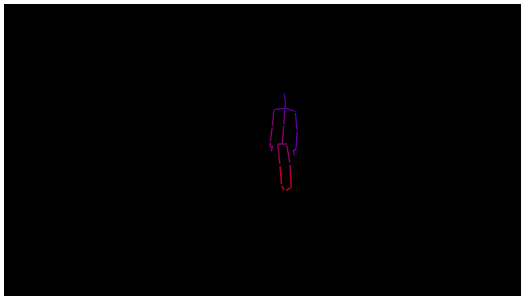


Figure 7. Animation of a skeleton for the class 7 (check time).

[2] Click on the image to watch the corresponding GIF.

4. Recurrent Neural Networks

4.1. Dealing with sequential data

Sequential data can be divided into two main categories: time series and ordered sequences. Since they inherently are the result of a temporal processes, sequences of skeleton representation are part of the former category. Given their versatility, **Recurrent Neural Networks** [10], or **RNN**s, are popular models used in a great variety of problems related to sequences.

Thus, we decided to use RNNs as a basis of our own whose architecture is described in the next section. Using standard neural network on sequence data such as skeleton data is most often not a good idea. Indeed, they are unable to take into account inputs of variable lengths, and thus to to take advantage of the sequential structure of the input data of a skeleton.

4.2. Description of RNNs

Since the input structure is a sequence of skeleton data, the inputs must be ordered. In order to transmit information through time, the architecture of a standard neural network is not appropriate because all the elements of the sequence input would be connected to all the neurons of the hidden layer. Unlike the standard neural network, in the RNN each input is sent to one of the hidden layer neurons which also takes as input the output of the previous hidden state.

Bidirectional RNNs are a variant of standard RNNs used to process sequential data with possibly interdependent time steps [7]. Their main characteristic is to rely on the use of two sub networks: one of them performing operations and sequence processing from left to right (or past to the future) and the other one from right to left (or future to the past). In other words, a prediction can result from both the information before and after the corresponding position in the input sequence. Such networks proved to be particularly useful in our problem, as the prediction at a given time may depend both on the data preceding it, but also on data following it.

4.3. Memory issues in RNNs

Most challenges in RNNs are related to memory issues. More concretely, predicting an outputs from a sequence of data can sometimes be difficult, given the nature of the task and the output. For example, it is possible that one may need the data from the first element of the sequence to be perfectly relevant. Nevertheless, depending o,n the architecture of the RNN, it may be difficult to achieve decent results if the input sequence is too long. The last phenomenon can be seen as a lack of long term memory in RNNs.

4.4. Variants of the RNN

So as to overcome the problem of the lack of memory of classical RNNs and therefore to take into account the long-term dependencies of our skeleton data, we will now briefly study two types of units that can replace the hidden units in RNNs. Both are based on the following equation:

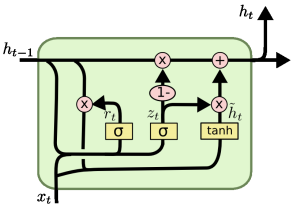
$$h_t = f_t \odot h_{t-1} + i_t \odot \tilde{h}_t$$

where f_t (forgetting gate) and i_t (entering gate) are sigmoid functions used on a linear combination of h_{t-1} (the internal memory), x_t (the input at time step t), and a bias, and where $\tilde{h}_t$ represents a possible candidate for h_t. The idea of this process is to allow the neural network to remember the components of the previous hidden states, which can help improve the long-term dependencies within the network. Indeed, the forgetting gate f_t is used for memorization and the candidate hidden state $\tilde{h}_t$ incorporates an output gate that allows to give more weight to some components of the previous hidden state.

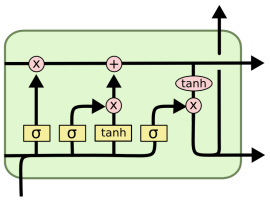
4.4.1 Gated Recurrent Units (GRU)

A solution to maintain the stability of the model is to use Gate Recurrent Units, or GRUs [**?**], which specify relation between the forgetting and input gates, such that $f_t = 1 - i_t$. We then obtain the following equation:

$$h_t = (1 - i_t) \odot h_{t-1} + i_t \odot \tilde{h}_t$$



(a) GRU Architecture.



(b) LSTM Architecture.

4.4.2 Long Short Term Memory Units (LSTM)

An other solution is to use Long Short Term Memory units, or LSTMs [3]. The idea is to apply an hyperbolic tangent to the hidden state to bind its values. Such model are useful in this project since they help remembering the long-term dependencies of the skeleton sequences.
Now that we have outlined the theoretical aspects and details of the models we used, and tried to explain why some specific models were interesting in our case, and what were the dangers of using them as they were, we will dive into the concrete approach we adopted for all our experiments.

5. Approach

5.1. Prior considerations

At the beginning of our research on the project we had to choose a either a 2D coordinate system (pixel coordinates) or a 3D one (from the Kinect camera). It was the latter that we favoured, on the assumption that the 3D data provided more relevant information about the positions of the joints, and more importantly we supposed that we would find a way to generalise the results obtained to two-dimensional data. This assumption made even more sense when we computed the correlations[3] between the 2D values and the 3D values of (x, y), which was approximately equal to 97%.
As previously explained, we decided to limit ourselves to 9 classes of actions[4] when presenting the dataset, notably in order to reduce the computation time.

5.2. Architecture of the model

The architecture of our model is relatively simple and remains the same throughout the different experiments or analyses conducted. Although time did not allow us to carry out more studies on other possible structures, the results we obtained are good enough to be satisfied with it. The *Figure* 9 illustrates the model we implemented.

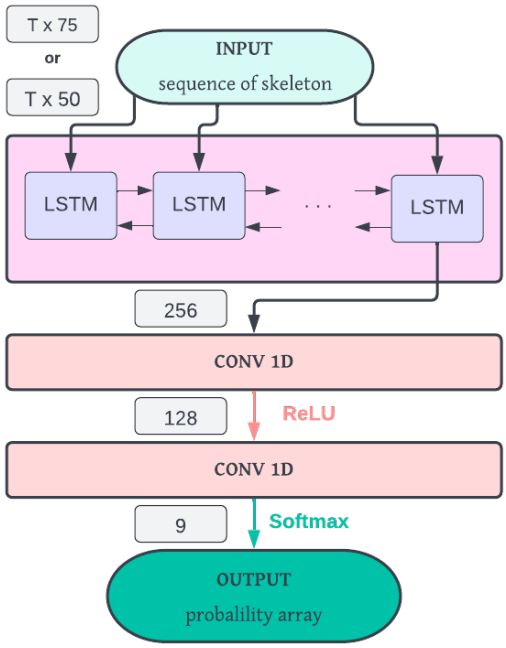


Figure 9. Architecture of our model.

[3]Pearson correlation value.
[4]Representing actions that are relevant to monitor in a public place.

The input represents a sequence of 25 points, one for each of the joints, either 2D or 3D. To ease the manipulation of the data we will format to a $(T, 50)$ shape if we consider two-dimensional data or to a $(T, 75)$ shape in case of three-dimensional data. Here the integer T represents the length of the sequence.

The input is then given to a bidirectional single layered LSTM network, with a hidden size of 256. Given the explicit sequentiality of the data, we immediately thought of including a recurrent part in our model in order to take into account the information from previous frames.
Once the input is processed, the final hidden state of the model is given as input to a classifier made of two one-dimensional convolutional layers with a ReLU activation function in between, and a softmax operator at the end which allow us to categorise the action accordingly.

When training the model, using a NVIDIA® GeForce® GTX 1660 Ti, on a given dataset we always consider that: the number of epochs is 200, we use a cross-entropy loss along with an Adam optimizer, the batch size is equal to 32, the learning rate starts at 10^{-4} and is divided by 10 every 10 epochs, and the split ratio for the train dataset will be 80%.

5.3. Influence of the dataset

5.3.1 Situations

In the context of our research we decided to explore three situations corresponding to three criteria that make sense to consider in the prediction of actions.

The first situation is relatively simple. We consider as a dataset the set of sequences taken individually. Thus the task of our network is simply to take as input a sequence of skeletal representations and from this to return the index of the corresponding action. We will refer to this situation as the **complete samples** one.

The second situation echoes real life application of action recognition. Indeed sometimes for technical or practical reasons we don't have access to the exact position of some of the joints. Thus given a dataset with a certain random amount of hidden joints, we want to know how well the model can adapt and learn to perform valid predictions. We will refer to this situation as the **masked samples** one.

The third situation proposes to study the effect of the length of the input sequence on the predictive ability of the model. Indeed, on certain occasions, such as a fall, it can be interesting to be able to quickly detect the action performed. We will refer to this situation as the **truncated samples** one.

5.3.2 Building the datasets

Since the complete samples case corresponds to our base case, there is no need to transform our initial dataset for it. Since the samples processed had all different sequence lengths in terms of frames, during training and testing we had to adjust their sizes in the same batch using a PyTorch padding function dedicated to RNNs. In the latter process we decided to use 0 padding at the beginning.

For the masked samples case, given a percentage of number of joints to hide, we randomly sample the joints to hide for each frame of each sequences in the original dataset in order to create a new one. The percentages considered are all the multiples of ten starting at 0% up to 90% As shown in *Figure* 13 of the Appendix, the values considered are far from -100, as consequence we decided to use this value to define a missing joint.

For the truncated samples, given a percentage of number of frames, we only consider the corresponding first frames for each of the sequences in the original dataset. In this case, the percentage range is 10% to 100%.

5.3.3 Results

We trained three instances of the network over the three dataset (*Figure* 14 in the Appendix). We will respectively name them **basic model**, **masked model** and **trunc model**. Due to a very slow convergence process we had to double the number of training epochs for the masked samples case.

model	**0**	**1**	**2**	**3**	**4**
basic	99	98	99	98	98
masked	98	96	96	97	99
part	98	95	98	97	99
model	**5**	**6**	**7**	**8**	**avg**
basic	97	98	97	99	98
masked	97	95	94	94	96
part	95	93	88	97	96

Table 3. Accuracies over the validation set of the first situation.

Although the basic model perform betters on the first dataset, the other models are really close, which suggests that the first situation can be easily addressed while considering one of the other situations.

Then we decided to confront all the models to the second datasets from the masked samples situations. A plot of accuracies per percentage of the skeleton visible is available in the *Figure* 15 of the Appendix. The results we obtain are relatively disappointing since the second model, although trained specifically for this task, is not able to significantly outperform the other models when the percentage of hidden joints is too high. In particular when less than 90% of the skeleton is available it does not seem to be very useful to try to make a prediction.

Finally we confronted the models to the last dataset from the third situation. As the previous situation, a similar plot of accuracies per percentage of the sequence is available in the *Figure* 16 of the Appendix. Now, as expected, the part model performs way better on part of the sequences (it is able to reach a 75% accuracy with 40% of the sequences).

From the study of the datasets we understand the importance of having masked or truncated data to ensure that the model is able to generalise to very specific cases, and to have more finesse in the way it predicts the current action. Thus, from now on, our default dataset will have the following characteristics:

- We randomly hide joints of the skeletons, up to possibly 10% according to the results we obtained previously.
- We consider parts of sequences between 10% and 100%.

5.4. Influence of the memory

Now that we have seen the influence of the data structure over the prediction ability of our model, let's study the influence of the memory over the network and how it adapts in order to perform prediction while possibly having in memory information about previous events.

5.4.1 Consecutive actions

In this first situation, considering the same three models, we want to know how well they manage to do predictions given a context. More precisely, if an other action has been performed just before, are the networks able to correctly predict the last action ? The *Figure* 17, in the Appendix, displays the results obtained. The first three heat maps give the accuracy obtained, and the last three ones given the average number of frame of the second action needed to correctly predict the result.

As we can quickly see, all three models perform rather poorly, even very poorly for some classes. Even when it comes to predicting two successive actions that are of the same nature. One exception, common to all three models, is nevertheless visible. Indeed, for the last class, which corresponds to the action of falling, these models seem to have a good behaviour. This can be explained by the fact that it is a singular action for which all the joints of the skeletons have a very similar dynamics.

5.4.2 Time window

Similarly, we are interested in the ability to predict two consecutive actions, but this time considering a sliding window. The prediction made corresponds to that of the last sliding window, which makes it possible to either cut the second action or to consider the end of the first one according to the respective length of the two sequences. The *Figures* 18 and 19, in the Appendix, display the results obtained for respectively of time window of 100 frames and an other one of 75 frames.

There is a notable difference in terms of results. This phenomenon is understandable because here the long-term memory of our models is only slightly affected, as we restrict ourselves to the last frames which contain the essence of the action to be predicted, taking into account the average and median number of frames for a given action (which we briefly studied when we introduced the dataset). These good results raise the question of the relevance of training a model on sequences longer than 100 frames. In particular, to make live predictions on potentially long sequences on which we could restrict ourselves to a finite number of the last frames.

5.4.3 Video stream

Before implementing the our latest models, which take into account the results of previous experiments, we wanted to create an interface to visualize predictions. Screenshots of this interface are available in the *Figures* 10 and 11.

This interface is relatively simple. On the left side we see the evolutionary bars that translate the probability of an action being performed. Whatever model is used to predict the current action uses 2D or 3D data, it is the pixel coordinates that we use to display the skeleton on the screen.

Figure 10. (A) Visualization of predictions (from 3D data).

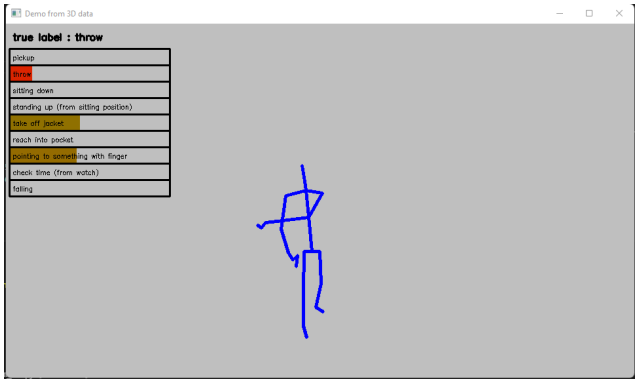


Figure 11. (B) Visualization of predictions (from 3D data).

5.5. Final models

The basic structure of the two final models we developed and analyzed is not different from the previous ones. In fact, it is entirely based on the architecture available in *Figure* 9. We believe that the study of the training dataset and how to make a prediction has a potentially greater impact on the accuracy and success of the results.

5.5.1 Dataset & Training

The formatting of the data given as input to the templates is somewhat different. Overall, it is the same as described at the end of the analysis of the three previous models, but two distinctions are now made: we consider successions of events. For both models, we concatenate a complete action, which we call a **context**, to a percentage of the sequence of a second action, which we call a **situation**.

In the first model, whether for training or prediction, we will consider the last 100 frames of a context + situation association. In the second model, during training we only consider a given situation, however we do not reset the internal memory of the LSTM network. Thus when predicting a video stream with the second model, we will only give one frame after another using the fact that giving 100 frames at once is equivalent to giving 100 frames but successively to this model.

5.5.2 Results

In contrast to the other models, we only trained on 40 epochs because the size of the new dataset is such that the processing time of all the batches is much larger. However, we obtained more than interesting results. It should be noted that the evolution of the loss and accuracy of these models may seem poor if we refer to *Figure* 20. However, as these values are calculated on a dataset of sequences that are potentially highly truncated or with hidden parts,

they do not necessarily make sense. It is more the shape of the resulting curves that carry more meaning, and attests to the good evolution of the learning process. The results obtained, similar in nature to those of the first models, are available in the Appendix (*Figures* 21 to 24).

When only complete situations are considered (100% of the sequences), the first models are impressive for their efficiency, as shown in the *Table* 3. In comparison, the second model is not very convincing, but this phenomenon will be explained later.

model	**0**	**1**	**2**	**3**	**4**
First	100	99	99	100	100
Second	95	73	85	87	98

model	**5**	**6**	**7**	**8**	**avg**
First	99	99	99	99	99
Second	73	78	34	96	80

Table 4. Accuracies over the validation set of the first situation.

As expected, the results around the prediction according to a certain percentage of hidden skeleton (*Figure* 21) is not very promising, regardless of the two models.

The results of the predictions according to the percentage of sequence given as input (*Figure* 22), approximately follow the results that were also expected. We should precise that the prediction limit is no longer 40% of the sequence but a little higher because in the dataset that we now consider the lengths of the sequences are potentially even shorter.

When it comes to studying the memory influence of these two models the results are mixed. Indeed, assuming a context and a full situation (100% of a sequence), we observe that the first model is not very efficient, which is to be expected given its nature. However, where the surprise lies is that the second model does not achieve the performance that one might hope for (Figure 23). By using the visualization interface, we realize that it takes some time to adapt to the second model. More particularly, it seems difficult for the latter to make predictions starting from a new or recent long memory. Thus we have developed a more accurate metric to assess the predictive capacity of this model.

When we restrict ourselves to a window of 100 frames the results of the second model are not very interesting, but it makes sense because it was not developed for that. The results of the first model are very interesting (*Figures* 24) and attest the relevance of restricting to a finite time window.

5.5.3 Inference methods

In order to make direct predictions, we make a distinction between two modes of prediction for each of the two final models considered.

In the first one when given a new frame at a certain time step, we will put it in the end of a sequence, and of this sequence has a length greater than 100 we will move the first element. This dynamic structure can be described as a First In First Out, or FIFO, structure. To perform a prediction at the given time step we simply give this data structure to the first model which will then output an action label.

In the second prediction mode each considered frame is given input of the second model which retains its short-term and long-term memory. In terms of calculations, the first method is more complex because each frame requires a process of 100, while for the second method we only process one.

6. Conclusion and future work

The project was very interesting in terms of research and the deployment of knowledge that we have built during our studies. It is particularly pleasing to see what we have achieved without prior knowledge on the subject. We were fortunate to be guided by motivated tutors, who were able to give us the freedom to orient our lines of research, while providing us with relevant insights and advices.

Relevant critics could be made to the project we have developed, and therefore could be considered as **future work**. First of all, certainly for computational choices, we only considered a single model architecture without ever changing it or studying its hyper parameters. Having studied transformers several times, it would make sense to see the impact of these in the prediction results.
Second, it would be interesting to test a complete prediction pipeline to test the robustness of our model. To do this we could consider the OpenPose library [4] to extract skeletons from an image, and use our models to make a prediction.
Finally, we could consider a CNN upstream of the LSTM in order to individually process the frames and possibly derive more information from them.

References

[1] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *international conference on engineering and technology (ICET)*. IEEE, 2017.

[2] Haodong Duan, Yue Zhao, Kai Chen, Dian Shao, Dahua Lin, and Bo Dai. Revisiting skeleton-based action recognition. 2021.

[3] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997.

[4] Yaadhav Raaj, Haroon Idrees, Gines Hidalgo, and Yaser Sheikh. Efficient online multi-person 2d pose tracking with recurrent spatio-temporal affinity fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019.

[5] Hossein Rahmani, Ajmal Mian, and Mubarak Shah. Learning a deep model for human action recognition from novel viewpoints. *IEEE transactions on pattern analysis and machine intelligence*.

[6] Suneth Ranasinghe, Fadi Al Machot, and Heinrich C Mayr. A review on applications of activity recognition systems with regard to performance and evaluation. *International Journal of Distributed Sensor Networks*, 2016.

[7] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 1997.

[8] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang. Ntu rgb+d: A large scale dataset for 3d human activity analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[9] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Skeleton-based action recognition with directed graph neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[10] Ah Chung Tsoi. Recurrent neural network architectures: an overview. In *International School on Neural Networks, Initiated by IIASS and EMFCSC*. Springer, 1997.

[11] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

7. Appendix

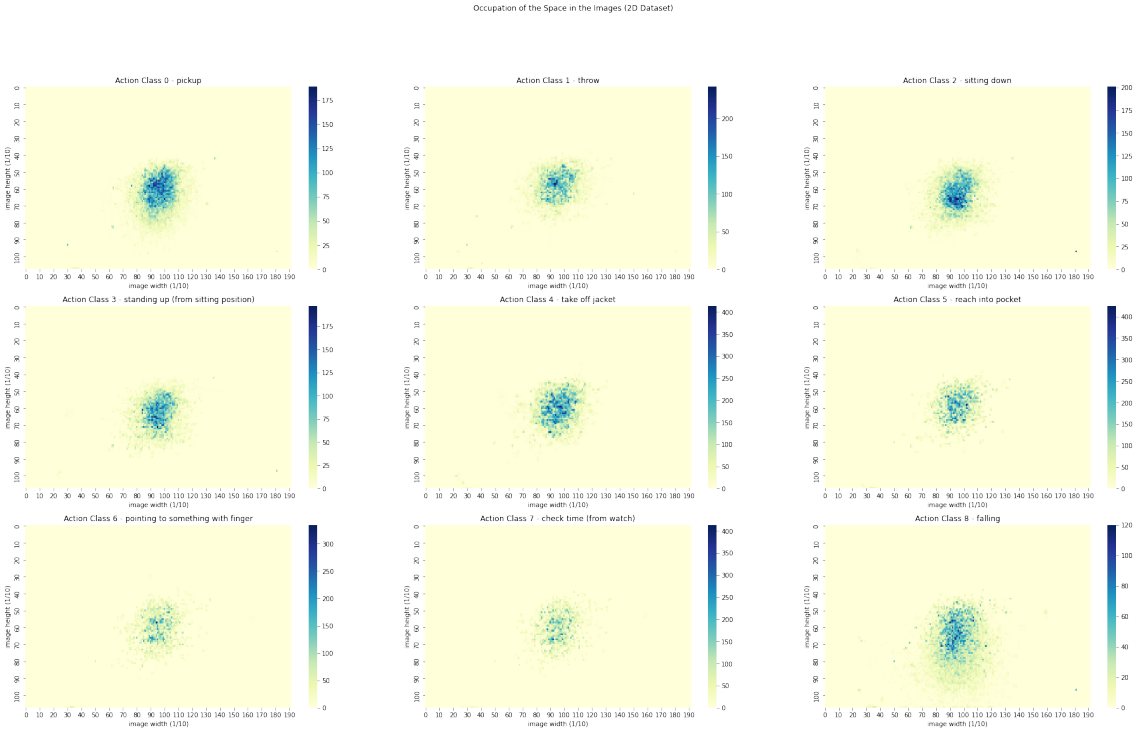


Figure 12. Heat map of the action location per class (2D data).

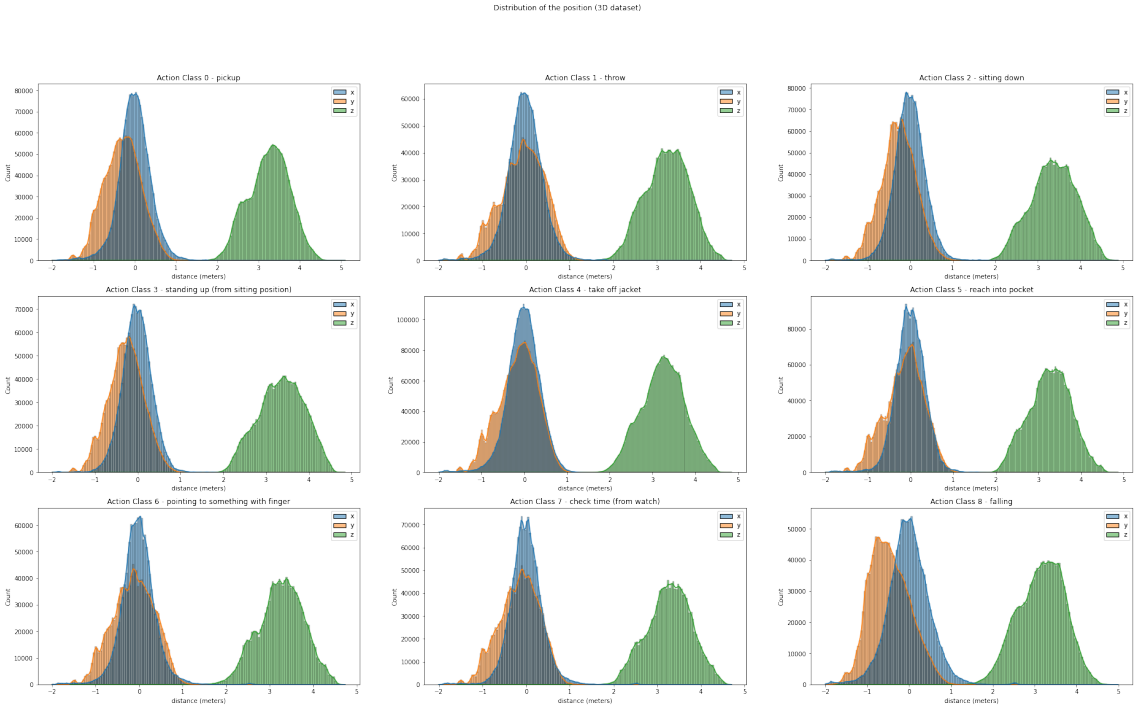
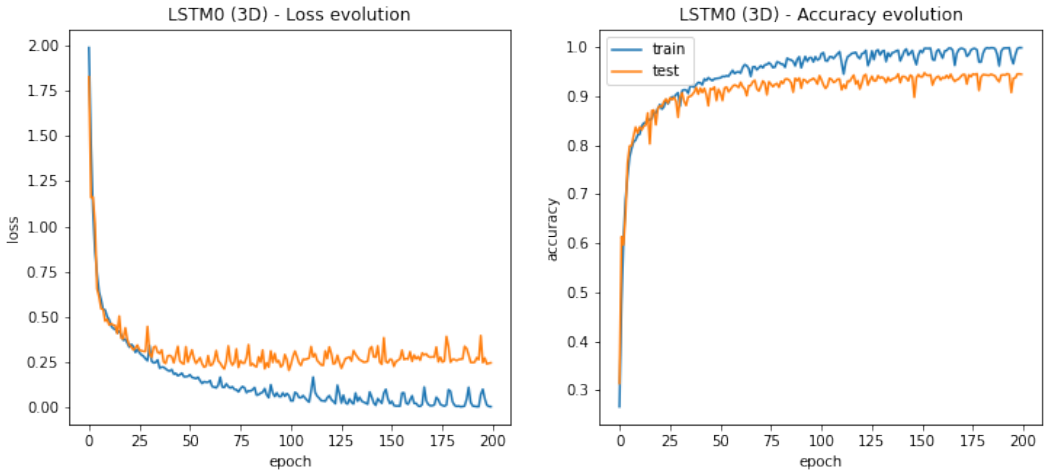
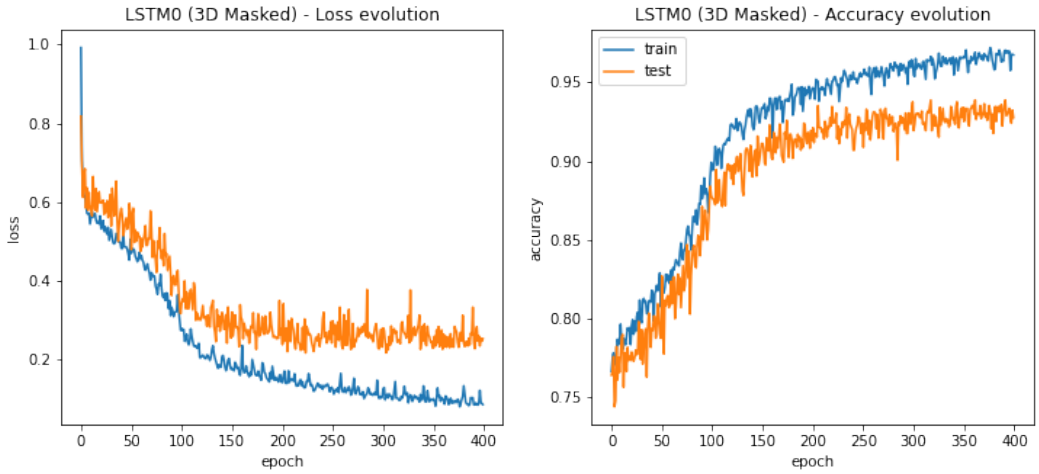


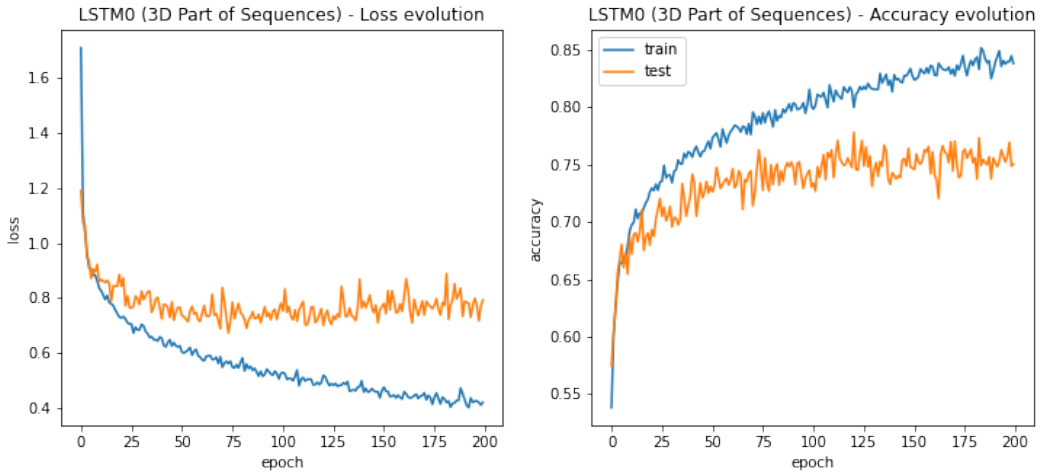
Figure 13. Histogram of the action location per class (3D data).



(a) Basic model.



(b) Masked model.



(c) Trunc model.

Figure 14. Evolution of the loss for each of the models.

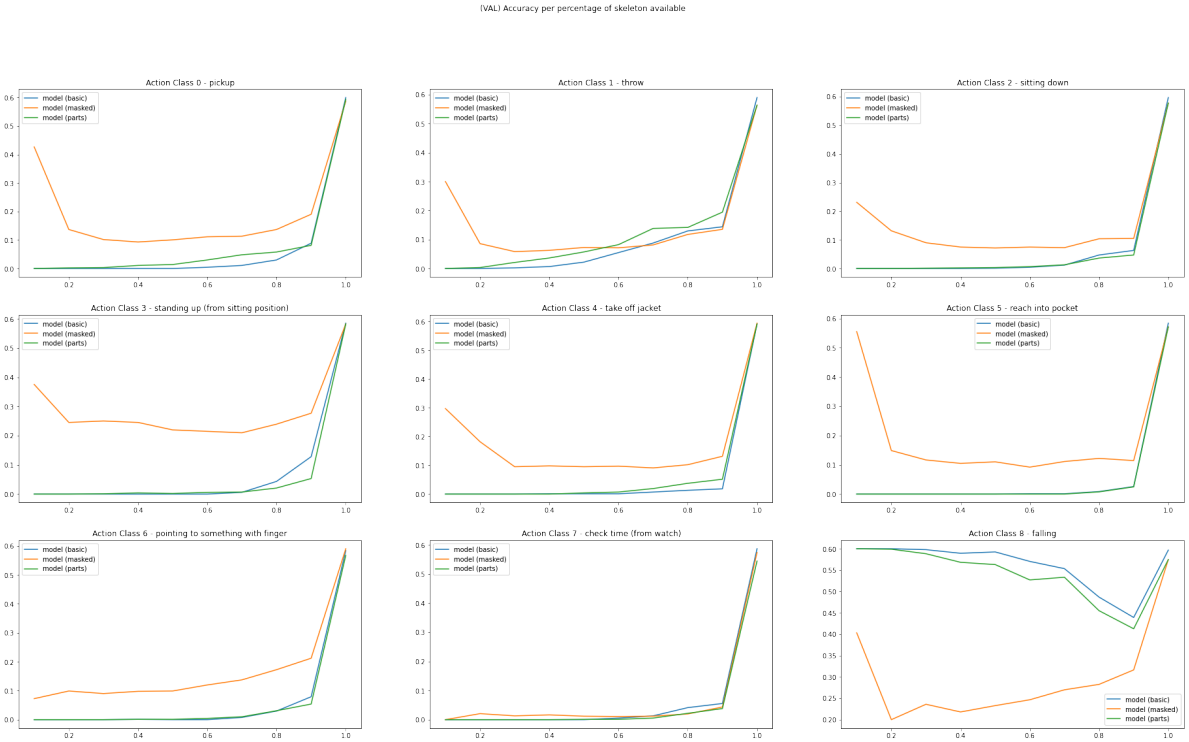


Figure 15. Accuracy per percentage of skeleton available.

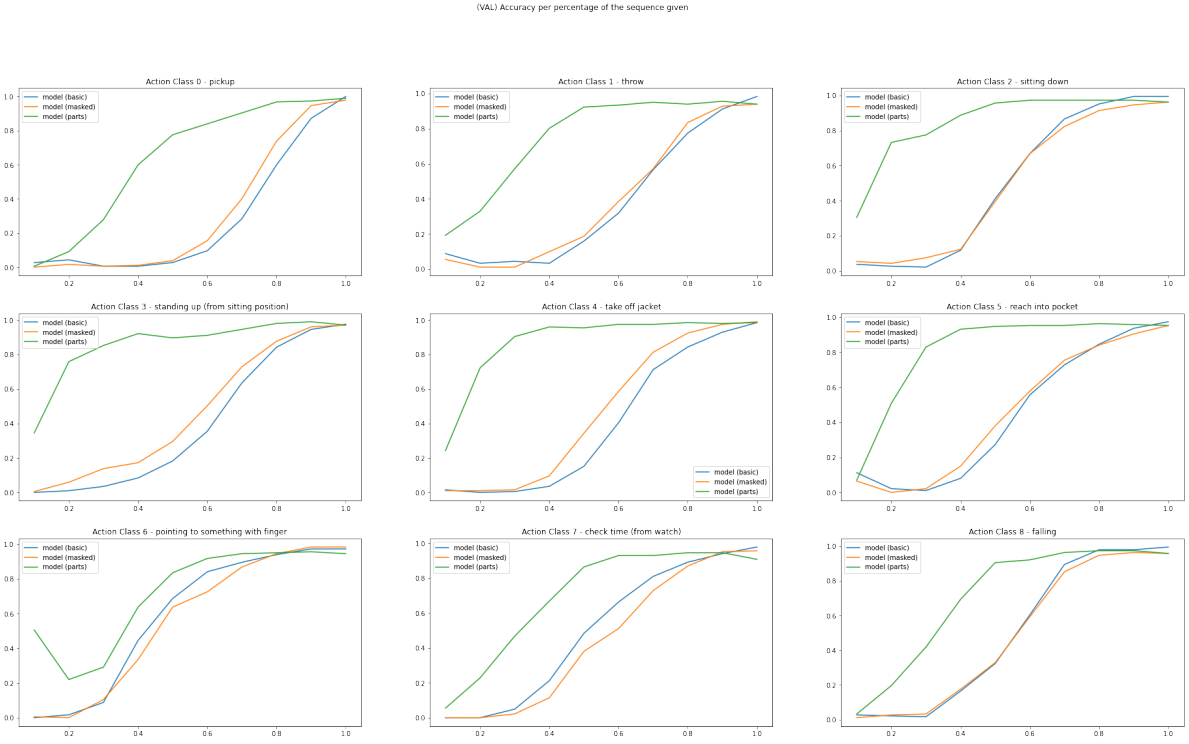


Figure 16. Accuracy per percentage of the sequence available.

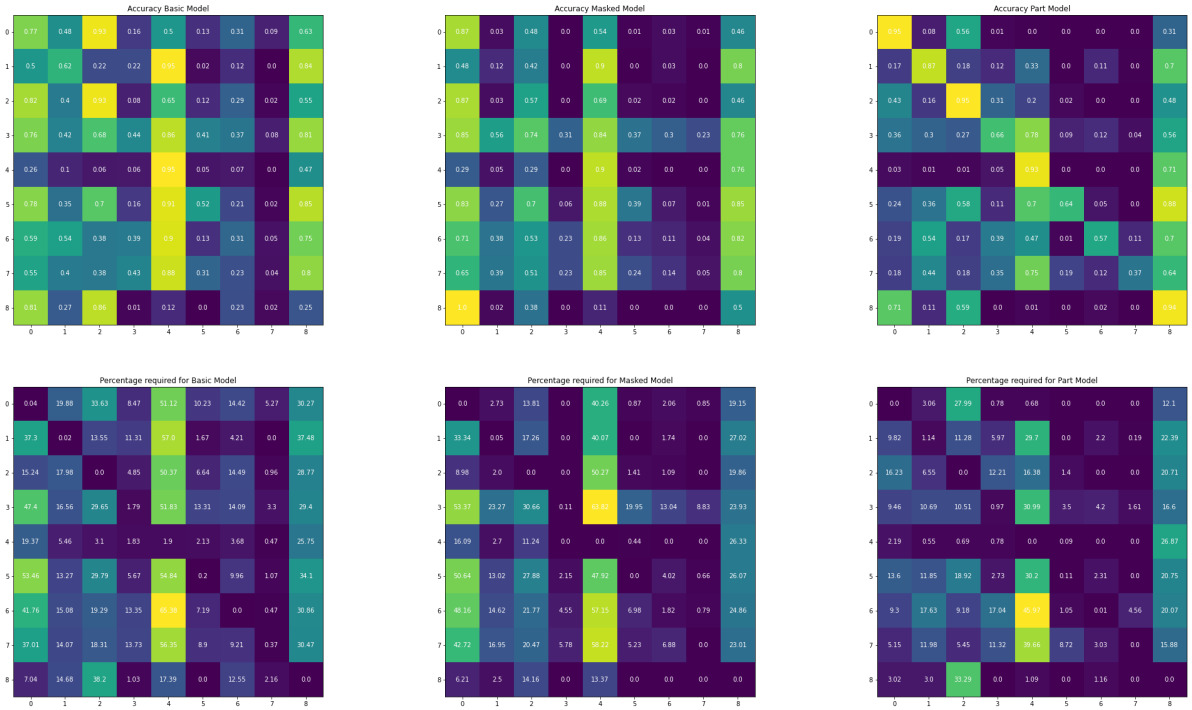


Figure 17. Accuracy per actions (lines) given a previous one (columns).

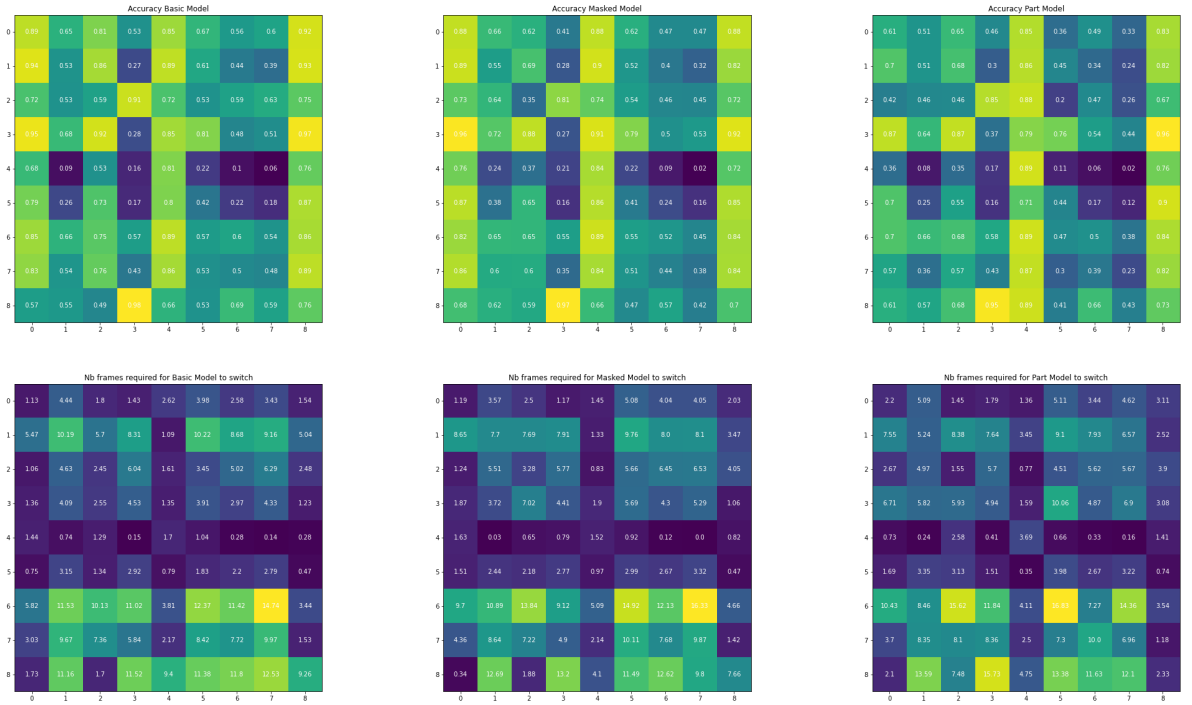


Figure 18. Accuracy per actions (lines) given a previous one (columns) using a time window of 100 frames.

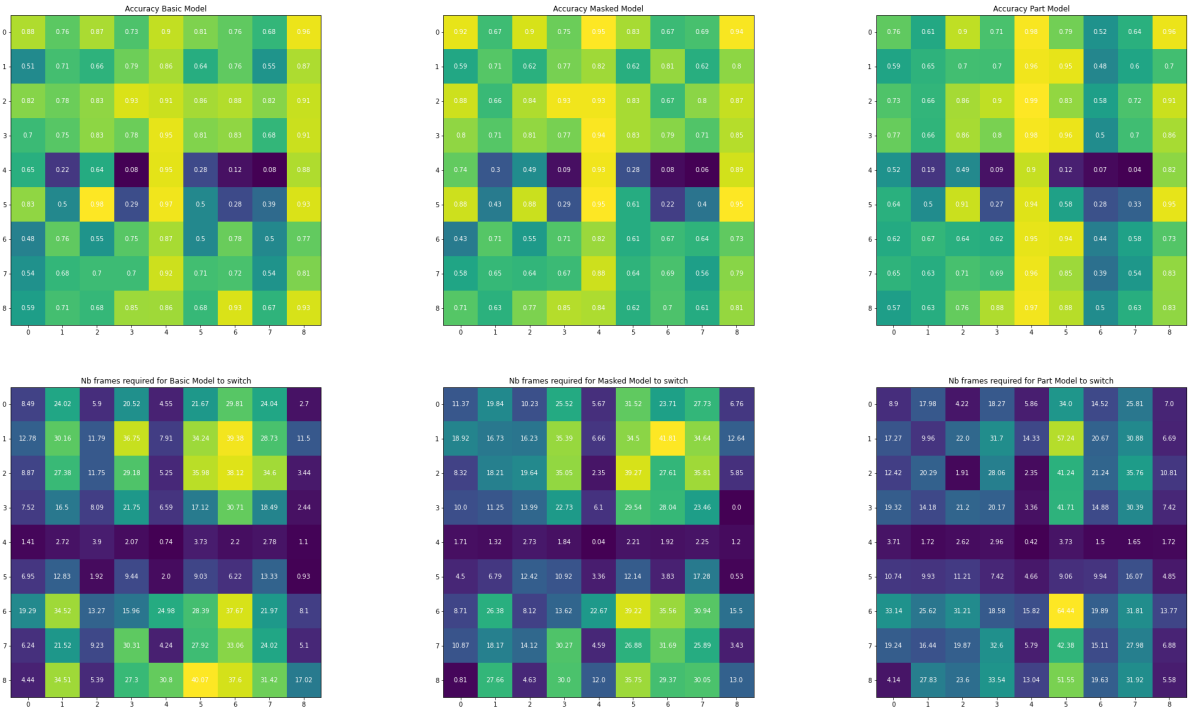
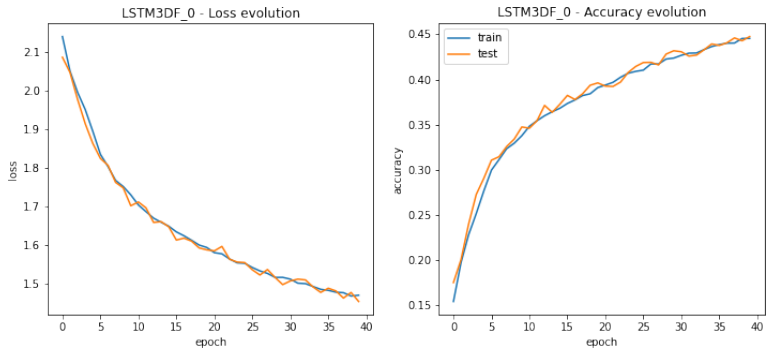
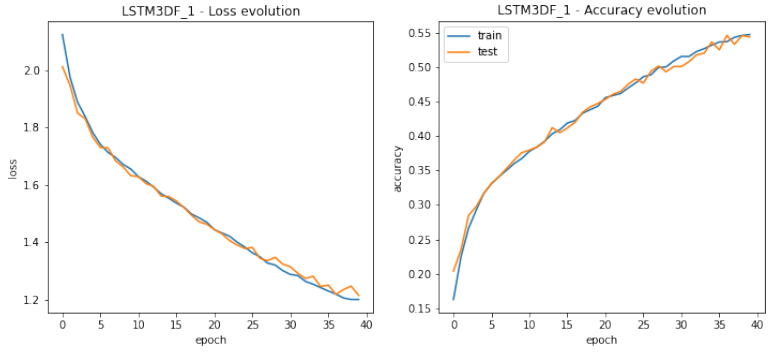


Figure 19. Accuracy per actions (lines) given a previous one (columns) using a time window of 75 frames.



(a) Basic model.



(b) Masked model.

Figure 20. Evolution of the loss for each of the final models.

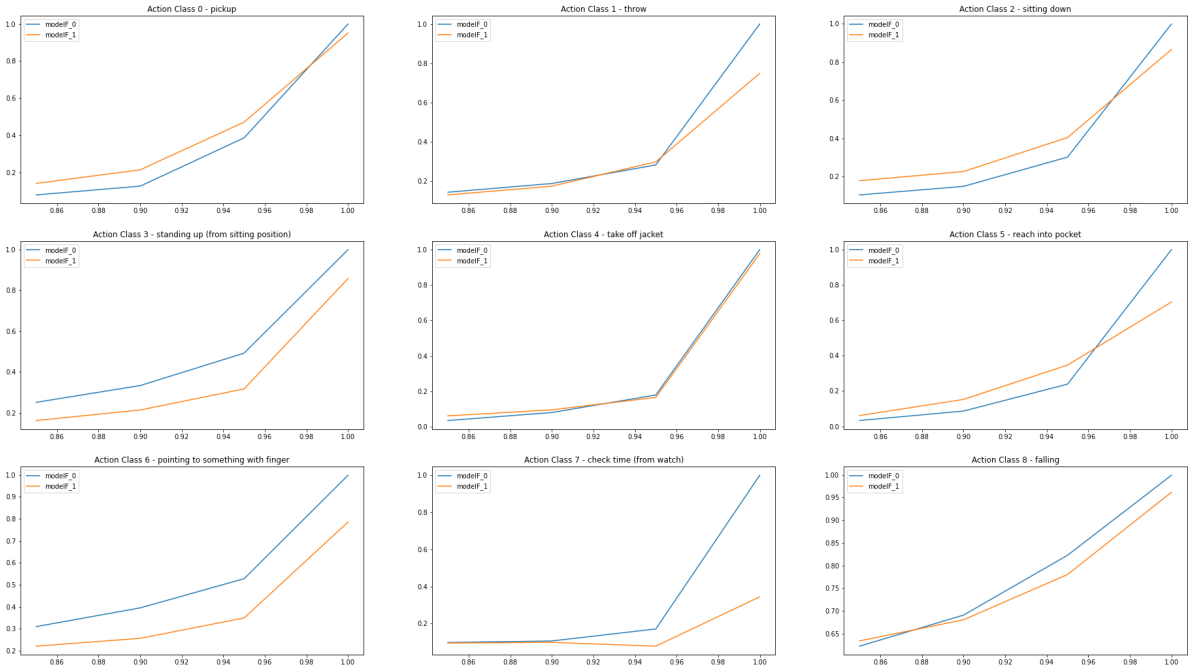


Figure 21. Accuracy per percentage of skeleton available..

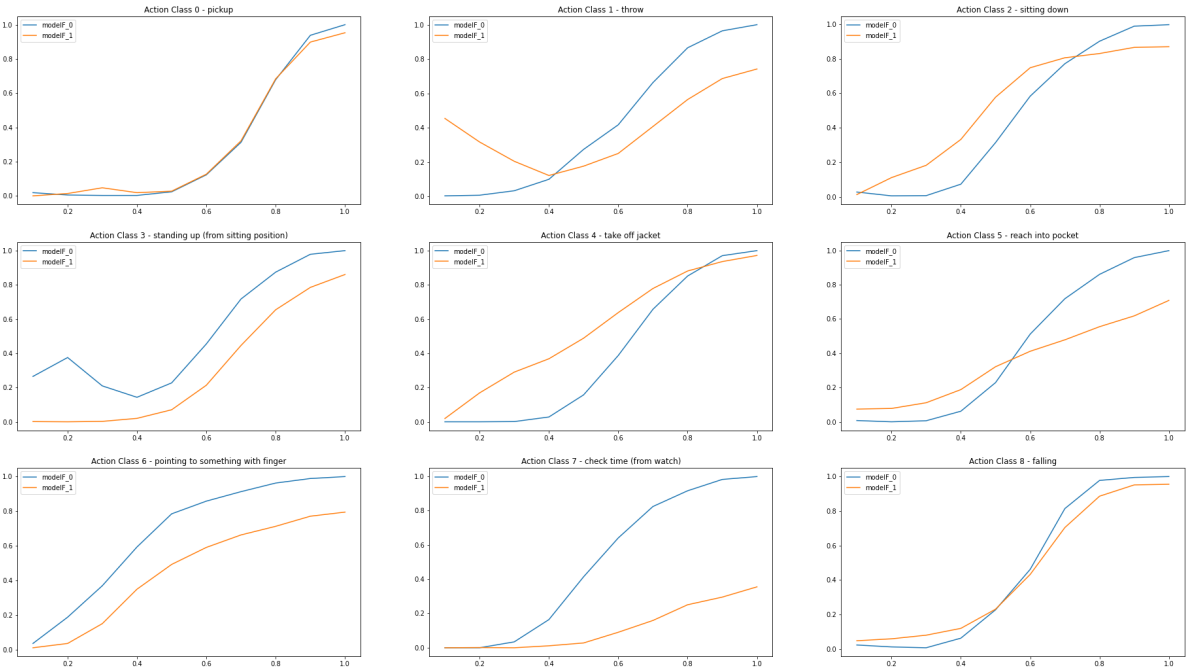


Figure 22. Accuracy per percentage of the sequence available.

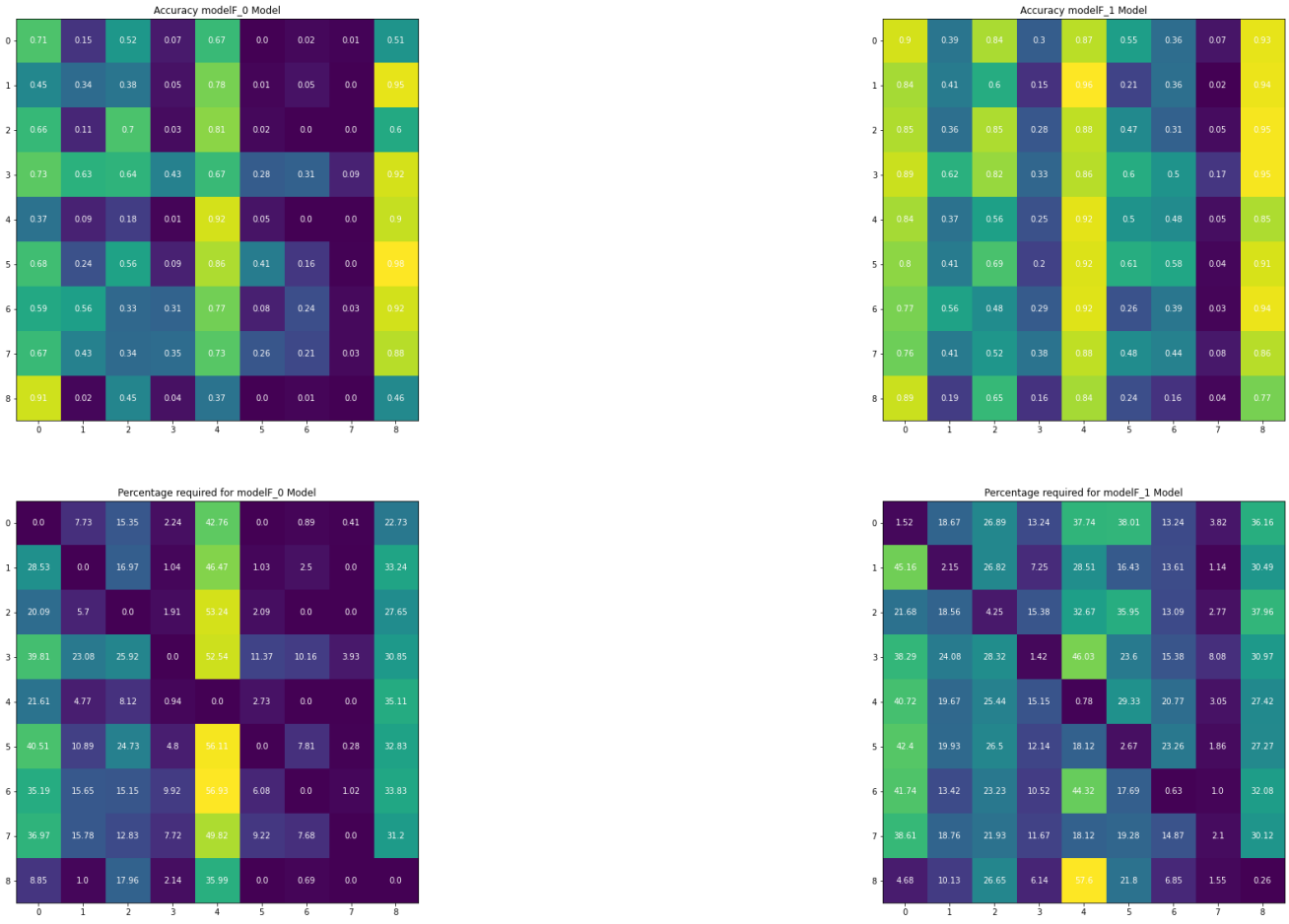
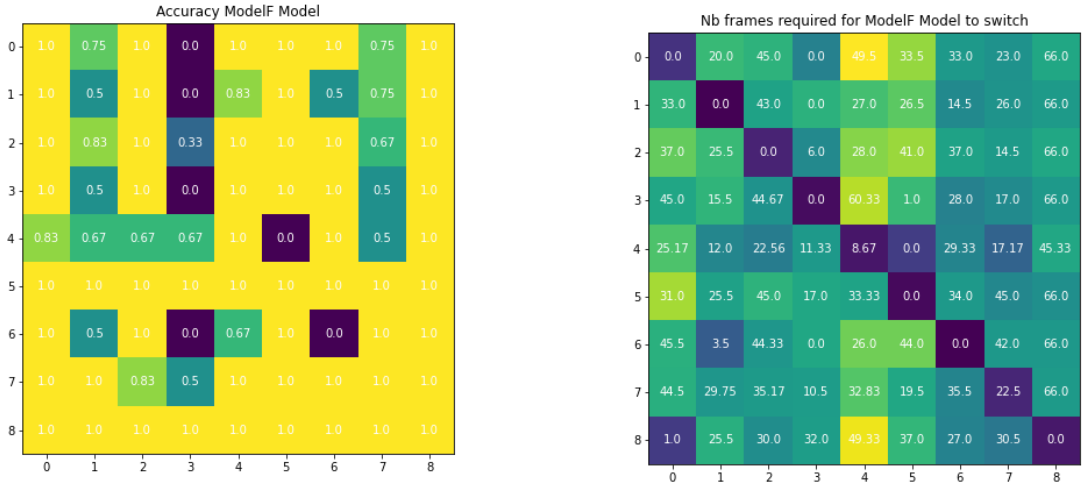


Figure 23. Accuracy per actions (lines) given a previous one (columns).



(a) Accuracy.

(b) Average number of frames to switch.

Figure 24. Accuracy per actions (lines) given a previous one (columns) using a time window of 100 frames.