Movement Primitive Analysis for Movement Recognition

A Neuro-AI Pipeline Using Markerless Motion Capture

Arefeh Farahmandi

April 2025

Movement Recognition

- What are primitives that distinguish different movements?
- Are those primitives aligning with our movement perception?
- Can we use these primitives to enhance ML applications?

Movements decomposed into fundamental building blocks called movement primitives, which the nervous system combines flexibly to produce coordinated actions. (Knopp et al. 2020)

Process

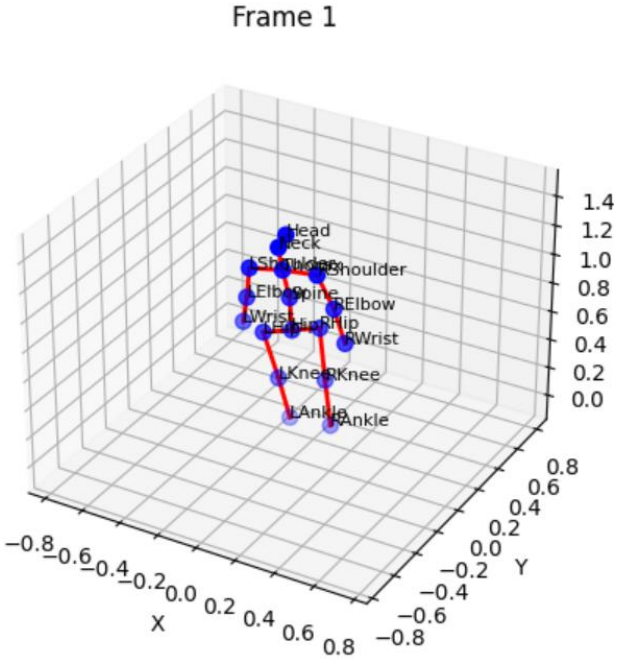
- TMP model requires 3D pose motion data:
  - VIBE : requires multiple view videos
  - DeepLabCut : requires multiple Camera Calibration
  - **Mmpose** : Powerful toolbox for 3D pose extraction

```
H36M_KEYPOINT_NAMES = [
    'Hip', 'RHip', 'RKnee', 'RAnkle', 'LHip', 'LKnee', 'LAnkle',
    'Spine', 'Thorax', 'Neck', 'Head',
    'LShoulder', 'LElbow', 'LWrist', 'RShoulder', 'RElbow', 'RWrist'
]
```

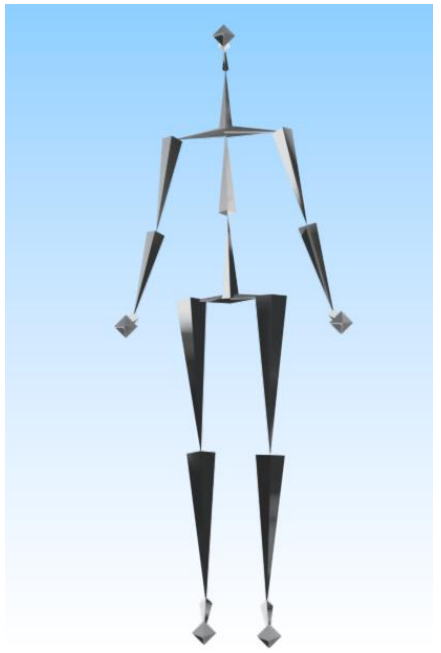
Data processing

1-Original 2D video

2-MMpose 3D extraction

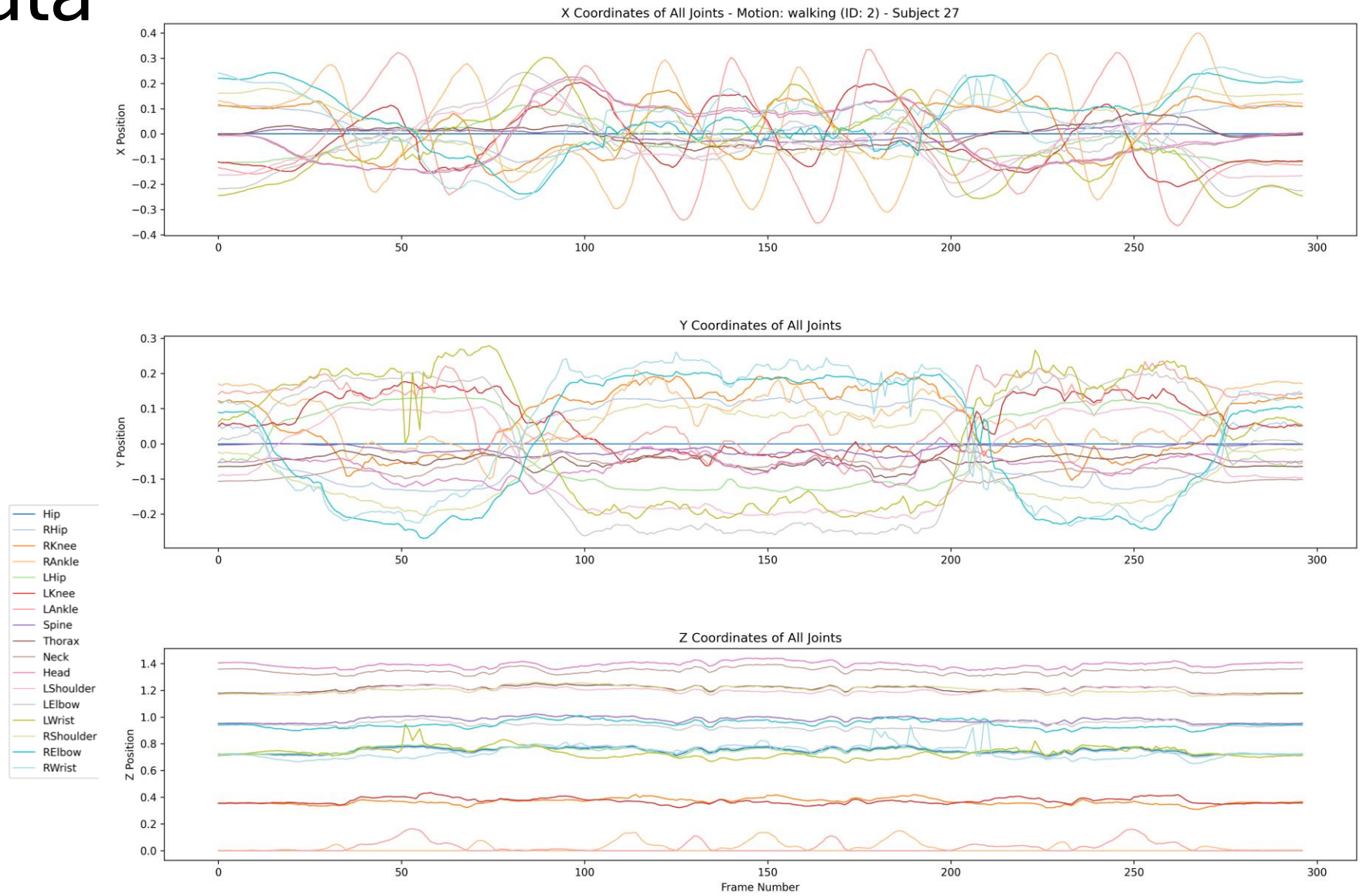


3-Motion Capture .bvh



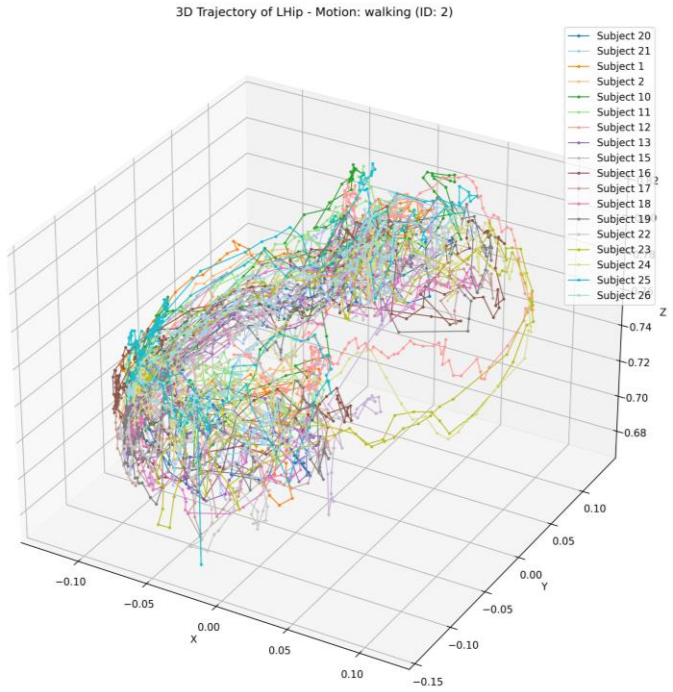
3D pose data

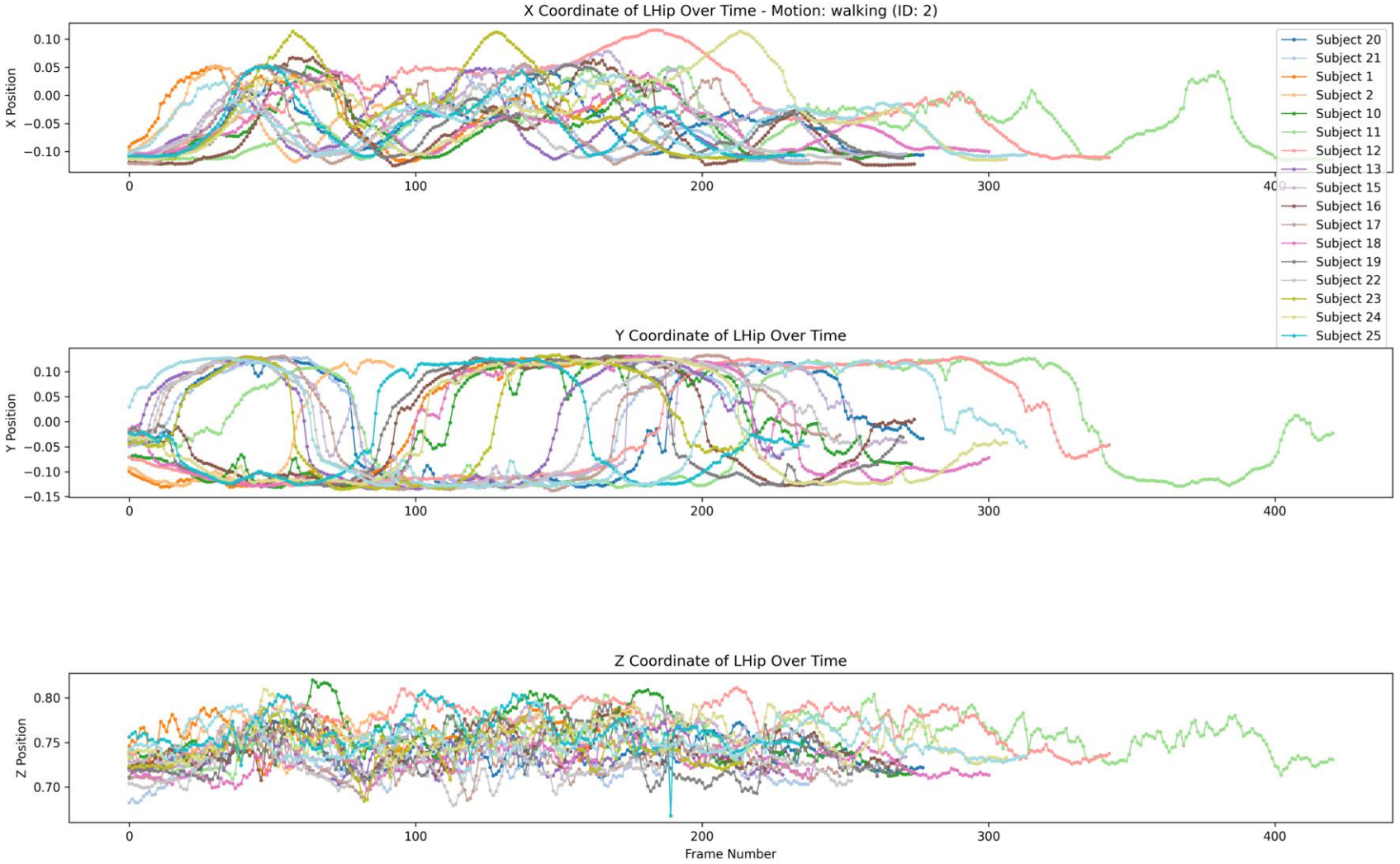
- Motion: walking
- Subject_id : 27



3D pose data

- Joint trajectory across all subjects
- Joint: Left Hip
- Motion: Walking





TMP model

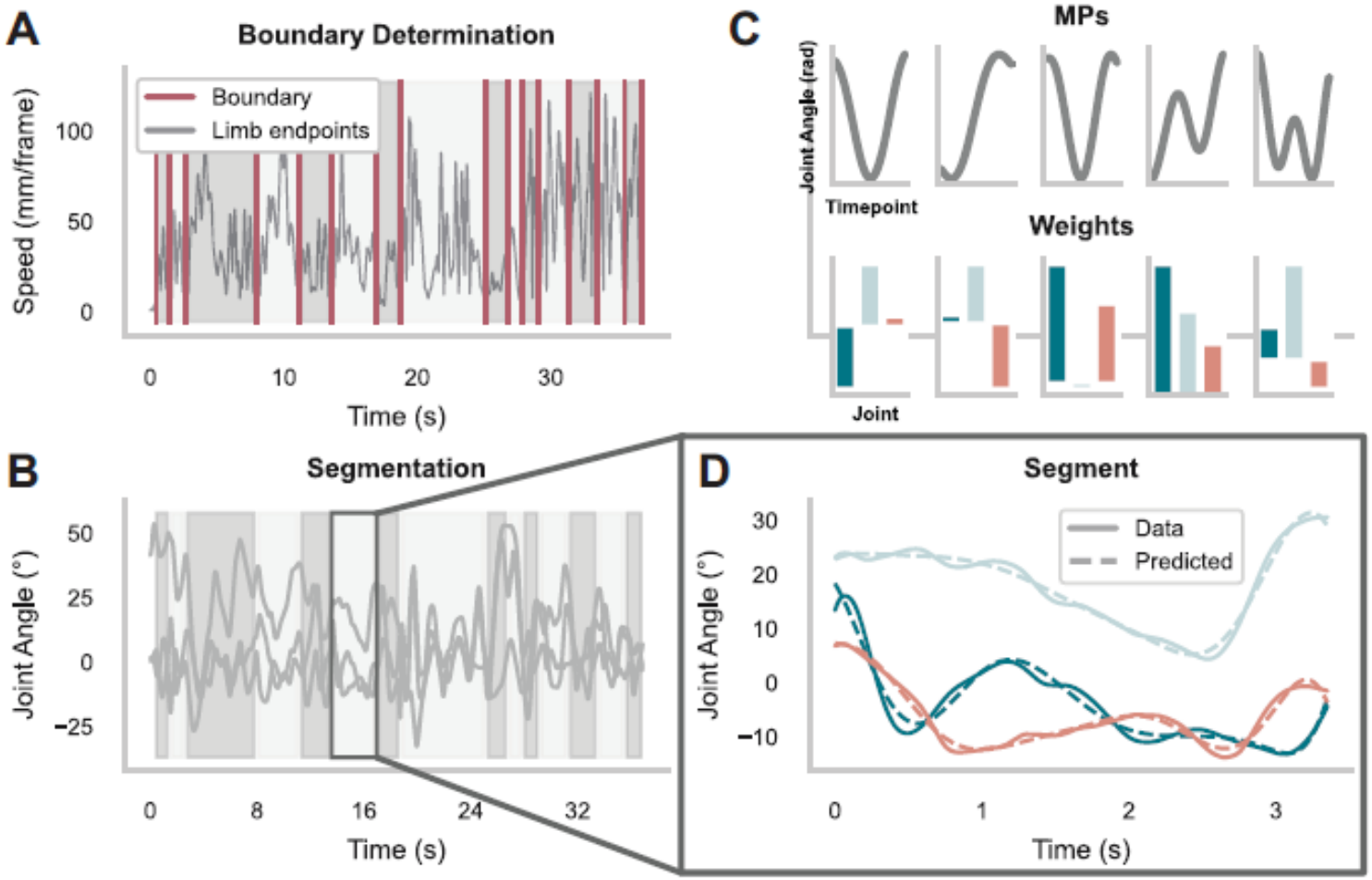
- Movement is represented as a weighted sum of MP in temporal dimension

$$X_{jt} = \sum_m W_{jm} MP_m(t) + \epsilon$$

- Gaussian prior on the weights : $W \sim N(0, \sigma_w)$, σ_w=1
- Gaussian Process priors on primitives: $MP \sim N(0, \Sigma)$, with RBF kernel
- The data were decomposed via PCA to initialize the MP and W.
- Then the model learns with maximizing the log joint probability of the data and the model parameters

$$log(p(X, W, MP(t)) = log(p(X|W, MP(t)\, p(W)\, p(MP(t)))$$
$$= log(p(X|W, MP(t)) + log(p(W)) + log(p(MP(t)))$$

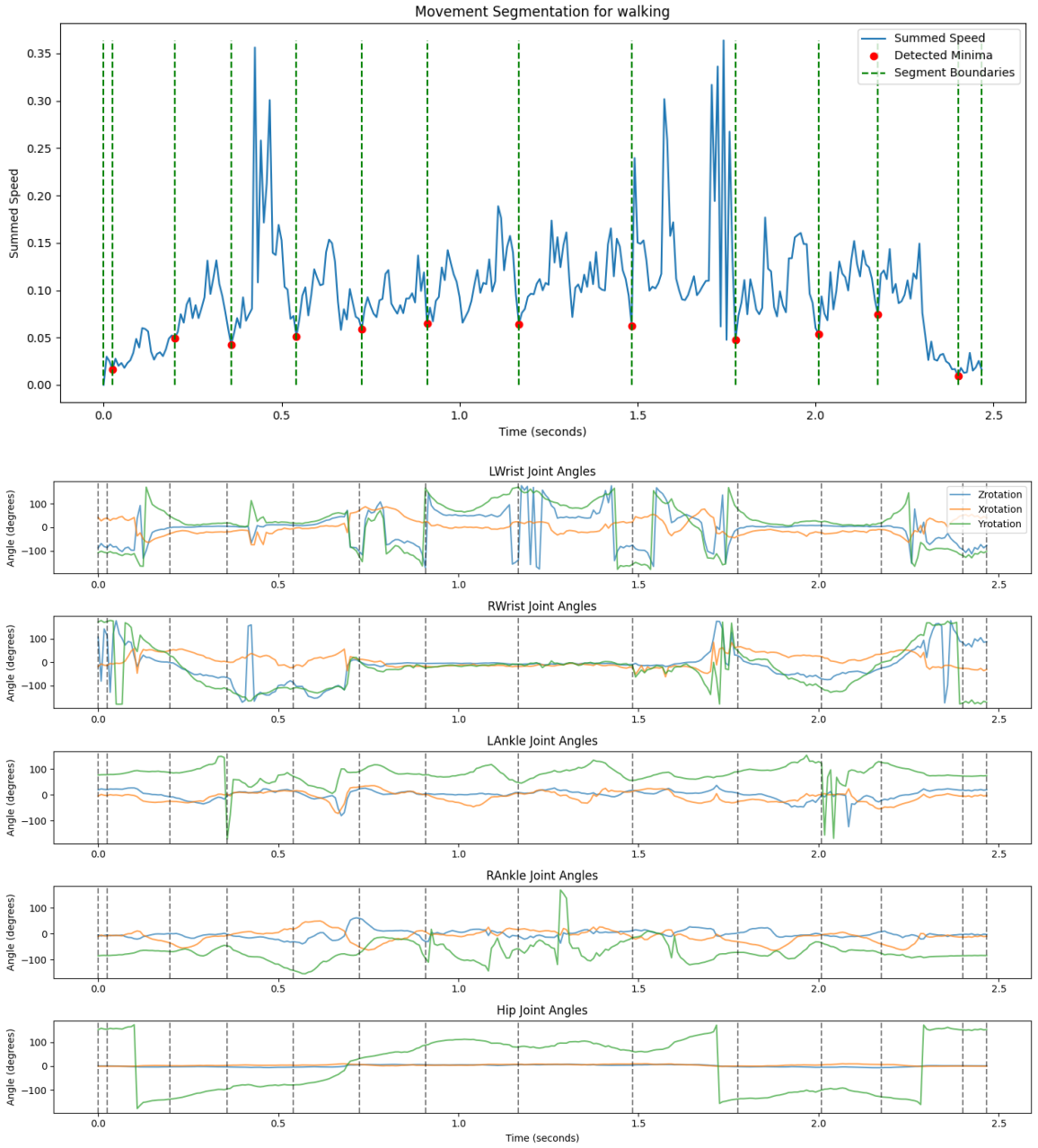
TMP model



Leh et al (2023)

Data segmentation

- Boundaries: min of summed speed of the wrist and foot markers limited to a min distance of 160 ms (19 frames)



Training

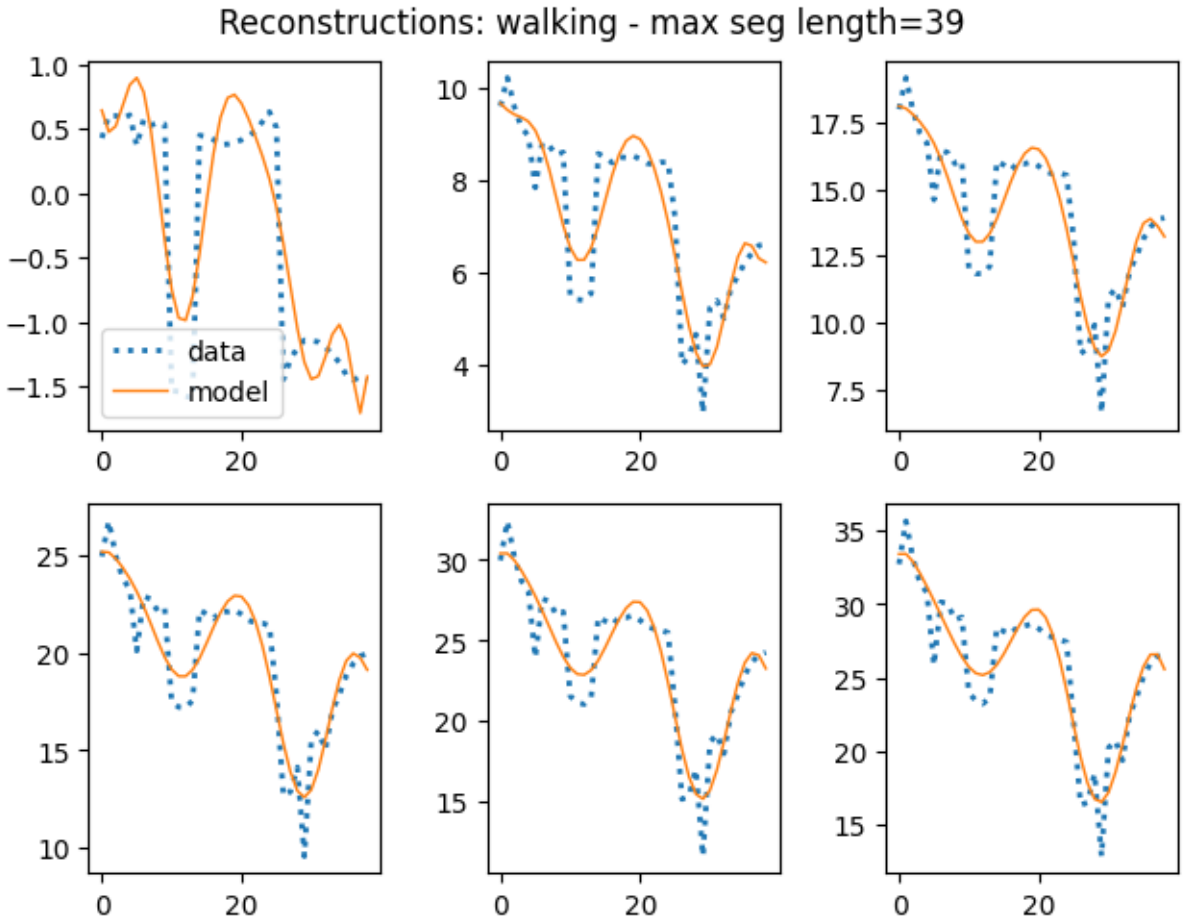
- **Optimization:**
  1. Adam optimizer for initial optimization (faster but less precise)
  2. L-BFGS optimizer for fine-tuning (slower but more precise)
- **Loss** :
  - Minimizing the negative log joint probability
- **Monitoring**:
  - Learning curve and VAF (Variance Accounted For)
- **Model Evaluation**:
  - VAF & Laplace approx. of joint prob

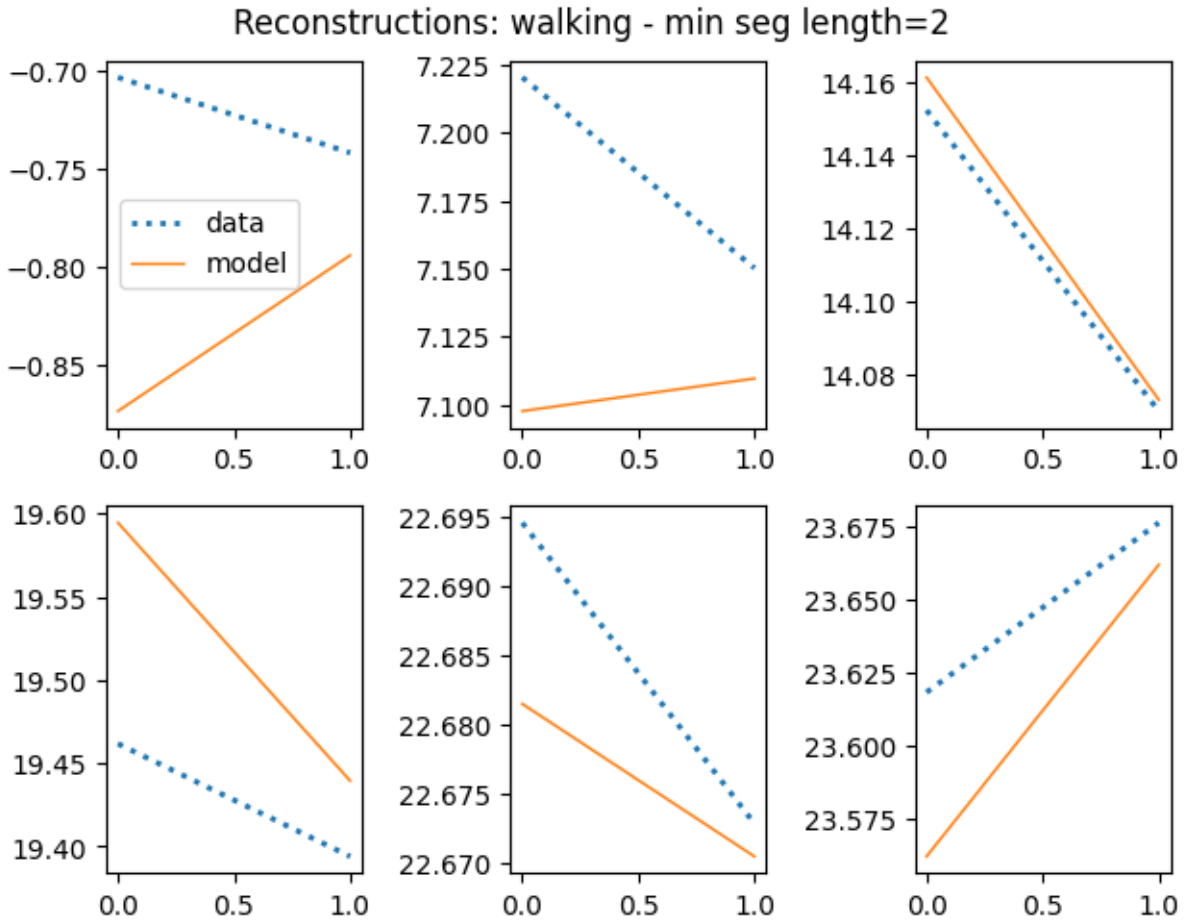
Achievements

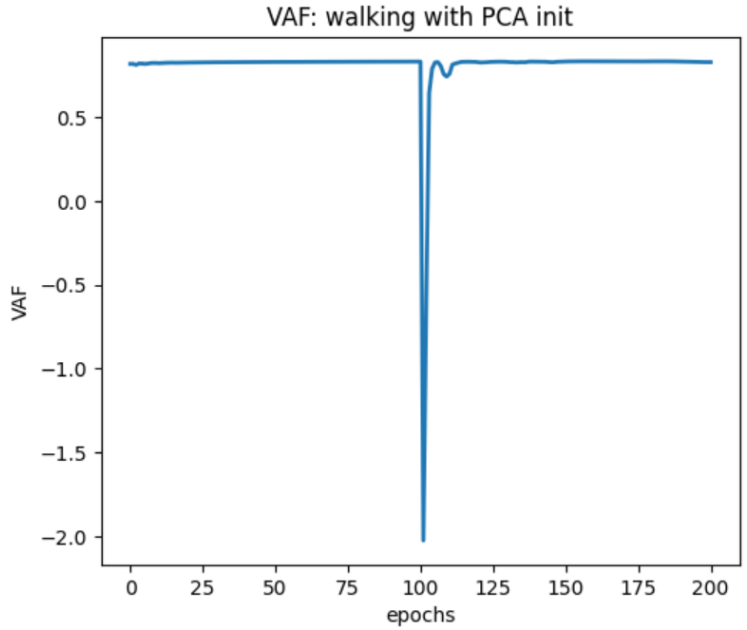
- Video sequences should be divided into separate movements:
  1. Weights are constant during the time
  2. Different performance time
  3. Limited number of segments
- Current stage:

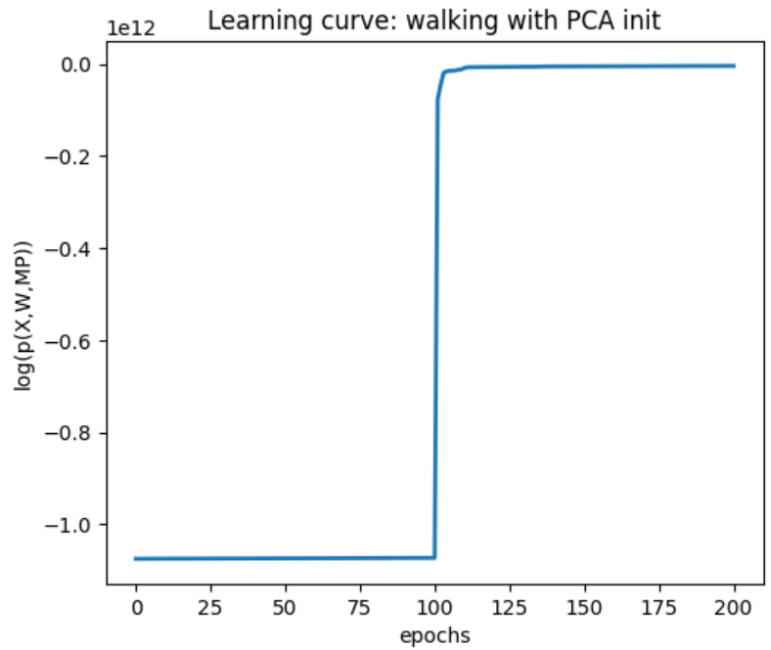
1. Single motion extraction for each subject
2. Segmentation
3. Feed all segments into a TMP model for each motion
   - VAF = 70-80%

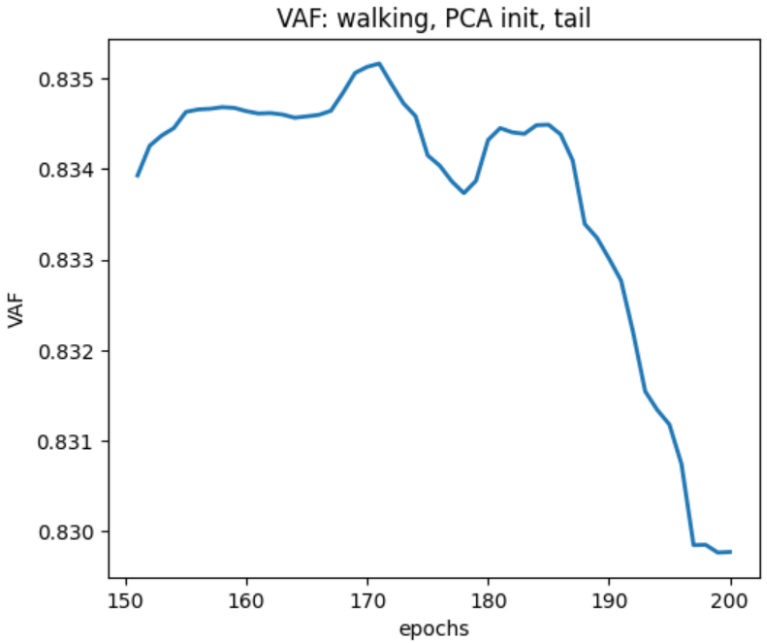
- Motion= **walking** , 327 segments
- num_mp = 10 , num_t_points = 30
- VAF = 80%

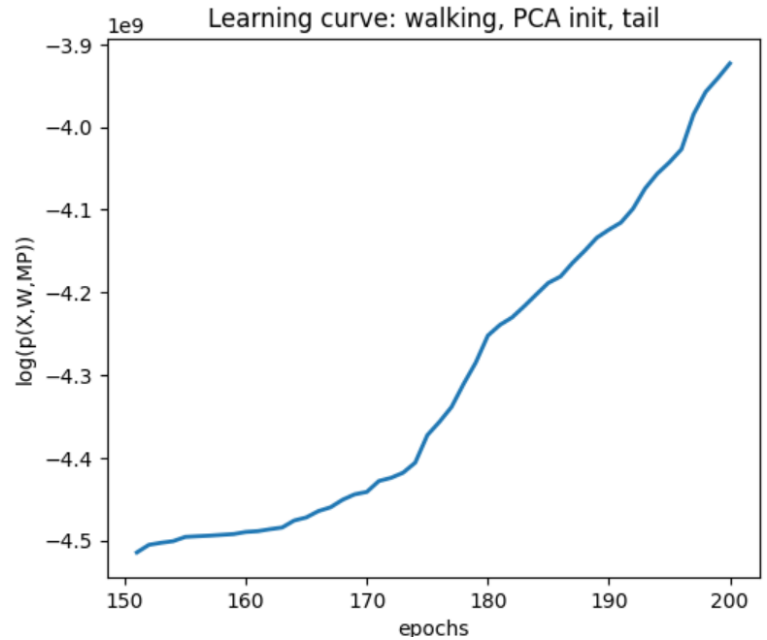




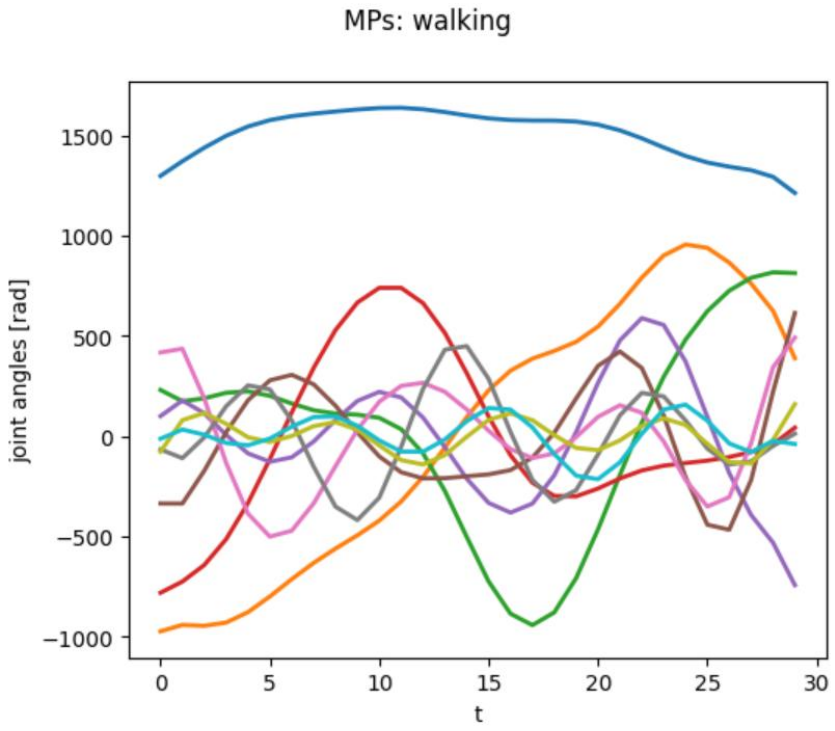

VAF: walking with PCA init
VAF
epochs
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0
25
50
75
100
125
150
175
200

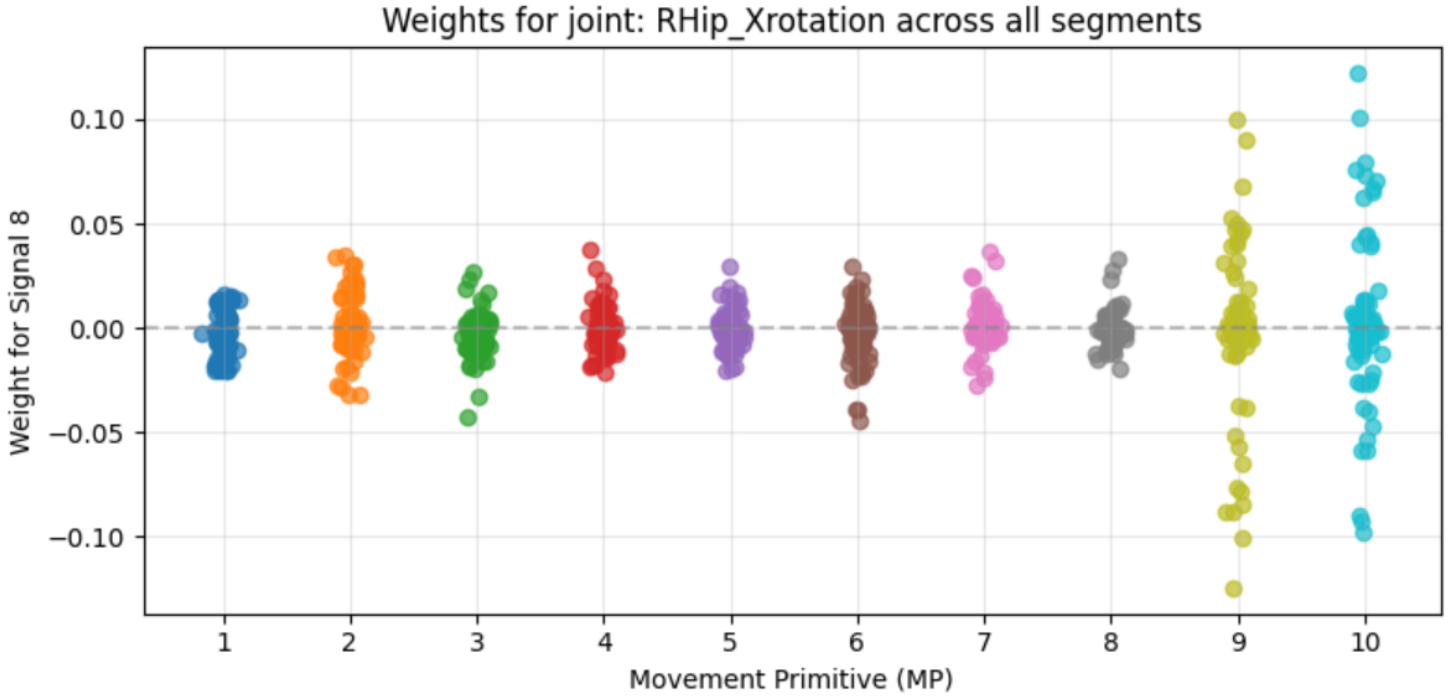

Learning curve: walking with PCA init
1e12
log(p(X,W,MP))
epochs
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
0
25
50
75
100
125
150
175
200


VAF: walking, PCA init, tail
VAF
epochs
0.835
0.834
0.833
0.832
0.831
0.830
150
160
170
180
190
200

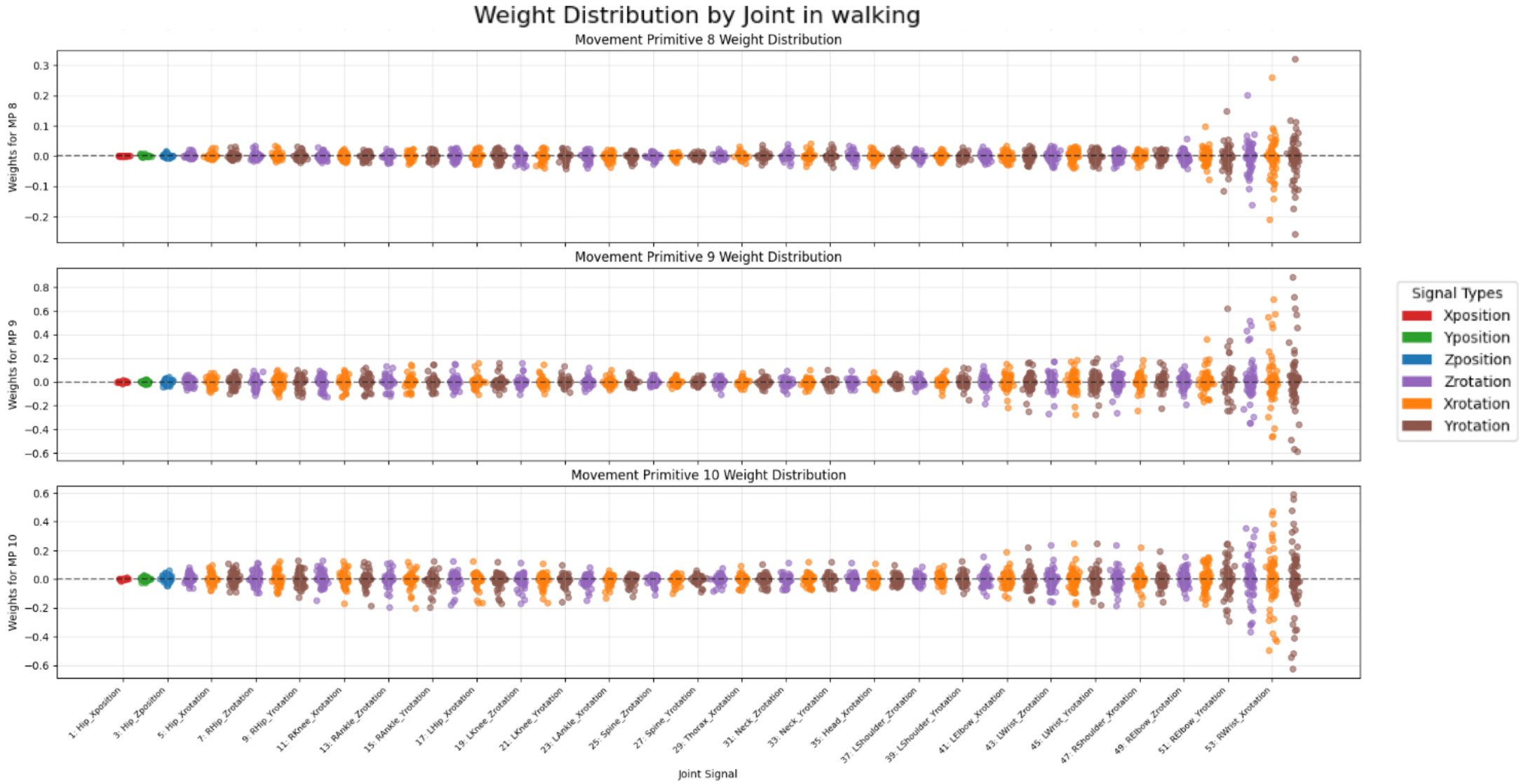

Learning curve: walking, PCA init, tail
1e9
log(p(X,W,MP))
epochs
−3.9
−4.0
−4.1
−4.2
−4.3
−4.4
−4.5
150
160
170
180
190
200

MP visualization

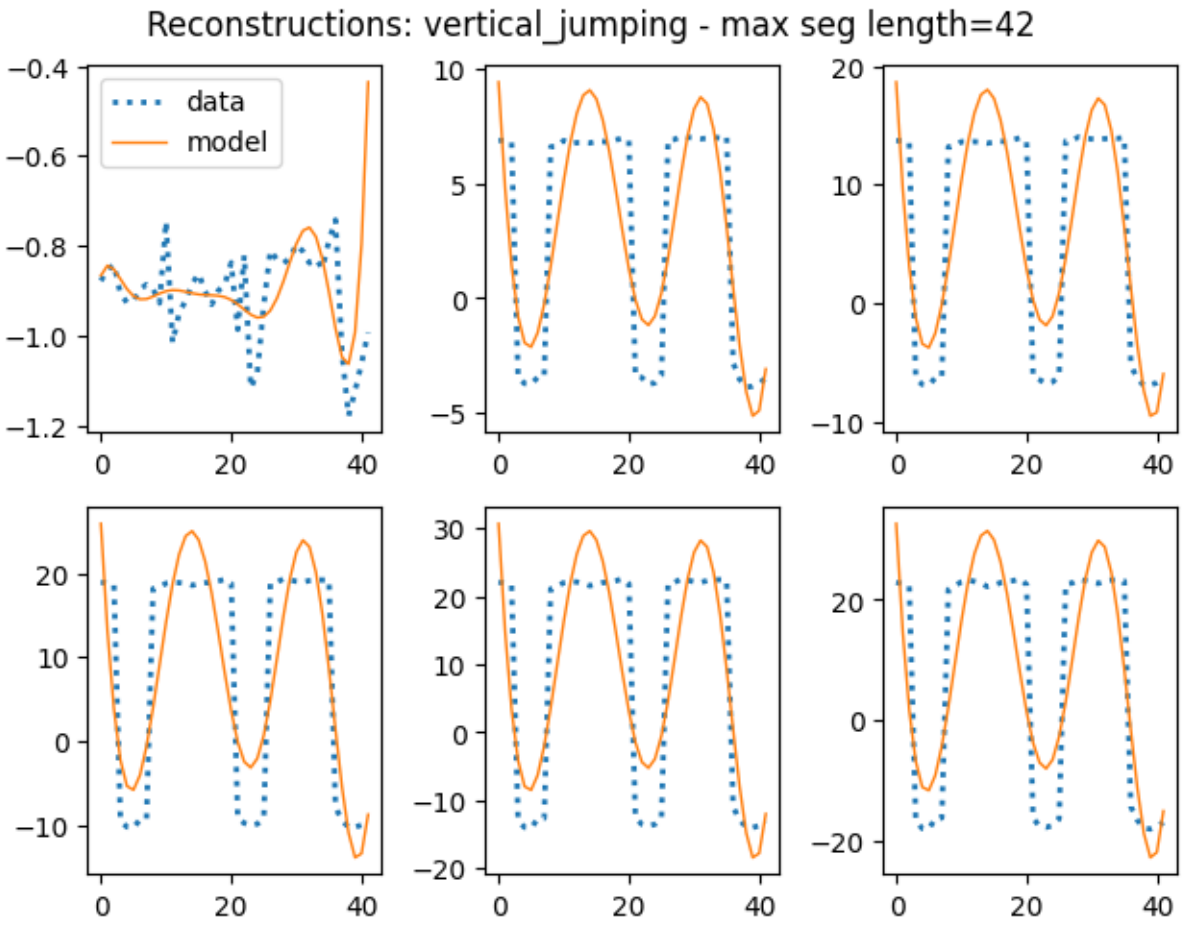


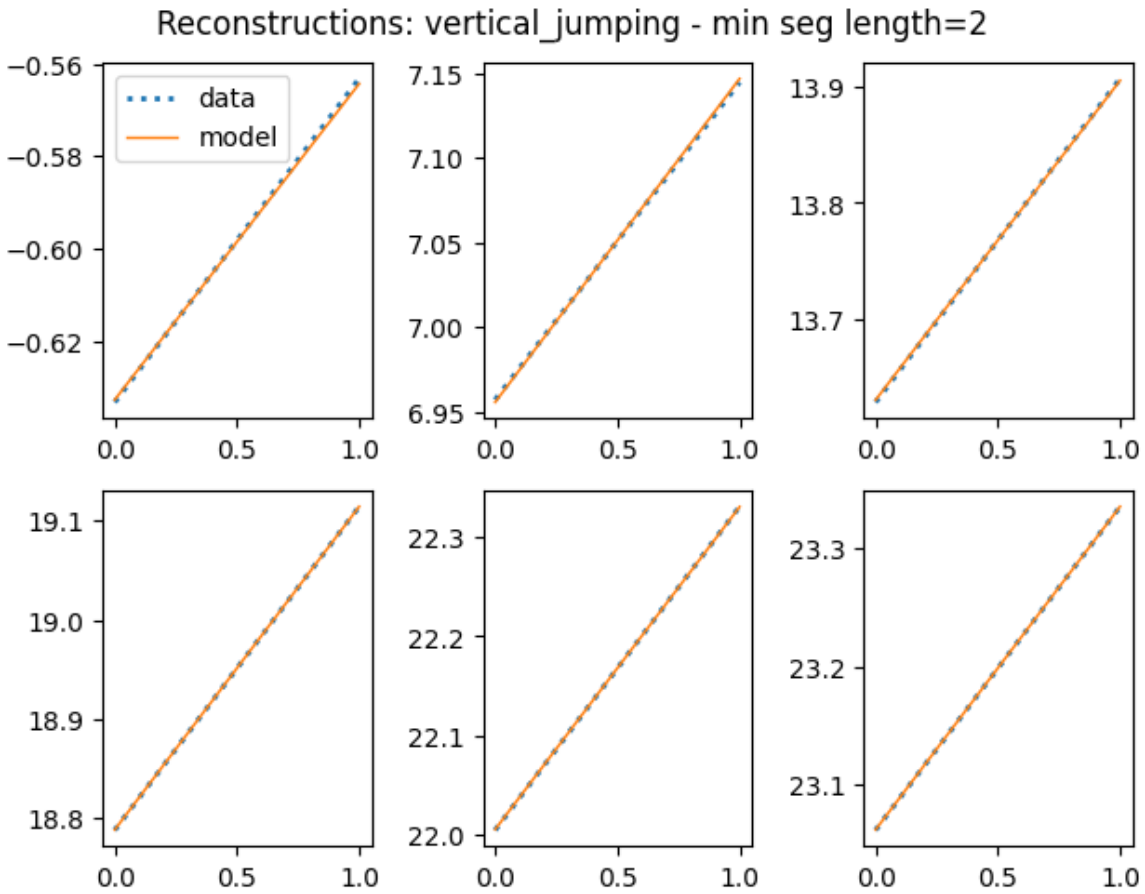


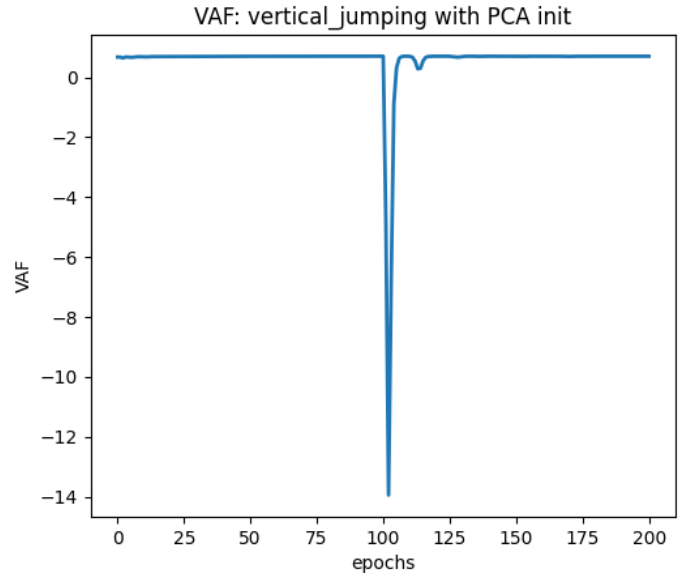
Weights distribution

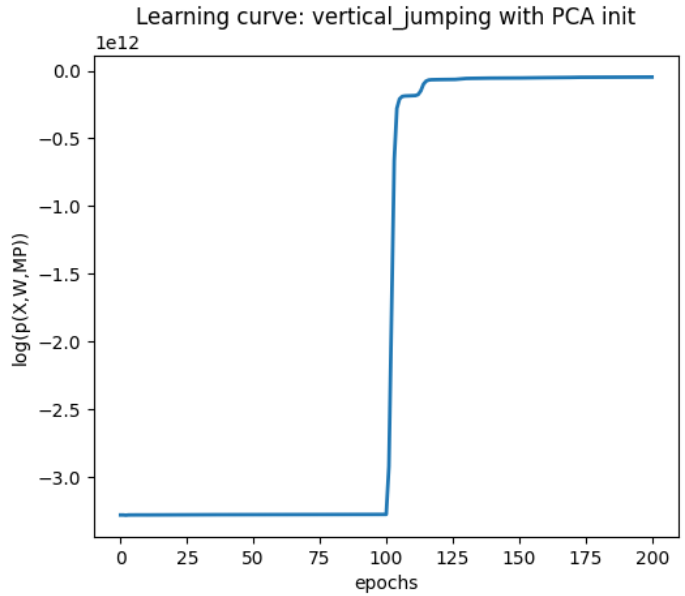


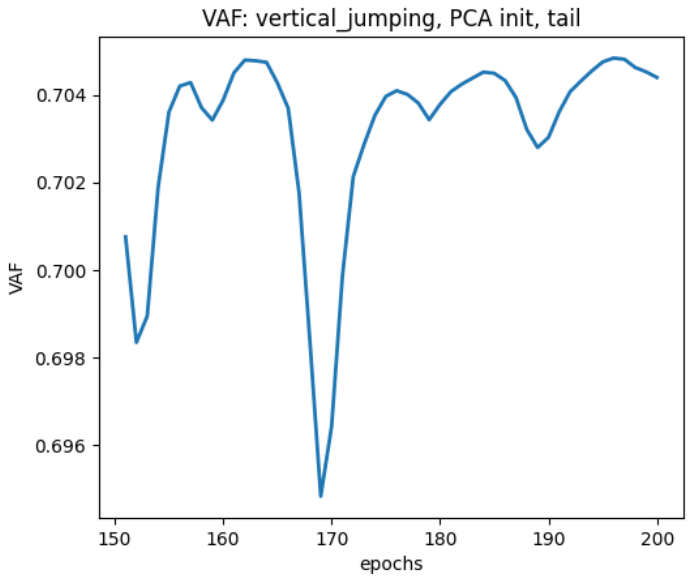
- Motion= **vertical_jumping** , 241 segments
- num_mp = 10 , num_t_points = 30
- VAF = 67%

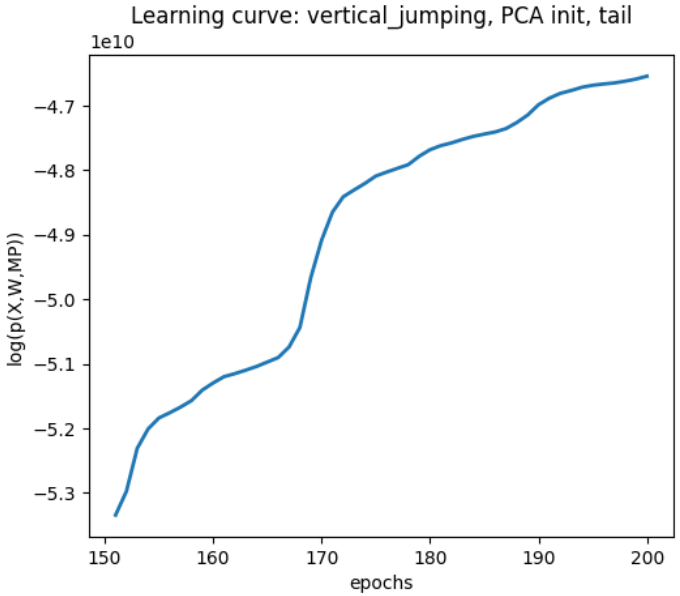




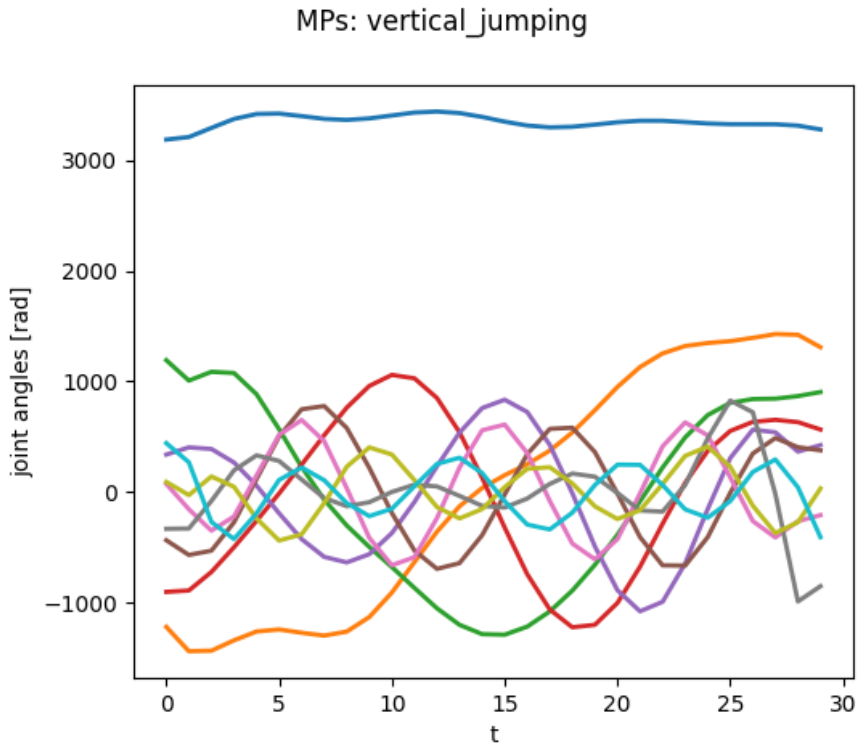

VAF: vertical_jumping with PCA init
VAF
epochs
0
−2
−4
−6
−8
−10
−12
−14
0
25
50
75
100
125
150
175
200

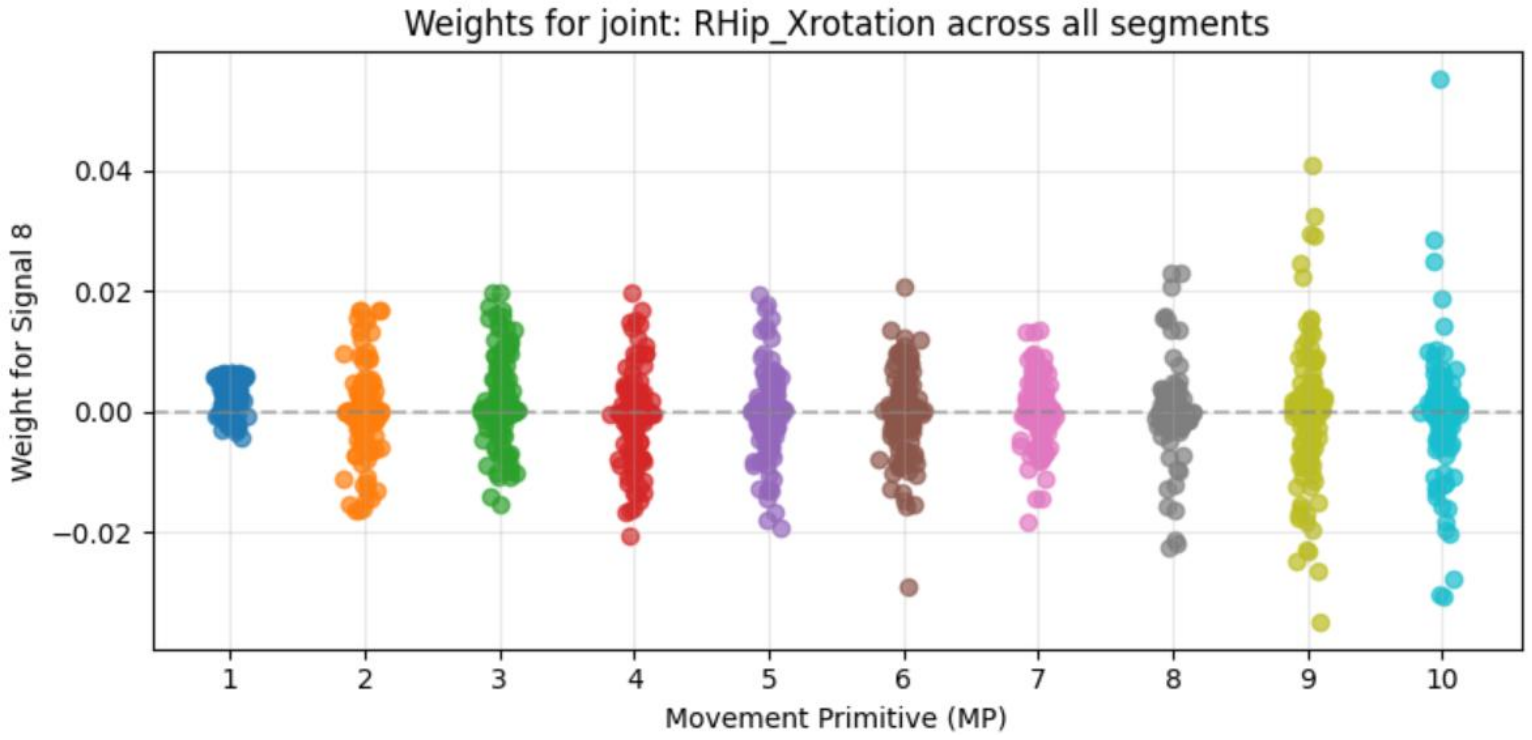

Learning curve: vertical_jumping with PCA init
1e12
log(p(X,W,MP))
epochs
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−3.0
0
25
50
75
100
125
150
175
200


VAF: vertical_jumping, PCA init, tail
VAF
epochs
0.704
0.702
0.700
0.698
0.696
150
160
170
180
190
200


Learning curve: vertical_jumping, PCA init, tail
1e10
log(p(X,W,MP))
epochs
−4.7
−4.8
−4.9
−5.0
−5.1
−5.2
−5.3
150
160
170
180
190
200

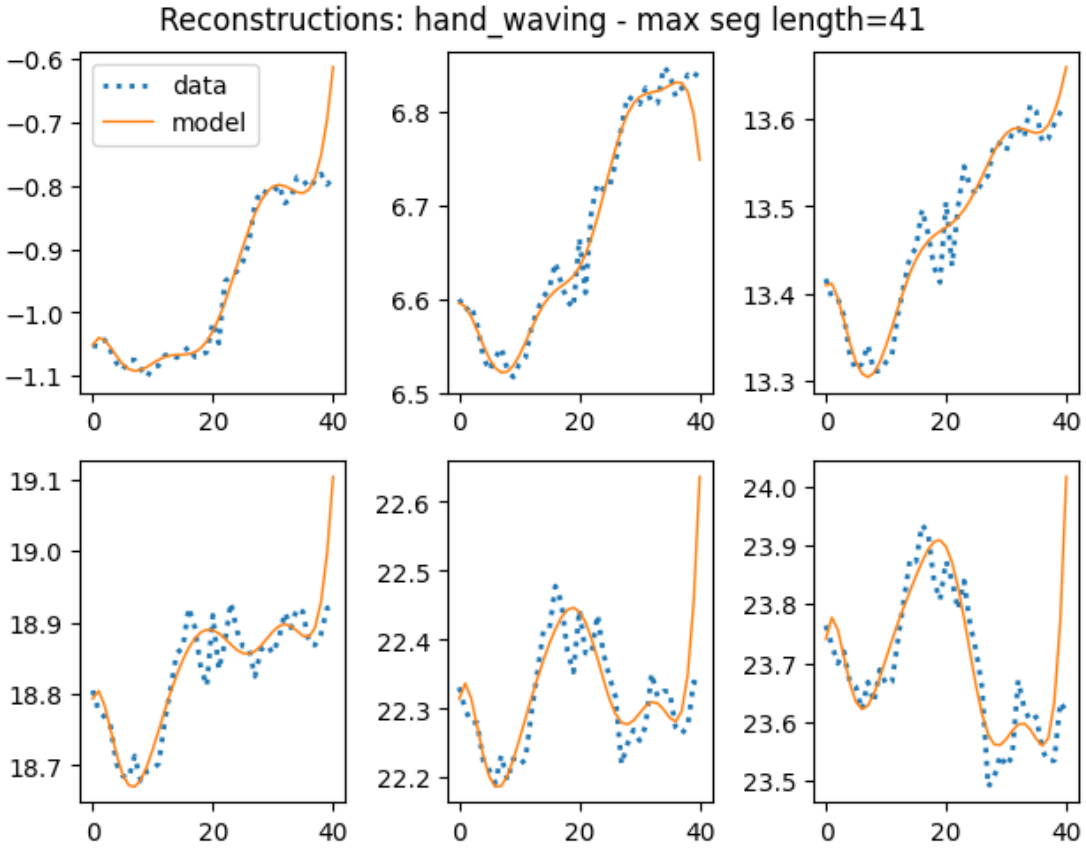
MP visualization

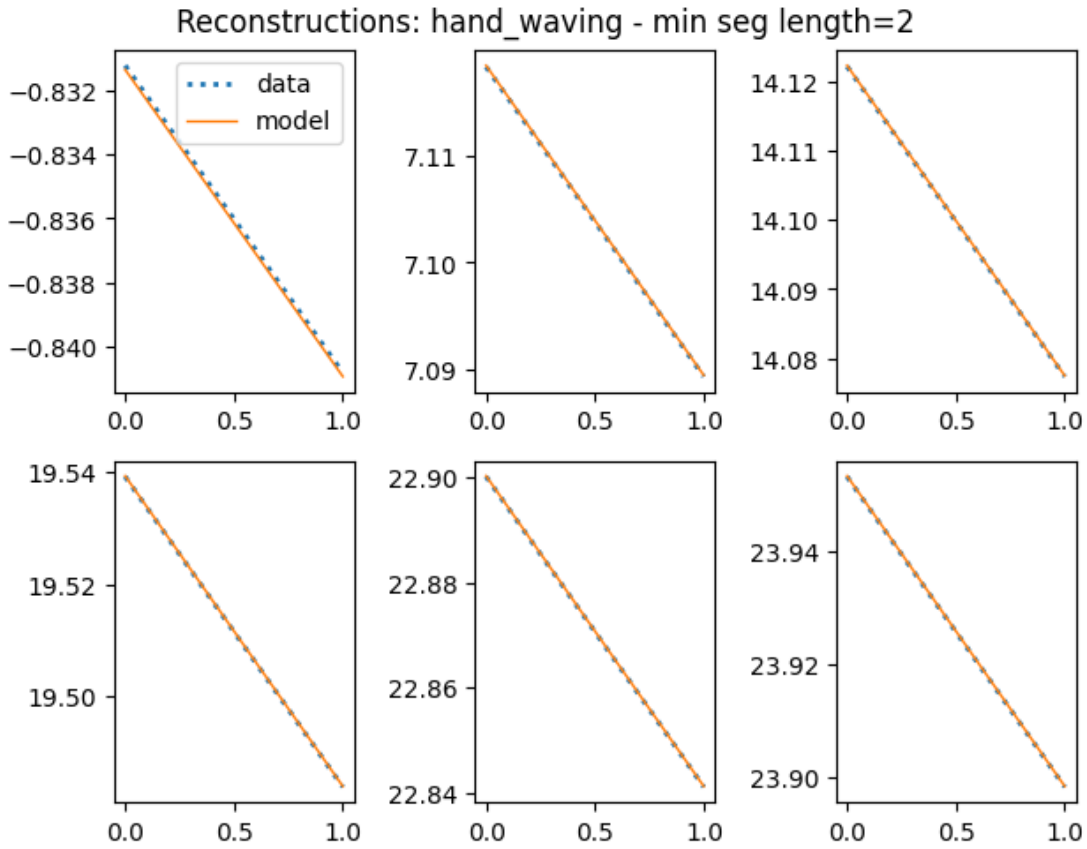


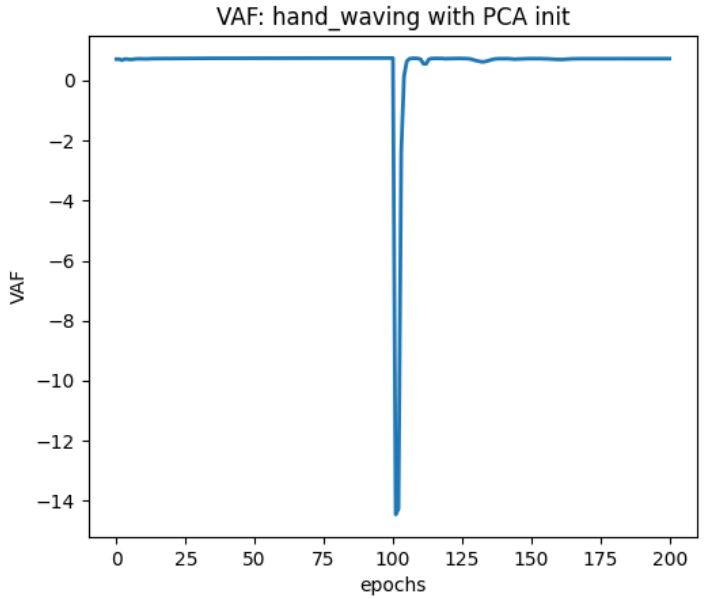


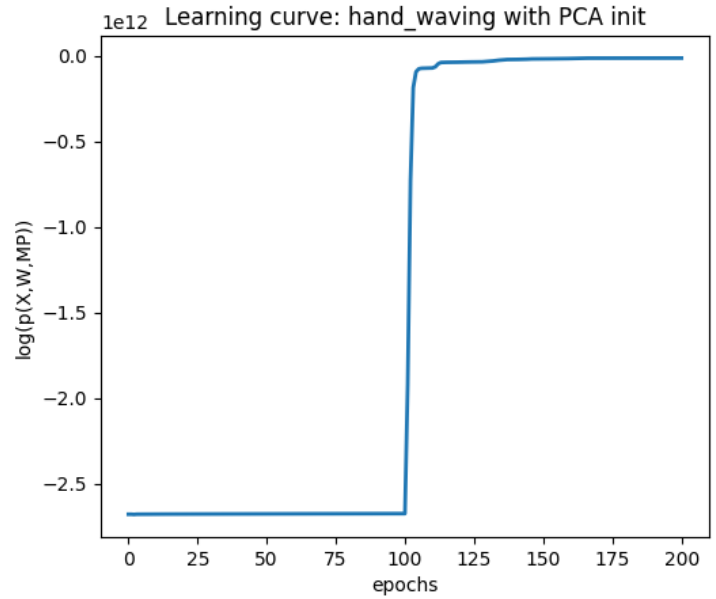
Weights distribution

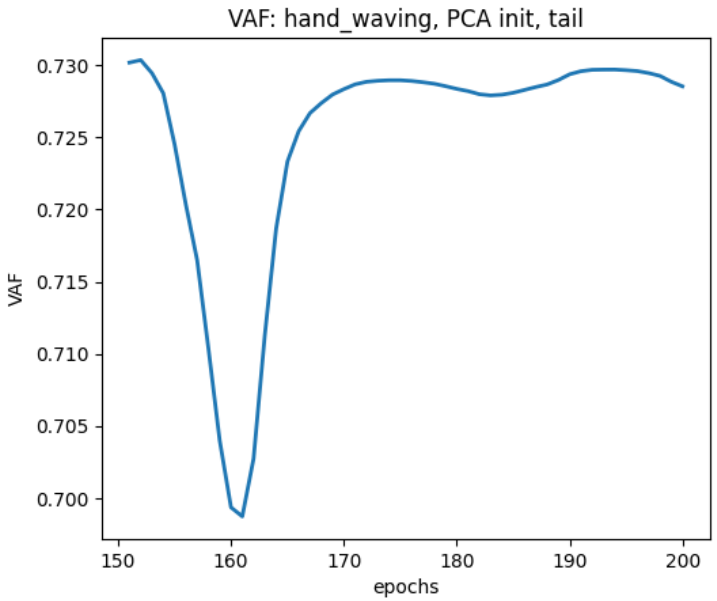
- Motion= **hand_waving** , 224 segments
- num_mp = 10 , num_t_points = 30
- VAF = 70%

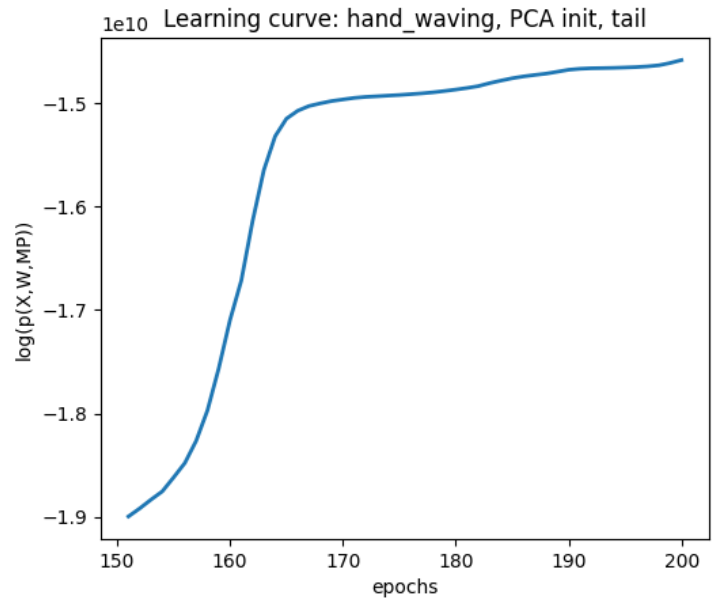




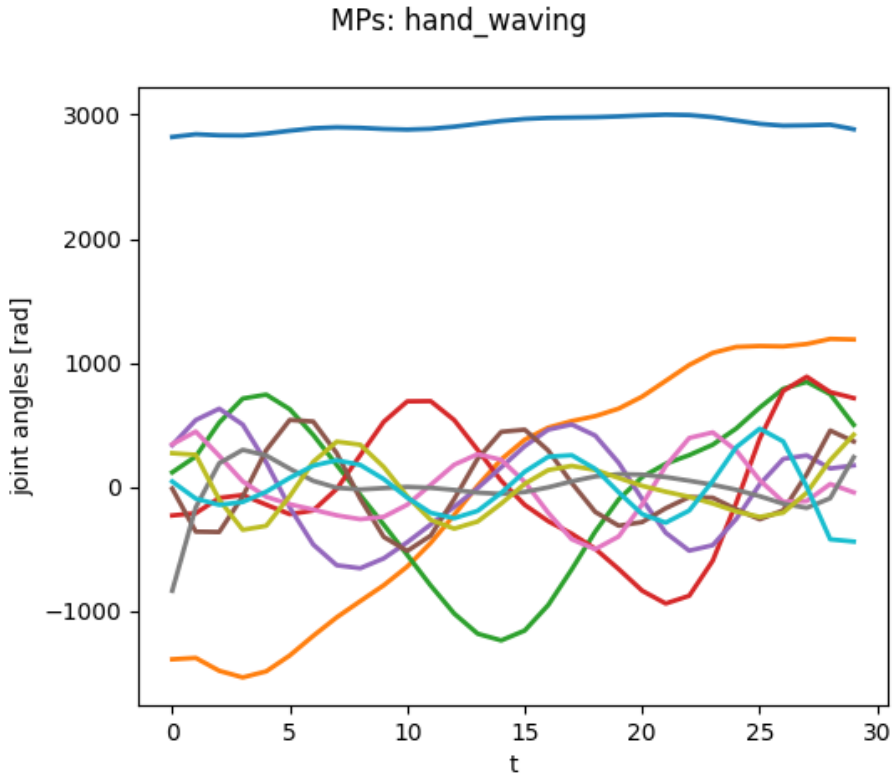

VAF: hand_waving with PCA init
VAF
epochs
0
−2
−4
−6
−8
−10
−12
−14
0
25
50
75
100
125
150
175
200

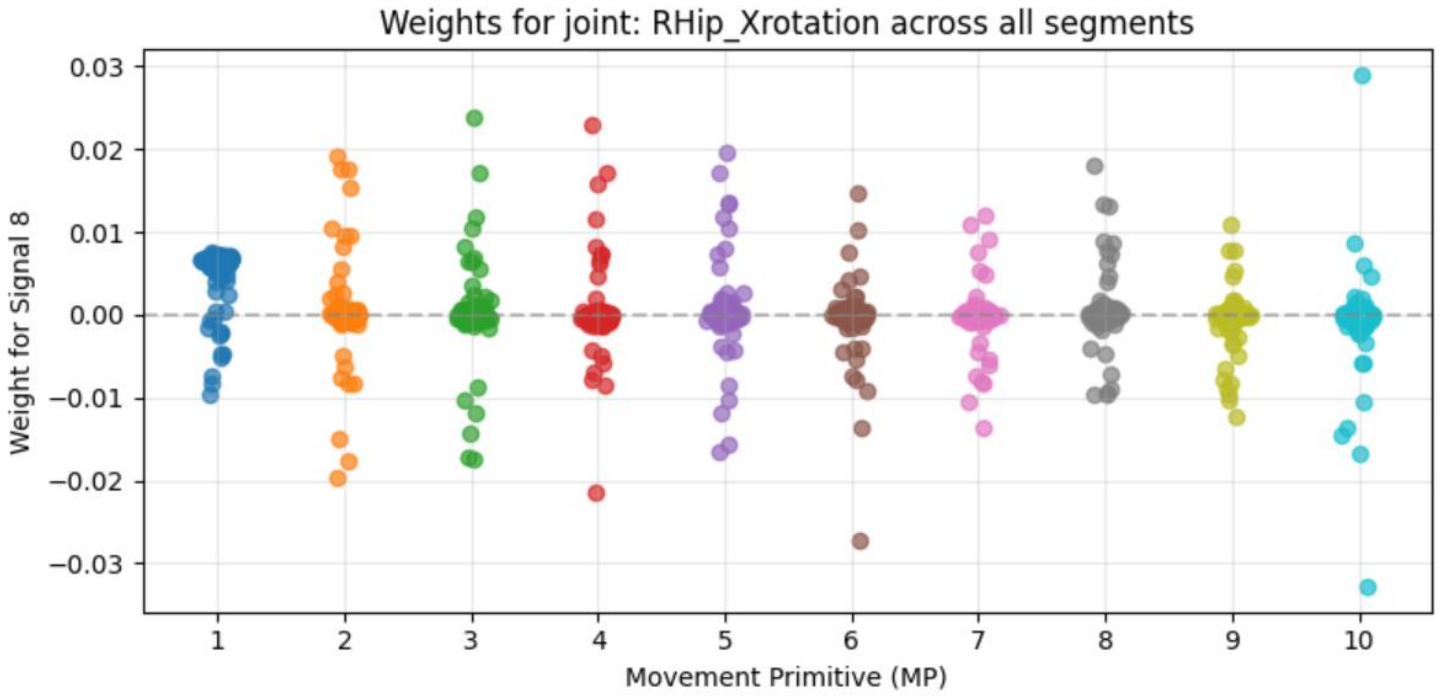

Learning curve: hand_waving with PCA init
1e12
log(p(X,W,MP))
epochs
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
0
25
50
75
100
125
150
175
200


VAF: hand_waving, PCA init, tail
VAF
epochs
0.730
0.725
0.720
0.715
0.710
0.705
0.700
150
160
170
180
190
200


Learning curve: hand_waving, PCA init, tail
1e10
log(p(X,W,MP))
epochs
−1.5
−1.6
−1.7
−1.8
−1.9
150
160
170
180
190
200

MP visualization





Weights distribution

Discussion

1. How can we create a unified pipeline that incorporates all of these models?
2. How to improve VAF?
3. Do we need movement reconstruction?
4. Different num of MPs and time_points?

Specific questions:

- Why using Laplace for posterior?
- Unrealistic range of joint angles for MPs

End