

Supplementary: Guided Attention for Interpretable Motion Captioning

Karim Radouane^{1,2}
karimradouane39@gmail.com

Julien Lagarde^{1,3}
julien.lagarde@umontpellier.fr

Sylvie Ranwez^{1,2}
sylvie.ranwez@mines-ales.fr

Andon Tchechmedjiev^{1,2}
andon.tchechmedjiev@mines-ales.fr

¹ EuroMov Digital Health in Motion

² IMT Mines Ales

³ University of Montpellier

This supplementary provides more details on the method implementation, and more visualization for global evaluation of interpretability. Furthermore, we discuss the effectiveness of architecture design. All following analysis are conducted on the test set.

Animations. For illustration, visual animations are included with this document. The transparency level of the gold box represents the temporal attention variation for each predicted *motion word* selected based on *adaptive attention*. We note that grammatical errors mainly stem from the datasets themselves, which contain valid action descriptions but sometimes with incorrect language structure.

1 Introduction

Our approach is **focusing on interpretability** while ameliorating motion captioning performance. This comes with additional challenging question on accurate methods for interpretability evaluations. To address this question, a first attempt is to draw multiple visualizations. However, for a global evaluation on test set, this become infeasible. To overcome this limitation, in addition, a simple solution, yet effective, is to display histogram and density distributions for attention weights across all test set instead of just sample wise visualizations.

2 Main contributions and motivations

The architectural design is primarily intended to be interpretable, allowing for the explanation of learned spatial, temporal, and adaptive attention weights. Designing an efficient architecture while maintaining interpretability can be very challenging, but has several advantages beyond focusing solely on increasing accuracy metrics. In addition to ensuring a reliable model, we can leverage the interpretability provided by attention mechanisms to

extract other semantic motion information: action localization, body part and motion word identification. Let’s recall the main novel contributions of our paper in this context:

- Interpretable architecture design.
- Supervision of adaptive and spatial attention.
- Effective tools for global interpretability evaluation.

Consequently, regarding each contribution aspect, we will show the concrete effectiveness of associated theoretical formulations.

3 Datasets

We use the two commonly used benchmarks KIT-MLD and Human ML3D with the following statistical details:

Subset	Number	Train	Test	Val.
KIT-ML-aug	motions	4886	830	300
	samples	10408	1660	636
HML3D-aug	motions	22068	4160	1386
	samples	66734	12558	4186

Table 1: Data splits, for KIT and Human ML3D after augmentation (aug).

4 Ground Truth generation for supervision

Predefined dictionary. We manually define a dictionary based on representative words in the dataset describing different motion characteristics. Intentionally the dictionary doesn’t cover all datasets actions with their synonyms, we want the model to be able to generalize to remaining unsupervised words for their spatial and gate attention. We will see later that the model effectively converges for this intended behavior.

Category	Words	Body part
Trajectory	circle, circuit, clockwise, anticlockwise, forward, backward	Root
Local motion	open, waves, wipe, throw, punch, pick, boxing, clean, swipe, catch, handstand, draw	Arms
	kick, stomp, lift, kneel, squat, squad, stand, stumble, rotate	Legs
	bend, bow	Torso
Connection words	is, the, of, his, her, its, on, their	-
Subject	a, person, human, man	-

Table 2: Predefined dictionary for both datasets.

During training, the words in Table 2, and targets words, are stemmed to find correspondence for spatial weight supervision.

Spatial attention supervision. The ground truth spatial attention weights α_{ti} are generated based on the predefined dictionary and it’s same for all frames, the temporal attention is the responsible for temporal filtering.

Adaptive attention supervision. The ground truth β_t is generated based on the Part Of Speech (POS) tagging.

5 Hyperparameters selection

We run experiments for different values of $(\lambda_{spat}, \lambda_{adapt})$. The quantitative results are reported in Table 3.

Dataset	λ_{spat}	λ_{adapt}	BLEU@1	BLEU@4	CIDEr	ROUGE _L	BERTScore
KIT-ML	0	0	57.3	23.6	109.9	57.8	41.1
	0	3	56.3	22.5	108.4	56.5	39.8
	1	3	57.6	23.5	102.6	57.2	40.1
	2	3	58.4	24.4	112.1	58.3	41.2
	3	5	57.6	23.7	105.7	57.5	40.9
	5	5	56.5	22.0	99.4	56.8	39.9
HML3D	0	0	69.3	24.0	58.8	54.8	38.7
	0	3	69.9	25.0	61.6	55.3	40.3
	0.1	3	69.5	23.8	58.7	55.0	38.9
	0.25	3	68.7	23.8	59.7	54.7	39.3
	0.5	3	68.8	23.8	60.0	55.0	38.6
	1	3	68.7	23.7	58.2	54.6	39.0
	2	3	69.2	24.4	61.7	55.0	40.3
	3	3	68.3	23.2	56.5	54.5	37.1

Table 3: Spat+adapt supervision impact w.r.t each corresponding weights.

6 Architecture compounds effectiveness

We aim in the following visualizations to demonstrate the global effectiveness of architecture design of each compound :

- Functionality of gating mechanism.
- Impact of Part based motion encoding.
- Spatio-temporal attention blocks.

Gating mechanism The gate variable β allows the model to use or not the motion information given the word time step. To visualize this internal process of switching between motion and language, we display predictions for the best model on KIT-ML (results on HumanML3D were shown in the paper). As we see in the following Table, the context vector ($\beta = 1$) is successfully used for all motion characteristics, *action, speed, body parts, trajectory, direction...* Particularly, we note that the end token $\langle e_{os} \rangle$ is also motion related, as outputting this word depends on the end of the relevant human motion range.

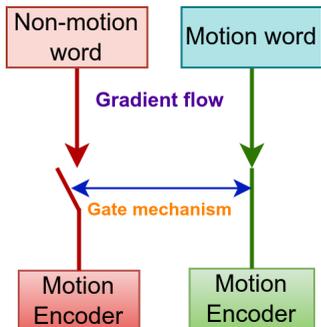


Figure 1: Illustration of our gating mechanism during training. This mechanism prevent the decoder from attending to motion for non-motion word. Consequently the motion encoder is prevented from receiving important gradients updates for non motion words.

Spatial+adapt attention supervision [KIT-ML] We show comparison of Spatio-temporal attention maps and text generated between the case of supervision and w/o supervision:

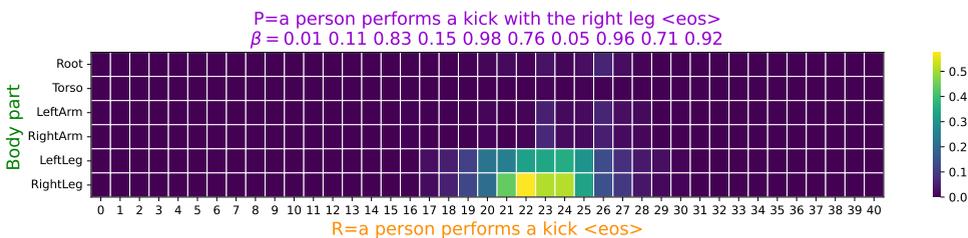


Figure 2: With supervision KIT-(2,3) (action range [19,28]/right kick).

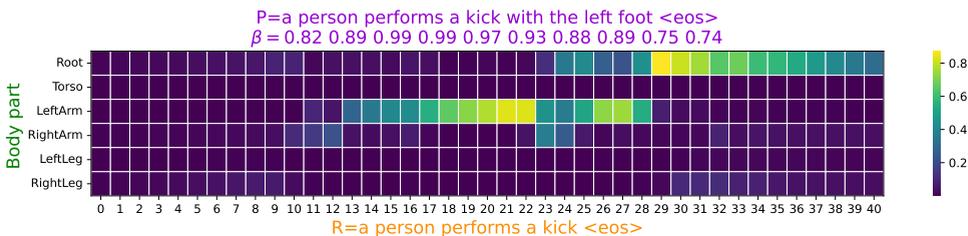


Figure 3: Without supervision KIT-(0,0) (action range [19,27]/right kick).

As we see in the case of supervision (Fig.2) the part were correctly identified and perfectly localized in the range [20,26] with corresponding manually identified range [19,28] and small β values are associated with non-motion words. Without supervision (Fig.3), the model focuses on irrelevant part and consequently the range of action was not precisely localized. Additionally the β values are high for all kind of words.

We visualize more samples (Fig.7) with Spatial+adapt supervision. Temporal range is mentioned for comparison, even if action localization wasn't the main focus in captioning task, the model was able to learn implicitly a temporal location through the temporal Gaus-

sian attention mechanism.

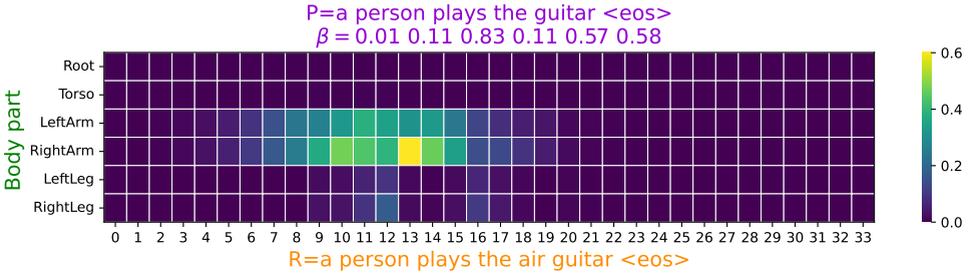


Figure 4: Play (action range [10,20]).

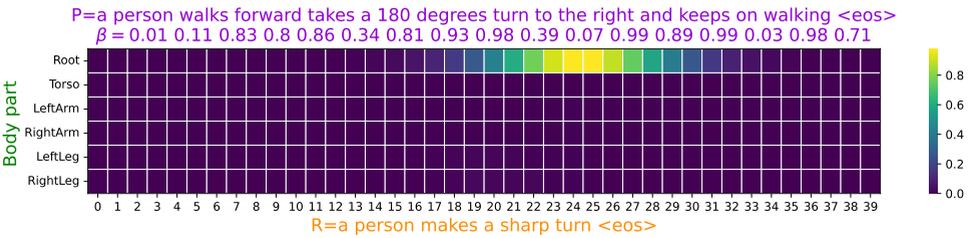


Figure 5: Turn (action range [22,27]).

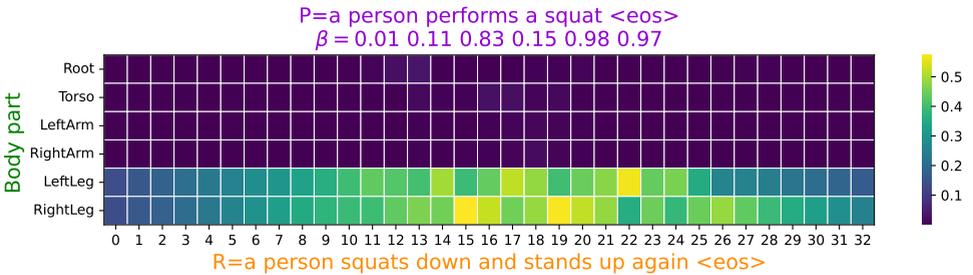


Figure 6: Squat (action range [10,28]).

Figure 7: Spatio-temporal attention for different motion words on KIT-ML.

Trajectory and global motion The attention was supervised only for words describing trajectory, but the model generalize successfully to motion words highly depending on global trajectory. This result on maximum attention distributed toward the *Root* body part, as we see in Figure 8.

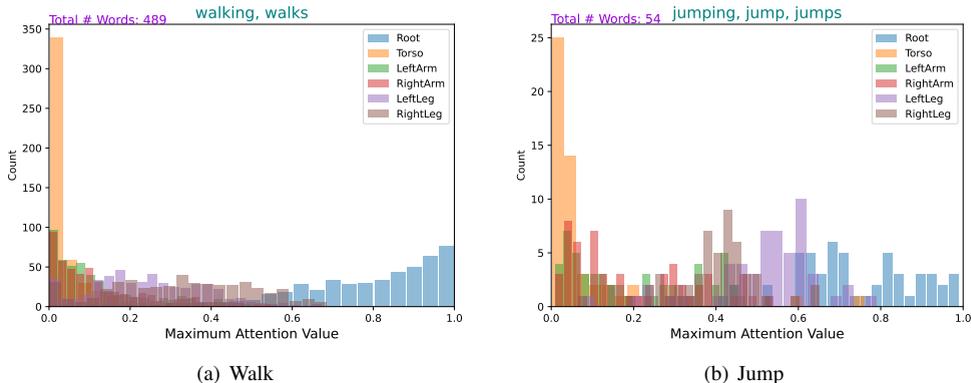
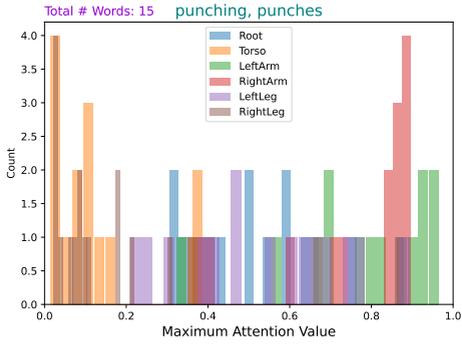


Figure 8: [KIT-(2,3)]: Body part distribution (spat+adapt).

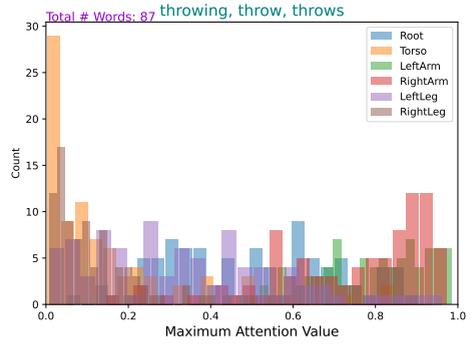
6.1 Part based encoding & spatio-temporal attention

As mentioned in the paper, our architecture design could be sufficient in learning a correct spatial attention maps using larger dataset with rich semantic descriptions. For demonstration, we will use the model with no spatial supervision, to show that part based encoding and spatio-temporal can work solely and correctly together for focusing on relevant body parts w.r.t to the associated generated motion word. To this purpose, we propose to display the histogram distribution of temporal maximum attention weights for each body part over all test set and given a different motion words. This allows for an effective global evaluation of interpretability over all test set.

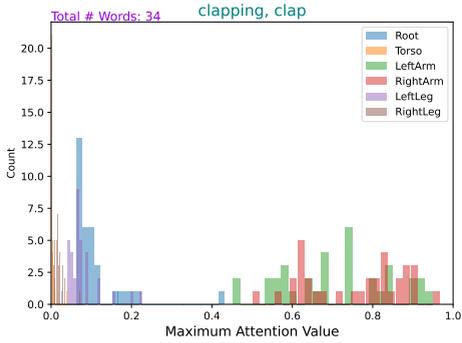
Histograms In the following, we display the body parts histogram distribution across the test set for different motion words on the model with *no spatial supervision* as demonstration for the effectiveness in finding relevant parts to focus on using our interpretable architecture design that includes part-based encoding along with spatio-temporal attention. This is only in the case of the larger dataset HumanML3D. The KIT-ML small dataset still requires spatial supervision to help the architecture focusing on relevant part, as the vocabulary and its size are limited. As demonstrated in all following Figures, depending on the motion word, arms-based/legs-based actions, and particularly some motions with an emphasis on Torso body part.



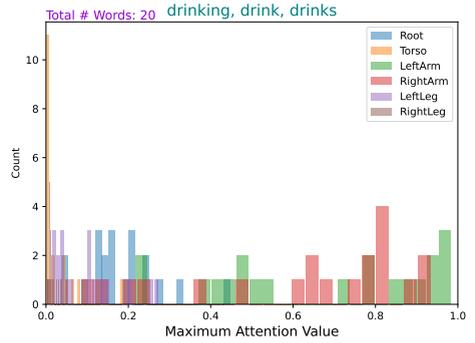
(a) Punch



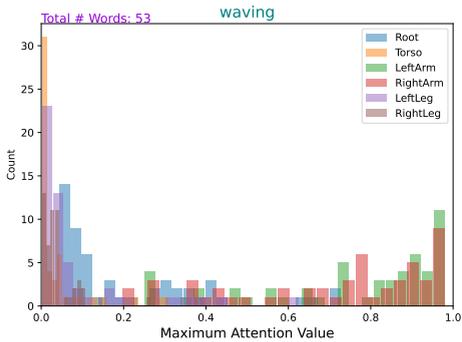
(b) Throw



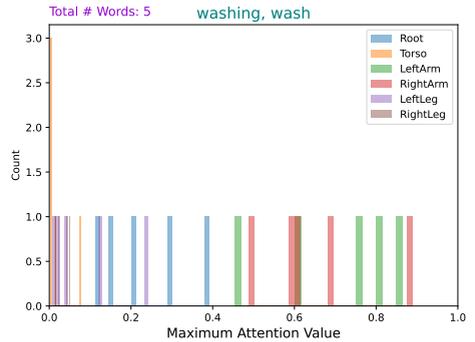
(c) Clap



(d) Drink

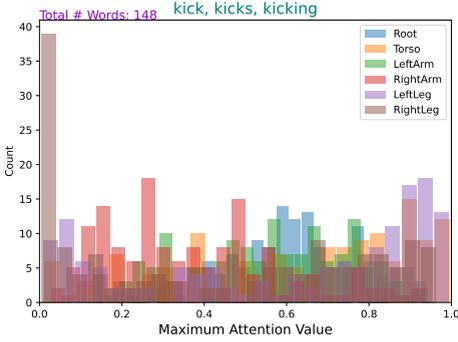


(e) Wave

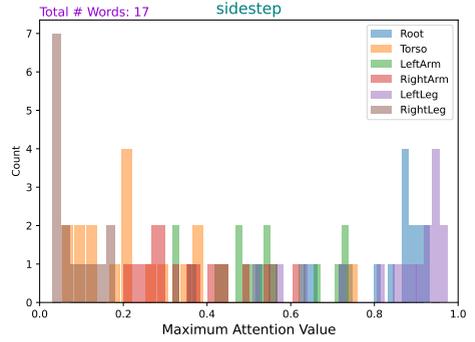


(f) Wash

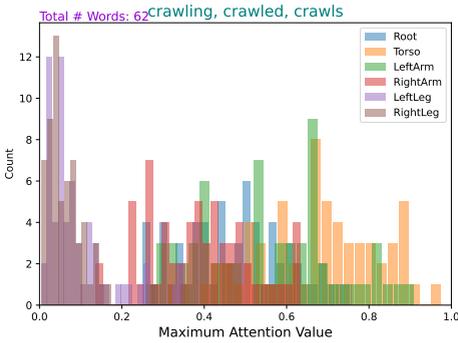
Figure 9: Histogram generated on the HML3D with the config (0,3).



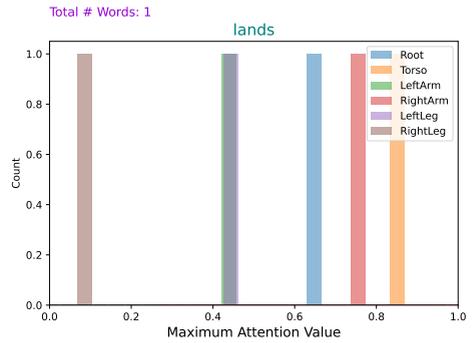
(a) Kick



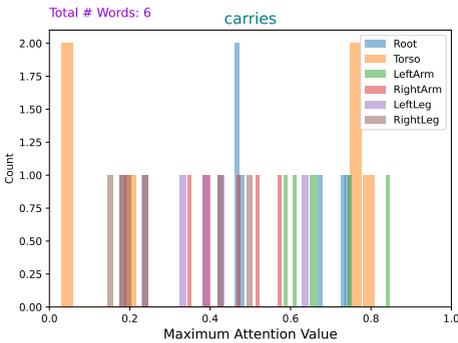
(b) Sidestep



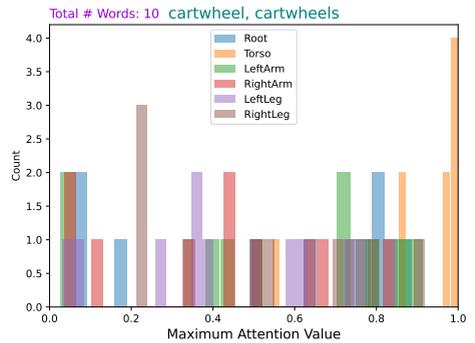
(c) Crawl



(d) Land

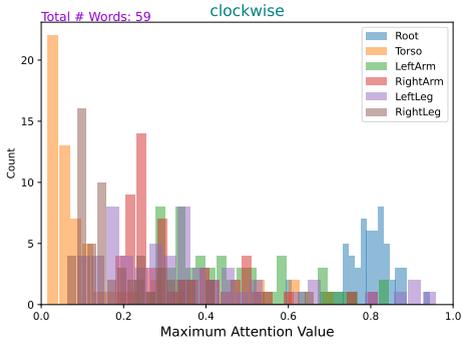


(e) Carries

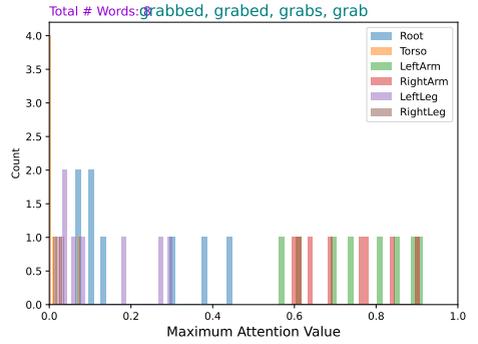


(f) Cartwheel

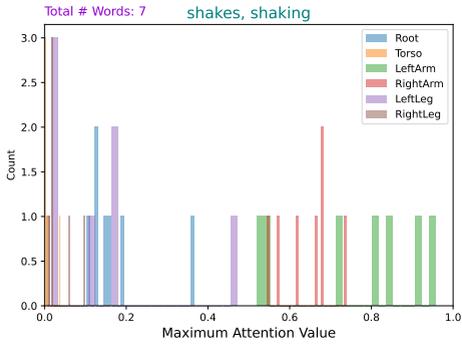
Figure 10: Histogram generated on HML3D with the config (0,3).



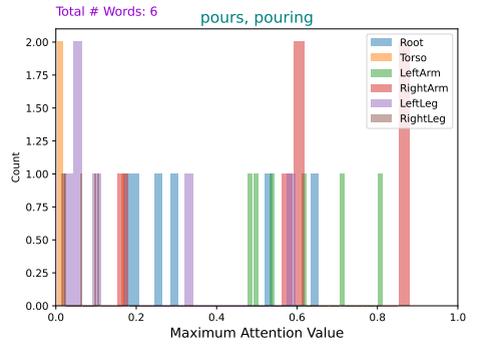
(a) Clockwise



(b) Grab



(c) Shake



(d) Pour

Figure 11: Histogram generated on HML3D with the config (0,3).

Spatio-temporal attention maps In this part, we display attention maps for some interesting words for HML3D (0-3) /adapt. In the case of the model without spatial supervision, we have found that the model performs a correct attention focus. When an action is performed using right leg/arm, the model focuses correctly on the corresponding parts. Moreover, for actions performed with both arms/legs, the model focus on both parts. For all cases, body part words (left/right/both) are always accurately identified into the generated text. These observations are common across different representative samples (from different actions).

Attention map: bi-part based human motion



Figure 12: Raises.



Figure 13: Lowers.

Attention map: single-part based human motion

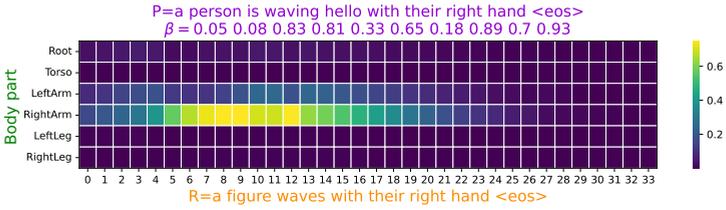


Figure 14: Waving.



Figure 15: Opens.

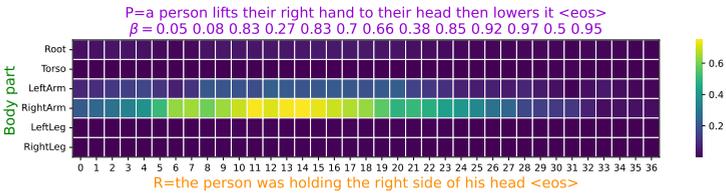


Figure 16: Lifts.

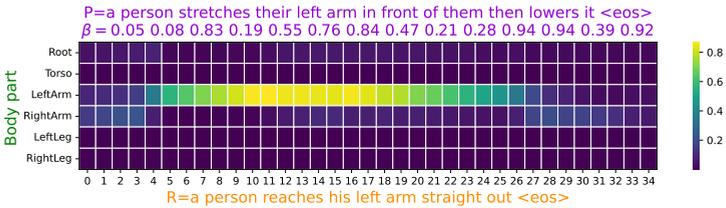


Figure 17: Stretches.

7 Network architecture

Figures 18, 19 give the architecture details for each dataset. (The design is similar for both datasets, the differentiation is only in layer sizes).

```
(enc_pose): Encoder(
  (dropout): Dropout(p=0.5, inplace=False)
  (six_layers): ModuleList(
    (0): Linear(in_features=3, out_features=256, bias=True)
    (1): Linear(in_features=18, out_features=256, bias=True)
    (2): Linear(in_features=12, out_features=256, bias=True)
    (3): Linear(in_features=12, out_features=256, bias=True)
    (4): Linear(in_features=12, out_features=256, bias=True)
    (5): Linear(in_features=12, out_features=256, bias=True)
  )
  (six_layers_2): ModuleList(
    (0): Linear(in_features=256, out_features=128, bias=True)
    (1): Linear(in_features=256, out_features=128, bias=True)
    (2): Linear(in_features=256, out_features=128, bias=True)
    (3): Linear(in_features=256, out_features=128, bias=True)
    (4): Linear(in_features=256, out_features=128, bias=True)
    (5): Linear(in_features=256, out_features=128, bias=True)
  )
  (six_layers_v): ModuleList(
    (0): Linear(in_features=3, out_features=256, bias=True)
    (1): Linear(in_features=18, out_features=256, bias=True)
    (2): Linear(in_features=12, out_features=256, bias=True)
    (3): Linear(in_features=12, out_features=256, bias=True)
    (4): Linear(in_features=12, out_features=256, bias=True)
    (5): Linear(in_features=12, out_features=256, bias=True)
  )
  (six_layers_v2): ModuleList(
    (0): Linear(in_features=256, out_features=128, bias=True)
    (1): Linear(in_features=256, out_features=128, bias=True)
    (2): Linear(in_features=256, out_features=128, bias=True)
    (3): Linear(in_features=256, out_features=128, bias=True)
    (4): Linear(in_features=256, out_features=128, bias=True)
    (5): Linear(in_features=256, out_features=128, bias=True)
  )
)
(dec): Decoder(
  (dropout): Dropout(p=0.5, inplace=False)
  (embedding): Embedding(3605, 128)
  (top_lstm): LSTM(256, 256)
  (bottom_lstm): LSTM(128, 256)
)
(mixture_feat): Linear(in_features=640, out_features=640, bias=True)
(final_layer): Linear(in_features=640, out_features=3605, bias=True)
(feats_extract_x): Linear(in_features=1536, out_features=256, bias=True)
(feats_extract_hdec): Linear(in_features=256, out_features=256, bias=True)
(feats_extract_henc): Linear(in_features=256, out_features=256, bias=True)
(spatial_att): Linear(in_features=256, out_features=6, bias=True)
(gate_var): Linear(in_features=256, out_features=1, bias=True)
(adapt_layer): Linear(in_features=128, out_features=1, bias=True)
(ctproject): Linear(in_features=256, out_features=256, bias=True)
(htproject): Linear(in_features=256, out_features=256, bias=True)
(feats_extract_g): Linear(in_features=1536, out_features=256, bias=True)
(tempfeat_extract_hdec): Linear(in_features=256, out_features=256, bias=True)
(temporal_att): Linear(in_features=256, out_features=1, bias=True)
```

Figure 18: Architecture details for HML3D.

```

(enc_pose): Encoder(
  (dropout): Dropout(p=0.5, inplace=False)
  (six_layers): ModuleList(
    (0): Linear(in_features=3, out_features=128, bias=True)
    (1): Linear(in_features=12, out_features=128, bias=True)
    (2): Linear(in_features=9, out_features=128, bias=True)
    (3): Linear(in_features=9, out_features=128, bias=True)
    (4): Linear(in_features=15, out_features=128, bias=True)
    (5): Linear(in_features=15, out_features=128, bias=True)
  )
  (six_layers_2): ModuleList(
    (0): Linear(in_features=128, out_features=64, bias=True)
    (1): Linear(in_features=128, out_features=64, bias=True)
    (2): Linear(in_features=128, out_features=64, bias=True)
    (3): Linear(in_features=128, out_features=64, bias=True)
    (4): Linear(in_features=128, out_features=64, bias=True)
    (5): Linear(in_features=128, out_features=64, bias=True)
  )
  (six_layers_v): ModuleList(
    (0): Linear(in_features=3, out_features=128, bias=True)
    (1): Linear(in_features=12, out_features=128, bias=True)
    (2): Linear(in_features=9, out_features=128, bias=True)
    (3): Linear(in_features=9, out_features=128, bias=True)
    (4): Linear(in_features=15, out_features=128, bias=True)
    (5): Linear(in_features=15, out_features=128, bias=True)
  )
  (six_layers_v2): ModuleList(
    (0): Linear(in_features=128, out_features=64, bias=True)
    (1): Linear(in_features=128, out_features=64, bias=True)
    (2): Linear(in_features=128, out_features=64, bias=True)
    (3): Linear(in_features=128, out_features=64, bias=True)
    (4): Linear(in_features=128, out_features=64, bias=True)
    (5): Linear(in_features=128, out_features=64, bias=True)
  )
)
(dec): Decoder(
  (dropout): Dropout(p=0.5, inplace=False)
  (embedding): Embedding(878, 64)
  (top_lstm): LSTM(128, 128)
  (bottom_lstm): LSTM(64, 128)
)
(mixture_feat): Linear(in_features=320, out_features=320, bias=True)
(final_layer): Linear(in_features=320, out_features=878, bias=True)
(feats_extract_x): Linear(in_features=768, out_features=128, bias=True)
(feats_extract_hdec): Linear(in_features=128, out_features=128, bias=True)
(feats_extract_henc): Linear(in_features=128, out_features=128, bias=True)
(spatial_att): Linear(in_features=128, out_features=6, bias=True)
(gate_var): Linear(in_features=128, out_features=1, bias=True)
(adapt_layer): Linear(in_features=64, out_features=1, bias=True)
(ctproject): Linear(in_features=128, out_features=128, bias=True)
(htproject): Linear(in_features=128, out_features=128, bias=True)
(feats_extract_g): Linear(in_features=768, out_features=128, bias=True)
(temp_feats_extract_hdec): Linear(in_features=128, out_features=128, bias=True)
(temporal_att): Linear(in_features=128, out_features=1, bias=True)

```

Figure 19: Architecture details for KIT-ML.

β (gate)	Prediction
1	waves waves waving with both hands <eos>
Adaptive	the person is waving both hands <eos>
REF	the person is waving both hands <eos>
1	walks walks slowly <eos>
Adaptive	a person walks slowly <eos>
REF	a person walks forwards quite slowly <eos>
1	kicking kicking kicking with left leg <eos>
Adaptive	a person kicks something with its left foot <eos>
REF	a human kicks something with his left foot <eos>
1	turns walking and turning on walking <eos>
Adaptive	a person turns on his right foot <eos>
REF	a human turns abruptly <eos>
1	jumping jumps forward <eos>
Adaptive	a person jumps with both legs <eos>
REF	a person jumping 1 step <eos>
1	running running running <eos>
Adaptive	a person runs <eos>
REF	a person runs forward <eos>
1	throws throws throwing <eos>
Adaptive	a person performs a right throwing <eos>
REF	a person throws a ball <eos>

Table 4: Comparison of the prediction when setting $\beta = 1$ and adaptive on KIT-ML (Spat+adapt (2,3)).