Ham2Pose: Animating Sign Language Notation into Pose Sequences — Supplementary Material

Rotem Shalev Arkushin
Reichman University
rotemroo@gmail.com

Amit Moryossef
Bar Ilan University
amitmoryossef@gmail.com

Ohad Fried
Reichman University
ofried@runi.ac.il

In this supplementary document, we present more evaluation results of our model, both qualitative and quantitative, as well as ablation studies of experimenting with different step amounts, shared vs. separate positional embeddings, hidden dimensions, and feedforward dimensions. In addition, we show the importance of weights in our loss function, by comparing it to the same loss without weights. Moreover, we provide hardware details, train & inference times, and more details about the model architecture and data. We also show a full derivation for our loss scaling factor.

A. Evaluation

A.1. Qualitative results

We supply additional qualitative results in Figure 4 and in the attached videos. We show examples of signs generated from a single HamNoSys sequence, and examples of sentences generated by concatenating a few signs generated one after the other. To generate a continuous sentence, we cut 20% off the end of each generated sign, and give the last frame of this sequence to the model as the reference start frame for the subsequent sign. As demonstrated, our model is able to generate correct movements, even when the ground truth pose detected by the pose estimation model is not full or incorrect, and is able to generate multiple signs one after the other to generate sentences.

A.2. Number of glyph occurrences vs. score

In Figure 1 we plot nDTW-MJE distance vs. the number of glyph occurrences of the rarest glyph per sequence, and observe that the more occurrences a glyph has, the lower the distance is. However, it is noisy, which suggests that the meaning of a glyph and the number of rare glyphs in a sequence may also affect the results.

A.3. Sequence length

We test the sequence length predictor separately from the full model using absolute difference between the real sequence length and the predicted one, and show a histogram of the differences in Figure 2. In addition, Figure 3 shows the percentage of error of the sequence length predictor. As demonstrated by the figures, the difference is low for most videos, both absolutely and relatively to the actual sequence length. Moreover, in both cases, as the sequence length difference error increases, the amount of samples with that difference decreases.

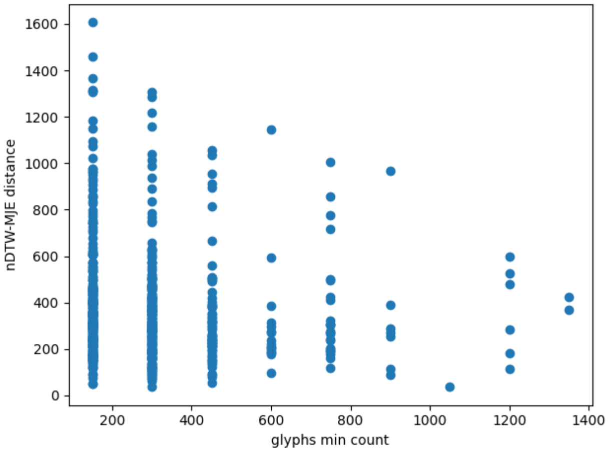

Figure 1. Minimal glyph occurrences number from sequence (split to bins) vs. *nDTW-MJE* distance.

B. Ablation studies

We conduct several ablation studies, experimenting with the amount of steps in the model, shared vs. separate positional embeddings for the text and pose, different hidden dimensions, and different feedforward dimensions. We present their results in Tables 1 to 4. For each of them, we train our model with the changed feature and perform the same evaluations explained in the main paper under *Distance Ranks*.

Change in step amounts: although the results of the 20-steps model are slightly better than the results of the 10-steps model, there is a trade-off between the number of steps

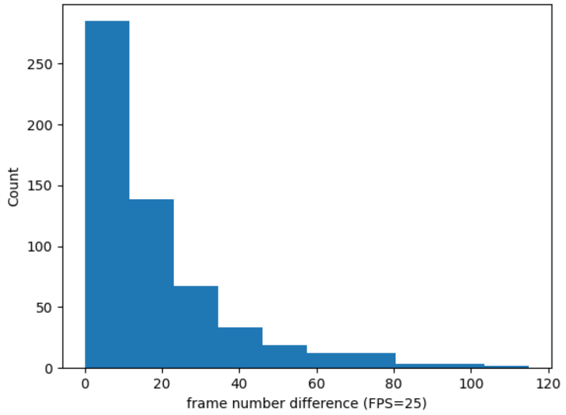

Figure 2. Absolute error between the real sequence length and the predicted one, in number of frames.

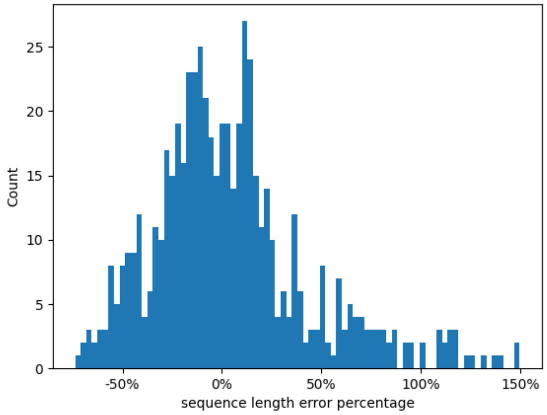

Figure 3. Signed percentage error, relative to the real sequence length, of the predicted sequence length. Negative values indicate that the predicted length is shorter than the real length; positive values indicate the opposite.

and the training time of the model, as can be seen in Table 5. Moreover, when looking at the results visually, while some results of the 20-steps model look better, others look worse. Therefore, since the improvement in the quality of the results is not drastic, we prefer to use less steps. Finally, the 5-steps model generates results that are worse both quantitatively when looking at the prediction reference, and quantitatively, hence we chose to use 10 steps in our model. We show qualitative results for each number of steps in the attached video.

Shared vs. separate positional embedding: as can be seen in Table 2, having two separate positional embeddings for the text and pose instead of having a shared one has a large effect on the results.

Reference	#steps	Rank 1 ↑	Rank 5 ↑	Rank 10 ↑
Prediction	5	0.09	0.16	0.28
	10	0.08	0.2	0.35
	20	0.09	0.22	0.35
Ground Truth	5	0.25	0.44	0.54
	10	0.21	0.44	0.56
	20	0.27	0.44	0.54

Table 1. Ablation: different step amounts. Top: distance to prediction. Bottom: distance to ground truth pose.

Reference	PE	Rank 1 ↑	Rank 5 ↑	Rank 10 ↑
Prediction	Shared	0.06	0.19	0.28
	Separate	0.08	0.2	0.35
Ground Truth	Shared	0.16	0.37	0.51
	Separate	0.21	0.44	0.56

Table 2. Ablation: shared vs. separate positional embeddings. Top: distance to prediction. Bottom: distance to ground truth pose.

Reference	Hidden dim	Rank 1 ↑	Rank 5 ↑	Rank 10 ↑
Prediction	64	0.06	0.19	0.35
	128	0.08	0.2	0.35
	256	0	0.002	0.007
Ground Truth	64	0.22	0.41	0.53
	128	0.21	0.44	0.56
	256	0.005	0.13	0.34

Table 3. Ablation: different hidden dimensions. Top: distance to prediction. Bottom: distance to ground truth pose.

Reference	FF dim	Rank 1 ↑	Rank 5 ↑	Rank 10 ↑
Prediction	512	0.08	0.23	0.36
	1024	0.06	0.18	0.32
	2048	0.08	0.2	0.35
Ground Truth	512	0.2	0.43	0.53
	1024	0.19	0.42	0.54
	2048	0.21	0.44	0.56

Table 4. Ablation: different feedforward dimensions. Top: distance to prediction. Bottom: distance to ground truth pose.

Varying hidden and feedforward dimensions: although the hidden and feedforward dimensions do not have a large effect, they still yield slightly better results (Tables 3 and 4).

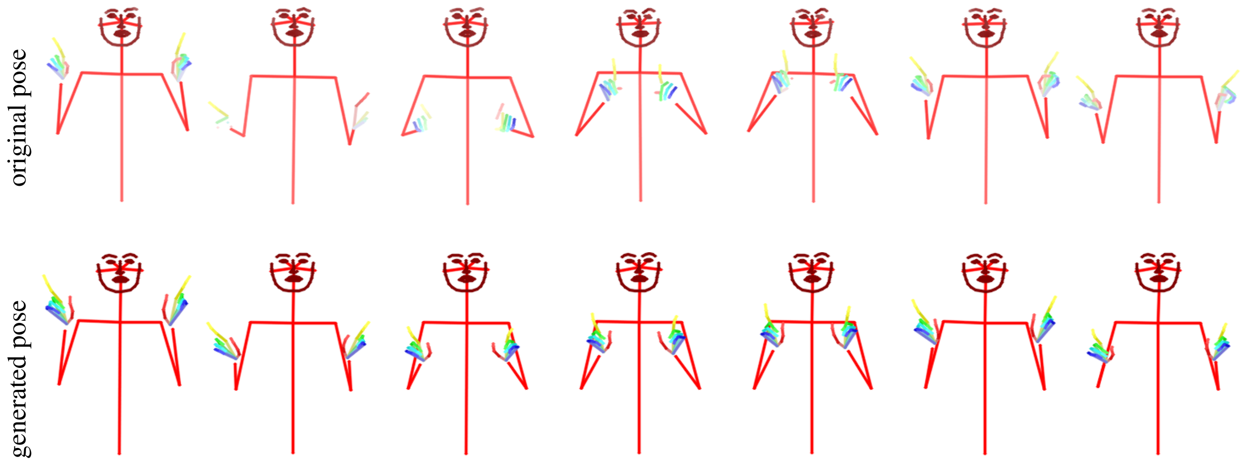

Figure 4. **Result example. Top row:** ground truth pose detected by OpenPose, **bottom row:** generated pose.

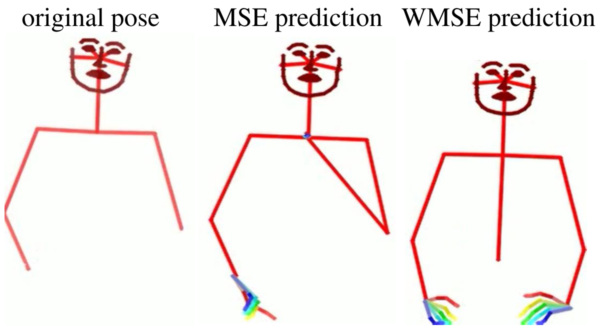

Figure 5. **MSE vs. weighted MSE loss example.** Left to right: original pose, pose generated by a model trained with MSE, pose generated by our model trained with weighted MSE. Regular MSE doesn't take missing keypoints into account, hence the model doesn't learn to generate a full pose.

C. Loss function importance

To show the importance of the weights in our weighted MSE loss function, we experiment with two other loss functions for the training of our model: regular MSE, and half-masked MSE—ignoring low confidence keypoints, but with equal weight for other keypoints. As our data contains many missing keypoints, a loss function that considers them is needed, so the model can learn to fill in the missing keypoints and predict a full pose. Therefore, when using regular MSE, as demonstrated in Figure 5, the model does not predict a full pose, and instead maps a lot of the keypoints to $(0, 0)$. The half-masked MSE loss performed better, but as keypoints with higher confidence are more likely to have correct locations, we wanted them to effect the loss of the model more than keypoints with low confidence, and indeed our masked MSE loss gave the best results.

#steps	train (hrs)	inference (sec)
5	9	0.03
10	20	0.06
20	39	0.12

Table 5. Train and inference duration for different number of steps. Train time is in hours, inference is in seconds.

D. Train and Inference Duration

We train our models on one machine with 4 NVIDIA GeForce GTX TITAN X GPUs. The training and inference time for 2000 epochs over all languages is presented in Table 5 for different step amounts. As demonstrated in the table, doubling the number of steps doubles the train and inference duration as well. Having said that, the inference time for either number of steps is very low.

E. Model Architecture Details

Our model is composed of two parts: the text processor and the pose generator. The text processor consists of:

- HamNoSys tokenizer which converts each glyph into a unique identifier (token).
- Learned embedding layer with dimension 128 to embed the HamNoSys text.
- Learned embedding layer with dimension 128 to embed the text tokens positions (positional embedding).
- Transformer encoder [3] with 2 heads and depths 2, with a feedforward dimension of 2048.

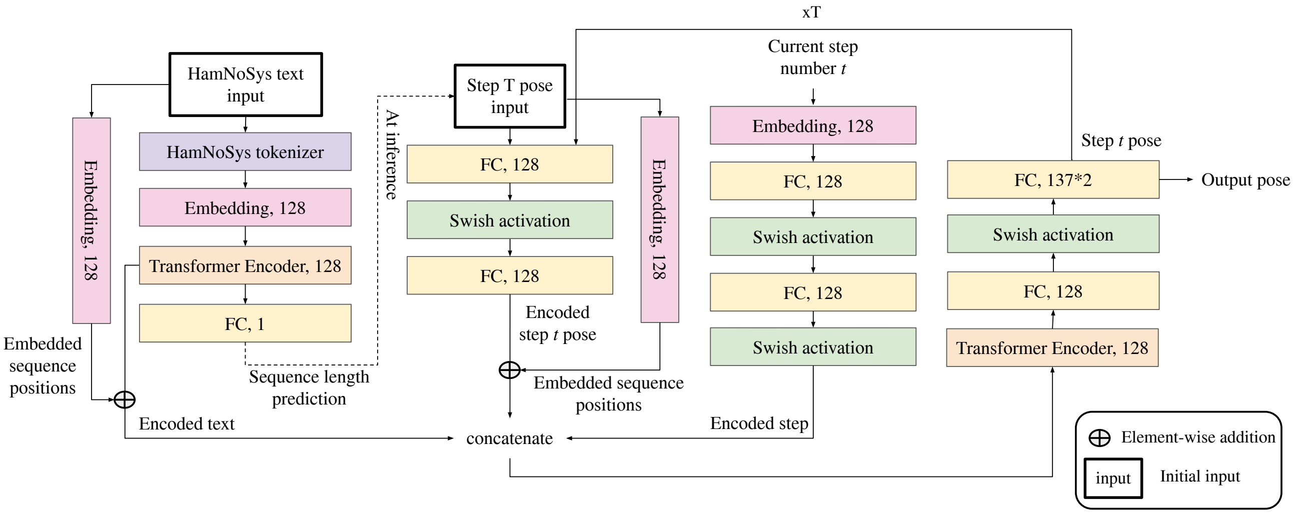

Figure 6. Detailed model architecture.

- Fully connected layer which acts as the sequence length predictor, that gets the encoded text as input and returns a single predicted number.

The pose generator consists of:

- Pose encoding: composed of two fully connected layers with dimension 128 for the hidden and output sizes, with a Swish [2] activation between them; and a positional embedding for the pose sequence locations, composed of a learned embedding layer. They are summed into pose embeddings of dimension 128.

- Step number encoding: composed of a learned embedding layer followed by two fully connected layers with Swish activations between them with hidden and output sizes of 128 as well.

- Encoding of the concatenation of all three encodings of the text, pose, and step. The encoding consists of a transformer encoder with 2 heads and depth 4, with a feedforward dimension of 2048, followed by a pose projection, which is composed of 2 fully connected layers with Swish activation between them, with hidden size of 128 and output size of the pose dimension, which is 137×2 in our case.

The pose is gradually generated over $T = 10$ steps, where each step gets the output of the previous step as input. Finally, after T steps, the output of the model is the output of the pose projection.
A detailed overview of our model architecture is presented in Figure 6.

F. Data

Our data consists of videos of Sign languages signs with their HamNoSys transcriptions. To use these videos as ground truth for pose sequence generation, we extract estimated pose keypoints from them using the OpenPose [1] pose estimation model. Each keypoint $k_i \in K$ consists of a 2D location (x, y) and the confidence of the model in the location of that keypoint, c_i. The number of extracted keypoints is 137 per video frame, spanning the body (K_B, 25 keypoints), face (K_F, 70 keypoints), and hands (K_H, 21 keypoints per hand).

We process the keypoints further to use them, and define $c_{min} = 0.2$ to be the minimal confidence for a keypoint to be considered as identified. As part of the pre-process, we define the following criteria:

$$\begin{gathered}\sum_{i \in K_F} c_i \leq c_{min} \cdot |K_F| \quad \text{or} \\ c_{r\_wrist} + c_{l\_wrist} \leq c_{min}\end{gathered} \tag{1}$$

and remove each leading or trailing frame for which they hold. Meaning, frames in which the face average keypoints confidence is less than c_{min}, or both hands are not identified, are removed.

G. Loss Scaling Factor Derivation

The pose generation process is gradual over T steps, where in each step we use a different step size as defined in the *Method* section in the main paper. The step size at each time step depends on the chosen number of steps. Since we calculate a refinement loss for each step, L_p, to avoid affecting the learning rate when experimenting with different step values, we scale the loss by $ln(T)^2$. This scaling factor emerges from the following derivation:

$$\begin{aligned}
&\sum_{t=T-2}^{0} \alpha_t = \sum_{t=T-2}^{0} \delta_t - \delta_{t+1} = \\
&\sum_{t=T-2}^{0} \log_T (T-t) - \log_T (T-(t+1)) = \\
&\sum_{t=T-2}^{0} \log_T \frac{T-t}{T-t-1} = \\
&\sum_{t=1}^{T-1} \log_T \frac{t+1}{t} = \sum_{t=1}^{T-1} \frac{ln(\frac{t+1}{t})}{ln(T)}
\end{aligned} \tag{2}$$

We do not include $t = T-1$ in the sum since we define α_{T-1} to be the same constant regardless of the step size, to avoid illegal calculations. As we can see from the equation above, in a full cycle of the pose generator, the denominator is the only part that depends on the step number, thus multiplying the loss by the square of the denominator eliminates the step number effect. Therefore, after scaling, the refinement loss term is: $ln(T)^2 L_p$.

References

[1] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7291–7299, 2017. 4

[2] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Swish: a self-gated activation function. *arXiv preprint arXiv:1710.05941*, 7(1):5, 2017. 4

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 3