A. Geodesic distance and Gaussian Distributions

In this section we explain how to interpret Eq. (4) as a product of independent (nearly) Gaussian distributions. Recall that we are attempting to define a distribution $p_\theta(P \mid \mathbf{z})$ over the Lie group $\mathrm{SE}(3)$. The key point is to factorize $P \in \mathrm{SE}(3)$ into the semidirect product $(\mathbf{t}, R) \in \mathbb{R}^3 \rtimes \mathrm{SO}(3)$ of translations and rotations. Assuming a fully factorized conditional model, we can write

$$p_\theta(P \mid \mathbf{z}) = p_\theta(\mathbf{t} \mid \mathbf{z}) p_\theta(R \mid \mathbf{z}).$$

We assume the decoder network $d_\theta(\mathbf{z})$ provides translational parameters $(\boldsymbol{\mu}_\mathbf{t}, \boldsymbol{\sigma}^2_\mathbf{t})$ and rotational parameters $(\boldsymbol{\mu}_R, \boldsymbol{\sigma}^2_R)$[3] from which we define Gaussian distributions on $\mathbb{R}^3$ and $\mathrm{SO}(3)$ as follows.

As $\mathbf{t} \in \mathbb{R}^3$, we can use the usual Gaussian distribution:

$$p_\theta(\mathbf{t}|\mathbf{z}) = \mathcal{N}(\boldsymbol{\mu}_\mathbf{t}, \boldsymbol{\sigma}^2_\mathbf{t}). \tag{13}$$

We put a Gaussian distribution over $R \in \mathrm{SO}(3)$ via the Lie algebra $\mathfrak{so}(3) \simeq R^3$. First, we sample noise from a Gaussian distribution over $\mathfrak{so}(3)$: $\epsilon \sim \mathcal{N}(0, \boldsymbol{\sigma}^2_R)$. We then assume that

$$R = \boldsymbol{\mu}_R \exp(\epsilon),$$

where $\exp\colon \mathfrak{so}(3) \to \mathrm{SO}(3)$ is the exponential map on the Lie algebra. It is tempting to think that, using the logarithm map, $\log\colon \mathrm{SO}(3) \to \mathfrak{so}(3)$,

$$\log(\boldsymbol{\mu}_R^T R) = \epsilon \sim \mathcal{N}(0, \boldsymbol{\sigma}^2_R).$$

Unfortunately, this is not the case. The logarithm map on $\mathrm{SO}(3)$ is not surjective; it takes values in the open ball of radius π contained in $\mathbb{R}^3$. To remedy this, one would technically need to sample ϵ from a *wrapped* Gaussian distribution to ensure that it stays within the image of the logarithm map. In practice, however, a small enough covariance matrix $\boldsymbol{\sigma}^2_R$ means that sampling outside this sphere is extremely unlikely to begin with. As such, we are content to assume that

$$p_\theta(R \mid \mathbf{z}) \propto \exp(-\frac{1}{2} || \log(\boldsymbol{\mu}_R(\mathbf{z})^T R)||^2_{\boldsymbol{\sigma}^2_R}).$$

Note that when the covariance matrix Σ is a scalar multiple of the identity, then the function $d_{\mathrm{SO}(3)}(A, B; \Sigma) = || \log(A^T B)||_\Sigma$ defines a geodesic distance on $\mathrm{SO}(3)$. Therefore, we can write

$$\begin{aligned}
p_\theta(P \mid \mathbf{z}) &= p_\theta(\mathbf{t} \mid \mathbf{z}) p_\theta(R \mid \mathbf{z}) \\
&\propto \exp(-\frac{1}{2}||\boldsymbol{\mu}_\mathbf{t} - \mathbf{t}||^2_{\sigma^2_\mathbf{t}}) \exp(-\frac{1}{2} d_{\mathrm{SO}(3)}(\boldsymbol{\mu}_R, R; \boldsymbol{\sigma}^2_R)^2) \\
&= \exp(-\frac{1}{2}(||\boldsymbol{\mu}_\mathbf{t} - \mathbf{t}||^2_{\sigma^2_\mathbf{t}} + d_{\mathrm{SO}(3)}(\boldsymbol{\mu}_R, R; \boldsymbol{\sigma}^2_R)^2)) \\
&= \exp(-\frac{1}{2} d_{\mathrm{SE}(3)}(\boldsymbol{\mu}_P, P; \boldsymbol{\sigma}^2_P)^2),
\end{aligned}$$

where $\boldsymbol{\mu}_P = (\boldsymbol{\mu}_\mathbf{t}, \boldsymbol{\mu}_R) \in \mathrm{SE}(3)$, $\boldsymbol{\sigma}^2_P = \mathrm{diag}(\boldsymbol{\sigma}^2_{tb}; \boldsymbol{\sigma}^2_R) \in \mathbb{R}^{6\times 6}$, and the last line follows by Theorem 3 in [48].

[3] We have suppressed the dependence of the translational and rotational parameters on both θ and $\mathbf{z}$ to simplify the notation

B. Implementation Details

Here we provide further details on the implementation and training of the models.

Decoder. The decoder consists of a linear layer with d inputs and 256 outputs, followed by a number K of residual blocks and finally a linear layer with 256 inputs and 132 outputs. A residual block consists of two linear layers with 256 inputs and outputs, each followed by a leaky ReLU activation function with negative slope 0.01. Each linear layer in the residual blocks is followed by dropout with probability 0.2. The output of these two layers is multiplied by a learned scalar parameter that is initialized to zero [7] and then added back to the input of the residual block. At initialization, the decoder is therefore equivalent to a linear layer with d inputs and 132 outputs (plus a constant if these layers have bias). Minor detail: we initialize the weights of the last linear layer with a Normal distribution with scale 0.05, and the bias such that the initial generated poses are close to a standard "T-pose".

Encoder for sequences. The first component of the encoder is a linear mapping applied to each frame independently. This is implemented as a 1D convolutional layer with kernel size 1 and output channels 128. This is followed by $K/3$ residual blocks similar to the ones described above, except that they have convolutional layers, and by an average pooling with kernel size 2 and stride 2. After $K/3$ more residual blocks and another downsampling step through average pooling, the matrix is flattened to a vector of size 512. The output is normalized with layer normalization [6], passed through a linear layer with 256 outputs, $K/3$ residual blocks with linear layers, and finally mapped to a vector of size $2d$ that parameterizes the variational distribution.

Optimization. The objective function Eq. (7) is optimized with the Adam optimizer [29] with learning rate 10^{-4} and default parameters, using a batch size of 64 (either 64 static poses or 64 sequences, depending on the scenario).

C. Additional Results

In Fig. 8 we report the same results as in the main text, split according to the value of β. Generally, β does not seem to significantly affect the quantitative metrics considered here, as most results are within error bars. The only exception is $\beta = 5$ which in some cases might lead to lower velocity error and acceleration, but possibly slightly higher position error. Note that, since the lowest value of β roughly approximates a deterministic model, these results

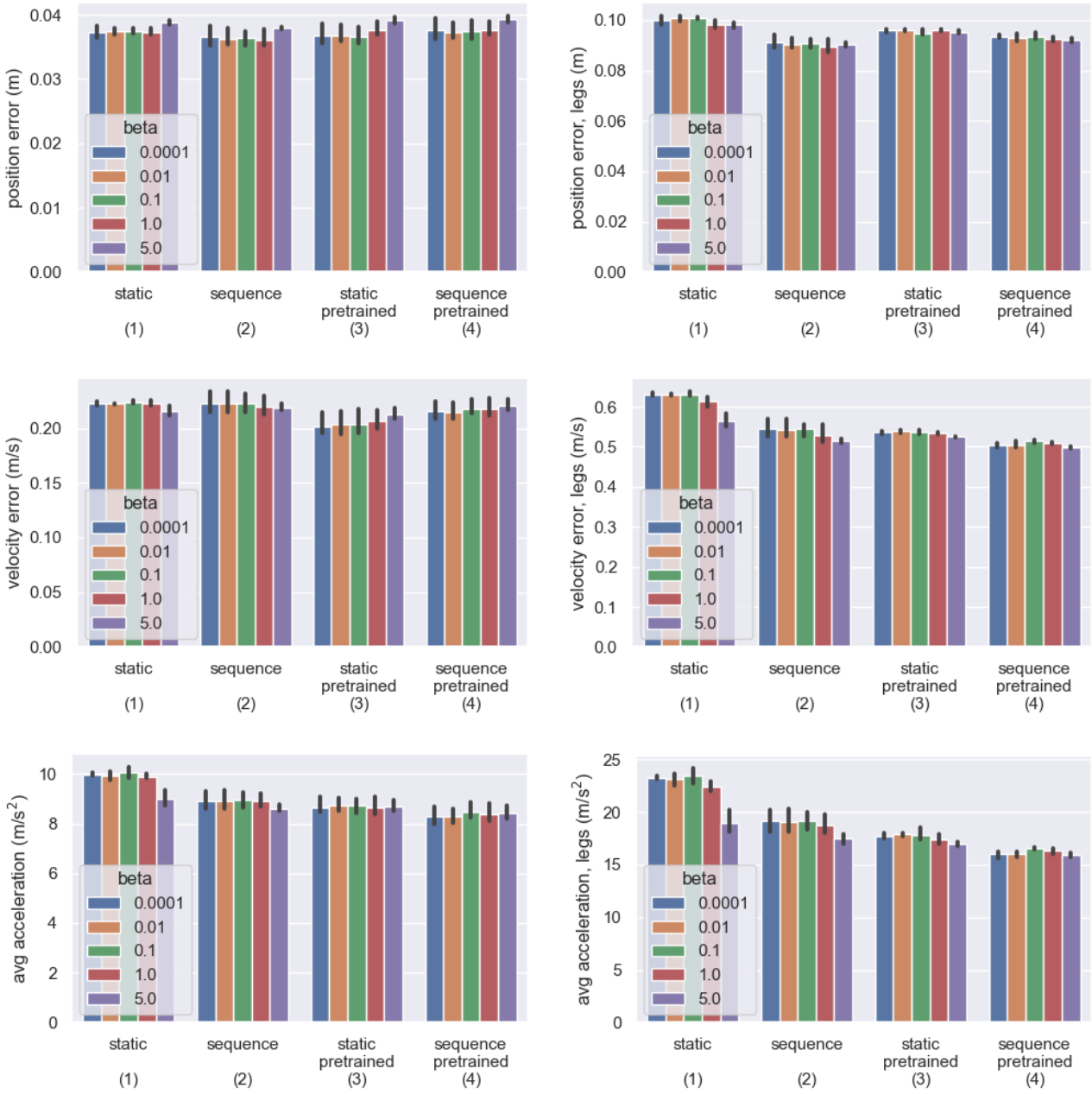

Figure 8. Experimental results broken down by value of β. **Left column**: average results over the entire body. **Right column**: average over leg joints only. **Top to bottom**: average position error, velocity error, and join acceleration. Bars and error bars represent the mean and 95% confidence interval.

suggest that using a vanilla deterministic autoencoder with our setup might work similarly. A possible interpretation of this result is that the task considered here is based on inference and reconstruction, not density estimation or sample generation, and therefore does not necessarily suffer from lowering β—the limit being vanilla deterministic autoencoders. On the other hand, such low values of β will lead to a poor data fit and inadequate generative models. This should be taken into consideration e.g. when sampling from the latent space is of interest.

When looking at the results broken down by latent space dimensionality in Fig. 9, we observe that $d = 15$ sometimes yields worse performance, while the difference between 30 and 60 is generally not significant.

Finally, regarding performance, we remark that the largest models in this paper have less than 2 million parameters

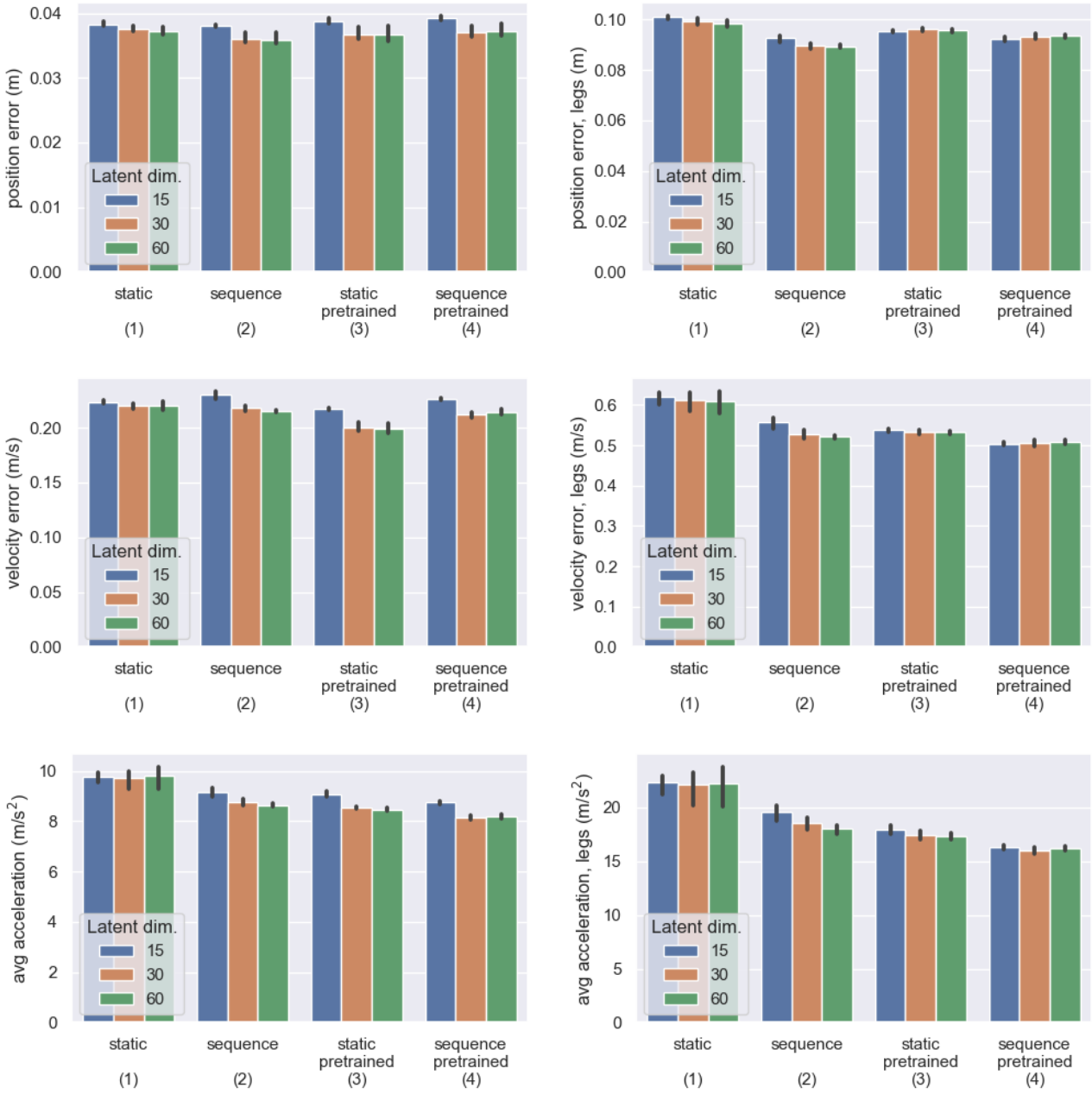

Figure 9. Experimental results broken down by value of the latent space dimensionality, d. **Left column**: average results over the entire body. **Right column**: average over leg joints only. **Top to bottom**: average position error, velocity error, and join acceleration. Bars and error bars represent the mean and 95% confidence interval.

and run at 200Hz on a data center GPU (NVIDIA V100). Note that this performance is measured when encoding and decoding a single image, which reduces latency but is highly suboptimal compared to using larger batches.