Generative Model of Human Body Motion for Walking

Research Project
KIREPRO1PE

Mathias Bastholm
mbas@itu.dk

December 13, 2020

1 Introduction

2 Background

Several tasks exist around the modelling of human motion and the given task usually dictates the shape of the model to a certain extent.

One such example is animating a human 3D mesh to conform with input from a game controller or equivalent keyboard input. This task is of particular interest for the video game industry. One such model is a phase-functioned neural network, wherein multiple networks are samples in a phase, as to mimic the repetitive nature of walking [1]. This approach can be extended to other tasks, such as playing basketball [2]. This approach values being able to run in real-time higher than animation quality. This makes it ideal for video games, but less ideal for other applications where the animations are typically produced in advance and as such there is no requirement for real-time production.

Another example is motion prediction, where the model is given a sequence of human poses and asked to generate future human poses. This is of particular interest for the autonomous car industry, but also has a potential impact for all industries that produce animations, which includes industries working with: movies, video games, entertainment, infotainment, education, etc. As auto-completing an animation could potentially speed up the time it takes to produce a finished animation. A common approach to this task is to use recurrent neural networks [3, 4]. Generative adversarial networks (GAN) are commonly used in image based generative tasks, such as generating human faces [5]. GANs have been shown to also be effective for human motion prediction [6].

In-painting of human motion or interpolation between two different human motions can be considered a special case of motion prediction, where the model gets both the past and future motion, instead of only getting past motion. This could be used to help speed up the animation of humanoids for video games, movies, etc. It seems that data on this particular task is relatively sparse.

GANs have seen great success in image in-painting [7] and image generation [5]. The core idea of GANs being that you train two models in tandem. One model learns to generate content and the other learns to distinguish between generated content and the ground truth. This allows for a sort of ping-pong training that has yielded great results for image models and human motion models.

Recently variational autoencoders [8] (VAE) have been shown to produce results that are as good as the state of the art results produced by GAN models [9]. The VAE is built upon the concept of an autoencoder. An autoencoder is a model that encodes a given input to a latent space and then decodes the latent vector back into something that resembles the input. The latent space of an autoencoder model is analogous to the input given to the generator of a GAN model. Giving random values to the decoder of an autoencoder or as input to the generator of a GAN can be used to generate random realistic outputs. Editing data using GANs and traditional autoencoders is limited. Some models allow for blending between existing data and some support a limited kind of arithmetic [5]. Think for example of adding 0.2 of the image of a face to another while subtracting 0.6 of another face from that image. Traditional autoencoders have no builtin way of ensuring that the latent space is continuous. This means that random sampling or interpolation between two latent values might yield a latent value that would never have been created by the encoder. Such a latent value is unlikely to yield good results. Such a latent space is said to lack regularity. VAEs are autoencoders that are specifically built to add regularity to the latent space. This means that the latent space of a VAE is continuous and can thus be randomly sampled to generate random values, and the interpolation between two latent values is going to always produce valid intermediate latent values. It is plausible that training a VAE for a given domain of human motion, for example walking, could be used for animation inpainting of that given motion. This is accomplished by encoded the human pose at the beginning and end of the gap to be filled. Blending between the two latent values then gives an approximation of what the human motion might look like in the gap. This is restrained by VAE not having any temporal component and by only using a single data point for the beginning and end of the gap. As such it is unlikely to work well for long gaps in motion.

3 Method

Using pytorch [10] several models are constructed, trained and evaluated.

3.1 Data

Archive of Motion Capture as Surface Shapes (AMASS) [11] is a collection of motion capture datasets. A visualization of an example motion can be seen in Figure 1. AMASS collects several datasets into one single place and unifies the data format to a single data format. All models will be trained on data AMASS. Specifically all data from the CMU [12] dataset (part of AMASS) that has been tagged with the `walk` tag. Motion is usually stored as rotations around joints, as human limbs generally speaking only rotate and do not stretch. This means that most motions consist only of rotations and a root position. Only the rotational data is used in order to simplify the different types of data the models are trained on. Some visualizations in this paper include root position, when that is the case the root position used is always the reference root position. All motion data is normalized relative to the training dataset. Since the training dataset is very small this might have a significant impact on the ability to perform on the test dataset, since it is likely that there are poses in the test dataset which are not in the training dataset. See 4.1.1 for more details.

In order to visualize motion a skeleton is needed. A skeleton is mostly just where the joints are positioned relative to each other and a mesh. For this the skeleton from [13] is used. This paper only visualizes the joints, the mesh is never used in visualization.

In order to load in the motion data the library fairmotion [14] is used. It provides an easy way to load and manipulate motion data and is also the tool used for visualization.

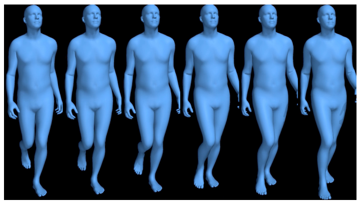


Figure 1: Small snippet of a walking motion from the AMASS dataset.

3.2 Models

An autoencoder consists of an encoder and a decoder. The encoder works by taking an input and then encoding it into a smaller space called the latent space. The encoder then decodes the latent value back into something similar to the original input value. Such an operation by itself does not add any new information, but by doing this we are forcing the model to learn which features of the input are necessary for faithful reconstruction and which features can be discarded. This essentially gives us a compressed version of the input, where each change in the latent space is likely to be clearly visible in the output, similar to principal components analysis [15].

Let x denote the input, e denote the encoder, d denote the decoder, z denote the latent value and $\hat{x}$ the autoencoded value, see Figure 2. We are then looking to minimize

$$\epsilon(x, \hat{x})$$

where ϵ denotes the loss function [16].

Choosing to use the mean squared error (MSE) as the loss function gives

$$\epsilon(x, \hat{x}) = \|x - \hat{x}\|^2 = \|x - \mathrm{d}(\mathrm{e}(x))\|^2$$

Two models are constructed one with a small latent space and one with a large latent space. The models consist of a sequence of dense layers.

The model with a large latent space has a latent space with dimensions 4 and uses the following configuration of layers for its encoder $|x| \to 1024 \to 64$, where each mapping happens in a dense layer and $|x|$ denotes the input size.

The model with a small latent space has a latent space with dimensions 4 and uses the following configuration of layers for its encoder $|x| \to 256 \to 256 \to 256 \to 256 \to 4$, where each mapping happens in a dense layer and $|x|$ denotes the input size.

The decoderes of both models are mirrored versions of the encoders. Between each layer a Gaussian Error Linear Unit (GELU) [17] is used as the activation function, with the exception of the last layer of the decoder. That layer is a linear layer.

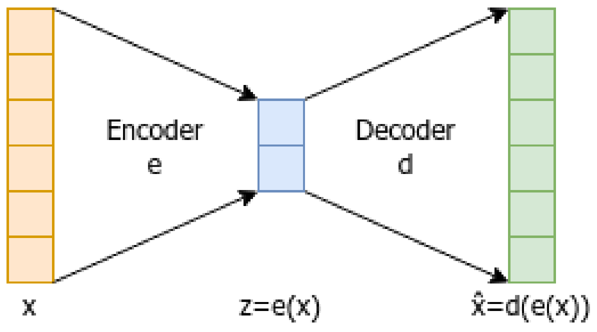


Figure 2: Diagram of the structure behind an autoencoder.

An VAE is an autoencoder whose training is regularized [16]. This is achieved by altering the model architecture and the loss function.

Instead of encoding to a value in the latent space and passing that on to the decoder, the VAE encodes to a distribution in latent space. Decoding then happens by sampling this distribution and using that sample as the basis of decoding. This gives a latent distribution defined by $p(z \mid x)$ and the sample is then $z \sim p(z \mid x)$. Choosing a normal distribution as the latent distribution then yields $z \sim N(\mu, \sigma)$. This means that instead of encoding to a single value the encoder outputs a standard deviation and a mean.

The loss function can then be augmented with the Kullback–Leibler divergence (KLD) [18] which measures relative entropy. The KLD is only dependent on the mean and variance which means that the loss function can be expressed as the MSE loss combined with the KLD loss.

$$\begin{gathered} \text{KLD} = -0.5 \sum 1 + \sigma_x - \mu_x^2 - \sigma_x^2 \\ \text{MSE} = \|x - \text{d}(\text{e}(x))\|^2 \\ \epsilon = \text{KLD} + \text{MSE} \end{gathered}$$

Minimizing on the KLD ensures regularization during the training process.

The VAE model is similar to the autoencoder with a small latent space. It has a latent space with dimensions 8 and uses the following configuration of layers for its encoder $|x| \to 256 \to 256 \to 256 \to 256 \to 8$, where each mapping happens in a dense layer and $|x|$ denotes the input size. In contrast to the autoencoder the VAE outputs two vectors of size 8, that correspond to μ and σ. Like the autoencoders GELU is the activation function for each layer, except the last which is linear.

3.3 Training

notes on how the training was performed

4 Discussion

4.1 Results

The models are evaluated using 3 different methods.

For training purposes the models are autoencoding a pose and the MSE between the original pose and the autoencoded pose gives a loss for training. This is a good baseline for training, but the actual interest lies around interpolation between poses.

Taking a motion and autoencoding each pose gives a good visual indication of how well the model manages coherency between related poses. This is important for animation as large gaps between poses around the same time interval gives a very jittery motion.

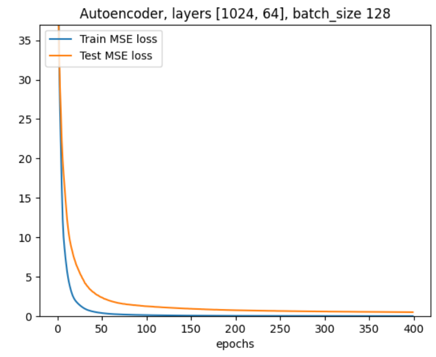
Performing in-painting on a motion, gives a visual indication as well as a numerical indication of how well the model performs. The in-painting is done by iterating through a motion and replacing the data between two poses with a linear interpolation of the two poses or an interpolation in latent space using a model. In-painting is done with increasing intervals of gaps in the motion. For small gaps, such as 1-4 frames, linear interpolation is likely going to perform near perfect, but for larger gaps, such as 32 frames, knowledge of human walking is required to fill the gap correctly.

4.1.1 Autoencoder

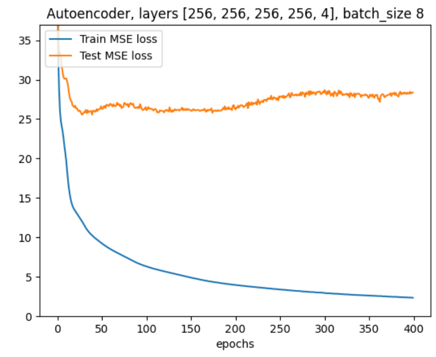
Two models are trained. One model has a small latent space (8 dimensions), while the other has a large latent space (128 dimensions).

The model with a large latent space becomes very good at autoencoding poses, as can be seen by the loss in Figure 3a. This is also evident in the very smooth autoencoded motion seen in Figure 5. Figure 7 shows a visualization of the motion inpainting, which reveals a problem with the large latent space. The model seems to encode rotations in a way that is so similar to the original rotations that the interpolation in latent space is almost identical to the interpolation in the motion space. This can also be seen in the comparison in Table 1a.

In an effort to combat this issue a model was trained with a smaller latent space. This model did not achieve as high of an accuracy as the model with the large latent space, but it is sufficient for comparison. Furthermore there is a large gap in accuracy on training vs test data. This can be explained by poses in the test dataset being very far away from the train dataset, Figure 4 shows an example of such a motion in the test dataset. Having a larger dataset would help alleviate this. Table 1b shows a worse performance on in-painting, this likely a result of the model not being as good at autoencoding poses. Figure 8 and Figure 6 shows that while the model autoencodes the poses pretty well it lacks the ability to blend smoothly between similar poses, resulting in very jittery motion. This motivates the training of a VAE.



(a) Loss over epochs for the model with a large latent space.



(b) Loss over epochs for the model with a small latent space.

Figure 3: MSE loss of autoencoding a pose for training and test datasets for the two autoencoder models.

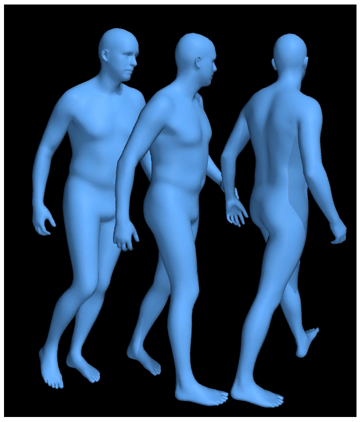


Figure 4: Subject 46, trial 1 from the CMU dataset. The motion depicts turning around, as opposed to simply walking straight ahead.

Step	Latent loss	Linear loss	Difference
1	0.000000	0.000000	0.000000
2	0.000004	0.000002	-0.000002
4	0.000008	0.000006	-0.000002
8	0.000029	0.000027	-0.000002
16	0.000142	0.000136	-0.000006
32	0.001255	0.001255	0.000000
64	0.003744	0.003736	-0.000008
128	0.011174	0.011202	0.000028

(a) Comparison for the model with a large latent space.

Step	Latent loss	Linear loss	Difference
1	0.000000	0.000000	0.000000
2	0.000169	0.000002	-0.000168
4	0.000307	0.000006	-0.000301
8	0.000497	0.000027	-0.000470
16	0.000825	0.000136	-0.000689
32	0.002039	0.001255	-0.000784
64	0.005852	0.003736	-0.002117
128	0.011285	0.011202	-0.000083

(b) Comparison for the model with a small latent space.

Table 1: Comparison of loss using an interpolation in latent space and an interpolation in the motion space on a single motion from the test dataset. Step denotes the distance between the start pose and end pose in frames.

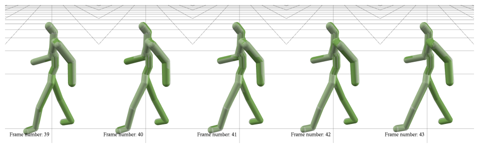


Figure 5: Frames from autoencoding a motion using the autoencoder with a large latent space. The green skeleton is the autoencoded poses and the white skeleton is the reference pose. The overlap makes it hard to distinguish the two skeletons from each other.

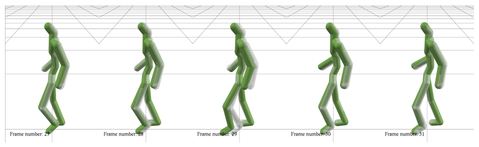


Figure 6: Frames from autoencoding a motion using the autoencoder with a small latent space. Frames from autoencoding every pose in a motion. The green skeleton is the autoencoded poses and the white skeleton is the reference pose. The third frame suddenly jumps compared with the other frames.

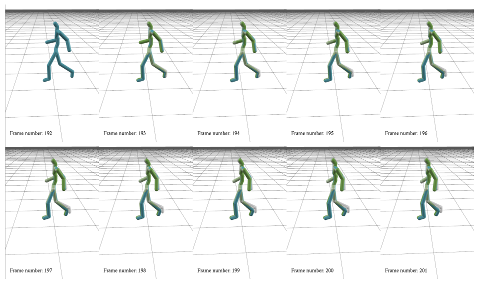


Figure 7: Frames from inpainting a motion using the autoencoder with a large latent space. The gaps are of length 32. The first frame shows the start pose. The green skeleton is the autoencoded poses, the white skeleton is the reference pose and the blue skeleton is a linear interpolation. Observe how closely the autoencoded poses match the linearly interpolated poses.

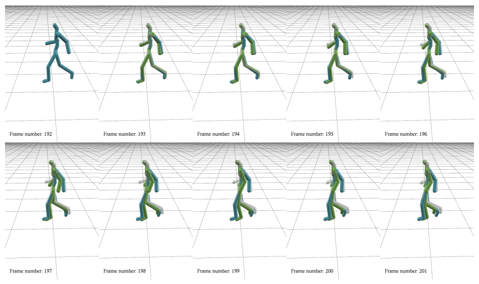


Figure 8: Frames from inpainting a motion using the autoencoder with a small latent space. The gaps are of length 32. The first frame shows the start pose. The green skeleton is the autoencoded poses, the white skeleton is the reference pose and the blue skeleton is a linear interpolation. Observe how the autoencoded poses diverge from both the reference poses and the linearly interpolated poses.

4.1.2 Variational autoencoder

A VAE is trained using a small latent space in order to avoid the issues around large latent spaces described in 4.1.1. It achieves a similar accuracy and has the same gap between test and train loss as the autoencoder with a small latent space, as can be seen in Figure 9.

Comparing Figure 10 and Figure 6 it becomes clear that the variational encoder is capable of blending between latents space without jitter, showing the difference between a regular autoencoder and a VAE.

Figure 11 shows the result of in-painting a motion using this model. As can be seen, motion of the legs that have been interpolated using this model is actually closer to the real motion than to that of the linear interpolation. This suggests that the model is not just blindly trying to compress rotations, but that it has actually learned something about what a human walking motion looks like. What is also evident is the fact that the motion has various artifacts, such as spurious arm movement. This is probably because the model struggles to autoencode poses accurately and also the likely reason why the model does not score higher than the linear interpolation in Table 2. This suggests that a higher accuracy variation autoencoder would likely be able to perform motion in-painting/interpolation for human walking motion.

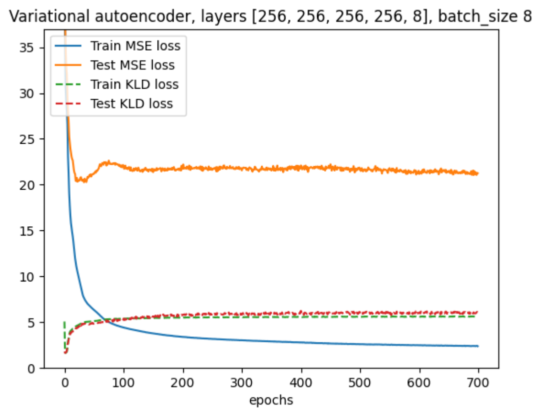


Figure 9: MSE loss of autoencoding a pose and KLD loss for both the training and test datasets.

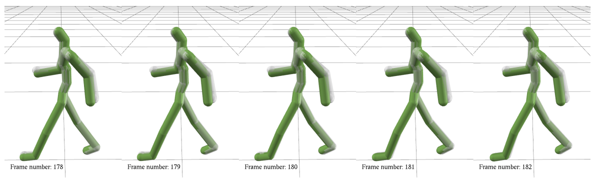


Figure 10: Frames from autoencoding a motion using the VAE. The green skeleton is the autoencoded poses and the white skeleton is the reference pose. Observe how the errors are mostly offsets and not sudden jumps.

4.2 Further research

clustered-specialists-paper [19]

something about motion prediction maybe (a way to utilize forwards/backwards info)

closer comparison with existing motion prediction models

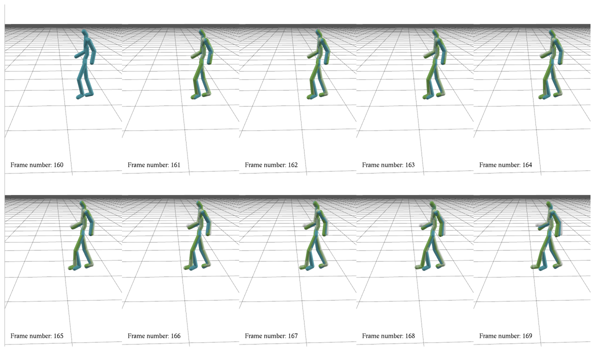


Figure 11: Frames from inpainting a motion using the autoencoder with a small latent space. The gaps are of length 32. The first frame shows the start pose. The green skeleton is the autoencoded poses, the white skeleton is the reference pose and the blue skeleton is a linear interpolation. Observe how the autoencoded poses overlap with the reference poses and with the linear interpolated poses.

Step	Latent loss	Linear loss	Difference
1	0.000000	0.000000	0.000000
2	0.021803	0.000002	-0.021801
4	0.032558	0.000006	-0.032552
8	0.037987	0.000027	-0.037960
16	0.039968	0.000136	-0.039831
32	0.041304	0.001255	-0.040048
64	0.038672	0.003736	-0.034937
128	0.038981	0.011202	-0.027780

Table 2: Comparison of loss using an interpolation in latent space and an interpolation in the motion space on a single motion from the test dataset. Step denotes the distance between the start pose and end pose in frames.

using vq-vae

using the full amass data training

5 Conclusion

References

1. Holden, D., Komura, T. & Saito, J. Phase-Functioned Neural Networks for Character Control. *ACM Trans. Graph.* **36.** ISSN: 0730-0301. `https://doi.org/10.1145/3072959.3073663` (July 2017).
2. Starke, S., Zhao, Y., Komura, T. & Zaman, K. Local Motion Phases for Learning Multi-Contact Character Movements. *ACM Trans. Graph.* **39.** ISSN: 0730-0301. `https://doi.org/10.1145/3386569.3392450` (July 2020).
3. Hu, J., Fan, Z., Liao, J. & Liu, L. *Predicting Long-Term Skeletal Motions by a Spatio-Temporal Hierarchical Recurrent Network* 2020. arXiv: `1911.02404 [cs.CV]`.
4. Jain, A., Zamir, A. R., Savarese, S. & Saxena, A. *Structural-RNN: Deep Learning on Spatio-Temporal Graphs* 2016. arXiv: `1511.05298 [cs.CV]`.
5. Karras, T., Laine, S. & Aila, T. *A Style-Based Generator Architecture for Generative Adversarial Networks* 2019. arXiv: `1812.04948 [cs.NE]`.
6. Ruiz, A. H., Gall, J. & Moreno-Noguer, F. *Human Motion Prediction via Spatio-Temporal Inpainting* 2019. arXiv: `1812.05478 [cs.CV]`.
7. Bau, D. *et al.* Semantic Photo Manipulation with a Generative Image Prior. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)* **38** (2019).
8. Kingma, D. P. & Welling, M. *Auto-Encoding Variational Bayes* 2014. arXiv: `1312.6114 [stat.ML]`.
9. Razavi, A., van den Oord, A. & Vinyals, O. *Generating Diverse High-Fidelity Images with VQ-VAE-2* 2019. arXiv: `1906.00446 [cs.LG]`.
10. Paszke, A. *et al.* in *Advances in Neural Information Processing Systems 32* (eds Wallach, H. *et al.*) 8024–8035 (Curran Associates, Inc., 2019). `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.
11. Mahmood, N., Ghorbani, N., F. Troje, N., Pons-Moll, G. & Black, M. J. *AMASS: Archive of Motion Capture as Surface Shapes* in *The IEEE International Conference on Computer Vision (ICCV)* (Oct. 2019). `https://amass.is.tue.mpg.de`.
12. Carnegie Mellon University. *CMU MoCap Dataset* `http://mocap.cs.cmu.edu`.
13. Romero, J., Tzionas, D. & Black, M. J. Embodied Hands: Modeling and Capturing Hands and Bodies Together. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia). 245:1–245:17* **36** (Nov. 2017).
14. Gopinath, D. & Won, J. *fairmotion - Tools to load, process and visualize motion capture data* Github. 2020. `https://github.com/facebookresearch/fairmotion`.
15. Shlens, J. *A Tutorial on Principal Component Analysis* 2014. arXiv: `1404.1100 [cs.LG]`.
16. Rocca, J. & Rocca, B. *Understanding Variational Autoencoders (VAEs)* Sept. 2019. `https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73`.
17. Hendrycks, D. & Gimpel, K. *Gaussian Error Linear Units (GELUs)* 2020. arXiv: `1606.08415 [cs.LG]`.
18. Kullback, S. & Leibler, R. A. On Information and Sufficiency. *Ann. Math. Statist.* **22,** 79–86. `https://doi.org/10.1214/aoms/1177729694` (Mar. 1951).
19. Won, J., Gopinath, D. & Hodgins, J. A Scalable Approach to Control Diverse Behaviors for Physically Simulated Characters. *ACM Trans. Graph.* **39.** `https://doi.org/10.1145/3386569.3392381` (2020).