Generative Model of Human Body Motion for Walking

Research Project
KIREPRO1PE

Mathias Bastholm
mbas@itu.dk

December 11, 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 [**TODO**]. 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 [**TODO**]. 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

3.1 Data

Training a model for human motion analysis requires a great deal of human motion data. There exists two efforts to help in this regard. First of the AMASS data set is an attempt to amass a large quantity of human motion data from motion capture [10]. Secondly the fairmotion library is a library that allows you to take motion from all of the heterogeneous formats currently available, including the AMASS data set, and access all of it through one homogenous interface [11]. The fairmotion library requires both motion data and skeleton/model data of a human. As such, AMASS was used for the motion data and [12] for the skeleton/model data.

Only a subset of the AMASS data set was used. This sped up training by training on less data and it meant not having to stream in training data, as the subset used was small enough to fit into memory.

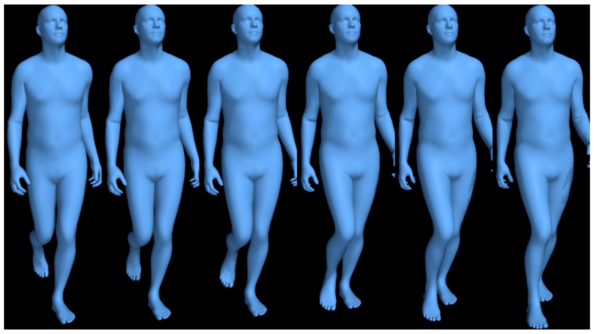

Figure 1: Small snippet of a walking motion from the AMASS dataset.

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 is pointless, 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 [**TODO**].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 is pointless, 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 [**TODO**].

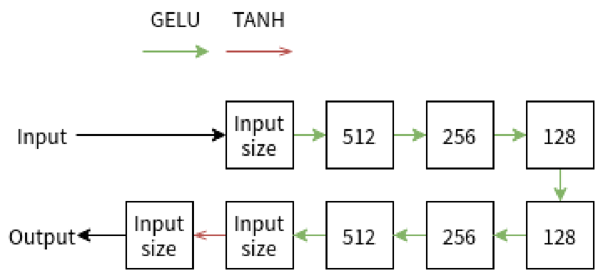

Figure 2: Diagram of the model used to implement a simple autoencoder. Each arrow represents a linear layer with either the GELU or tanh activation function applied to its output.

A simple autoencoder using decreasingly smaller/increasingly larger hidden layers for encoding/decoding can be seen in Figure 2. This will be used as a baseline of comparison against the VAE.

Since the above model is bad (non-continuous) we actually want to use a VAE which, TODO: describe VAE and my implementation.

3.3 Training

notes on how the training was performed

4 Discussion

4.1 Results

AE latent blending vs. VAE latent blending vs. linear interpolation (take a motion add gaps, fill gaps, record MSE)

insert some examples here (image sequences) of good results and bad results

4.1.1 Autoencoder

The autoencoder has been trained to encode a pose. The training then optimizes the MSE loss between the original pose and the autoencoded pose. While training we record the loss for the training dataset and the test dataset at the end of each epoch. A graph of the test and training loss over the epochs can be seen in Figure 3. There is a large gap between the training and test loss. This is because there are poses in the test dataset that are 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. Since both the training and test loss decrease monotonically over time there is no indication of overfitting.

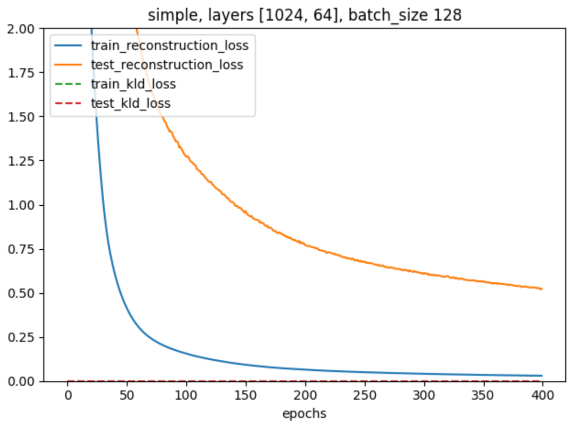

Figure 3: Loss over epochs for the autoencoder. The reconstruction_loss is the MSE loss, the kld_loss is not applicable for this model.

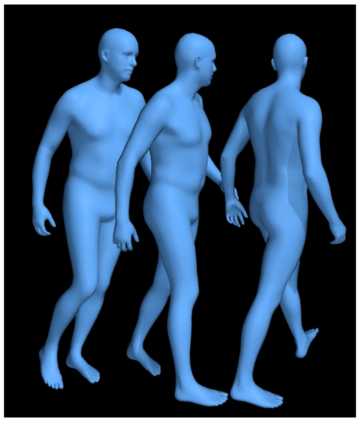

Figure 4: Subject 46, trial 1 from the CMU dataset. The motion depicts turning around, as opposed to simply walking straight ahead.

4.1.2 Variational autoencoder

4.2 Further research

clustered-specialists-paper [13]

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

closer comparison with existing motion prediction models

using vq-vae

using the full amass data training

5 Conclusion

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

Table 1: Caption here

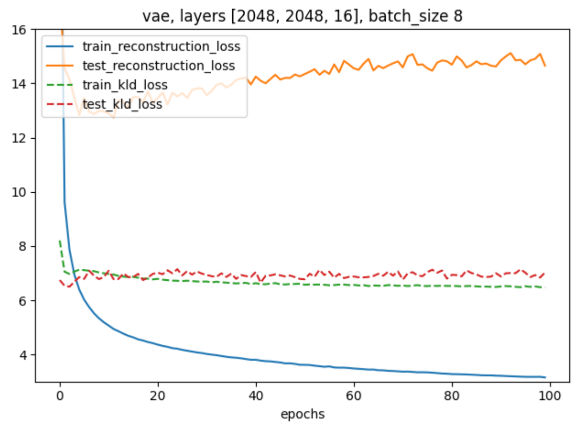

Figure 5: TODO

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. 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`.
11. Gopinath, D. & Won, J. *fairmotion - Tools to load, process and visualize motion capture data* Github. 2020. `https://github.com/facebookresearch/fairmotion`.
12. 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).
13. 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).