# Links

[Link to Pose Priors Drive folder](https://drive.google.com/drive/folders/1LQtQDqbmo3WiTiIp83QarD2PZFkXKwYg?usp=sharing)

# Plan

* Proposed goal: To create an open source VAE pose prior.
* Goal: 7/7/20 - **Step 0**: **Getting used to the CMU Mocap data set and visualizing the data.**
  + The data set is here: <http://mocap.cs.cmu.edu/>
  + Try downloading a little bit of it and plotting it in a [Colab](https://colab.sandbox.google.com/notebooks/intro.ipynb#recent=true) notebook (or Jupyter notebook if you don’t have access).
  + See if you can automate the download of a single file and visualize it as a sequence of frames.
  + (FYI, If you want to make an animated GIF, I’ve seen [pillow](https://note.nkmk.me/en/python-pillow-gif/) recommended.)
* Goal: 7/21/20 - **Step 1**: **Convert the CMU Mocap dataset into a format that TensorFlow can read.**
  + For each sequence in the data set, create a SequenceExample that stores a unique id for that sequence, timestamps, and the 3D pose keypoints.
    - We will discuss the exact format later.
    - I recommend the [MediaSequence](https://github.com/google/mediapipe/tree/master/mediapipe/util/sequence) format (which I wrote, so I’m biased). We can discuss using the actual library or a simpler way to get the same results.
  + Here’s an [example of converting the Charades dataset](https://github.com/google/mediapipe/blob/master/mediapipe/examples/desktop/media_sequence/charades_dataset.py), and something similar would be ideal. Here’s an even [shorter demo example](https://github.com/google/mediapipe/blob/master/mediapipe/examples/desktop/media_sequence/demo_dataset.py).
  + The goal of this step is to have a concrete output from your work regardless of how the research turns out.
  + Getting data out of sequence examples with [parse\_single\_sequence\_example](https://www.tensorflow.org/api_docs/python/tf/io/parse_single_sequence_example)
* (see subgoals below) **Step 2**: **Create a variational auto-encoder (VAE) pose prior based on CMU mocap data.**
  + There is one example of a similar attempt in the literature, but they used a different set of input data. [[VPoser paper (Section 3.3)](https://arxiv.org/pdf/1904.05866.pdf), [website](https://github.com/nghorbani/human_body_prior)]
  + [The original paper about VAEs is here.](https://arxiv.org/abs/1312.6114)
  + [This tutorial about VAEs is reasonable.](https://jaan.io/what-is-variational-autoencoder-vae-tutorial/)
  + [Another reasonable VAE tutorial.](https://www.jeremyjordan.me/variational-autoencoders/)
  + [Another VAE tutorial.](https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73)
  + [An example of creating a VAE in tensorflow.](https://medium.com/tensorflow/variational-autoencoders-with-tensorflow-probability-layers-d06c658931b7) [I can also make more examples if needed.]
  + Subgoals:
    - **MNIST example - 1 week after step 1**
    - **Simple case of CMU Mocap input and output - 2 weeks after previous**
    - Good working example - 2 or 4 weeks after previous?
  + Lines of work:
    - density estimation (here’s a recent VAE paper that probably lists a lot of these in the related works section): <https://arxiv.org/abs/2007.03898>
      * VAE - see links above
      * Autoregressive models - e.g. PixelCNN, PixelSNAIL
      * Flow based methods - GLOW
      * Deep energy-based models - <https://en.wikipedia.org/wiki/Energy_based_model>
      * (may be possible to mix a graph-neural network into most of the above. If with attention, this is Tranformer models.)
    - sequence generation tasks
      * <https://github.com/sebastianstarke/AI4Animation>
      * GAN
      * <https://arxiv.org/abs/2004.08692>
    - Predict words in the description from the motion
      * captioning / classification (can also use transformers)
  + Some highlights:
    - VAEs seem a good first step
    - Captioning
    - Flow based models are also good.
  + VAE planning
    - Try to implement on MNIST:
      * TFP walk through:
        + <https://medium.com/tensorflow/variational-autoencoders-with-tensorflow-probability-layers-d06c658931b7>
        + <https://www.tensorflow.org/probability/examples/Probabilistic_Layers_VAE>
    - Let’s talk through the math, for example:
      * General tutorial on VAEs: <https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73>
      * Original paper: [The original paper about VAEs is here.](https://arxiv.org/abs/1312.6114)
      * Understanding the ELBO terms: <https://www.zinkov.com/posts/2018-11-02-decomposing-the-elbo/>
      * Key terms:
        + Reparameterization trick
        + Evidence lower bound (ELBO)
        + Posterior / Surrogate Posterior / Approximate Posterior
        + Prior / Marginal = p(z)
        + Encoder / Decoder

posterior = q( z | x ) = encoder

reconstruction = decoder = p(x | z)

prior = marginal = p(z)

* + - Relevant papers for later:
      * Pose VAE [VPoser paper (Section 3.3)](https://arxiv.org/pdf/1904.05866.pdf), [website](https://github.com/nghorbani/human_body_prior)
      * Fixing a Broken ELBO: <https://arxiv.org/abs/1711.00464>

## Data format

Tensorflow works best with Tensorflow data formats. The standard ones are tf.Example and tf.SequenceExample. Personally, I like to use tf.SequenceExample for sequential data because I find it to be a better match. tf.SequenceExamples are basically two key value stores. The context portion stores things that apply to the entire example. The feature\_lists stores sequence data. I wrote a big project for using these SequenceExamples [here](http://go/mediasequence-ref) ([relevant code](https://github.com/google/mediapipe/blob/master/mediapipe/util/sequence/media_sequence_util.py)), but I’ll suggest writing your own lighter-weight version by hand. I will suggest using the same keys as the other project, so they may seem funny, but we can choose to use something else if you prefer.

|  |
| --- |
| context = {  “example/id”: “r4nd0m/01/01” # “a unique id for this sequence”  “example/asf”: “the entire asf file as a string. We won’t use it, but it’s good context”  “example/angle\_header”: [“head-x”, “head-y”, “neck-x”,...] # headers for ALL\_JOINT\_ANGLES  “example/xyz\_header”: [“head-x”, “head-y”, “head-z”] # headers for ALL\_JOINT\_XYZ  “example/motion\_description”: [“text copied from the page.”]  #### IF WE KNOW ANYTHING ABOUT THE ACTION OR SUBJECT WE SHOULD STORE THAT INFORMATION IN THE CONTEXT TOO!  }  feature\_lists {  “ALL\_JOINT\_ANGLES/feature/floats”: [0.1, 0.2, …] # the angles from the amc files concatenated into a long vector per timestamp. The order needs to be documented.  “ALL\_JOINT\_ANGLES/feature/timestamp”: [0, 1/120\*1000000, 2/120\*1000000] # the time, in microseconds of each frame. We use microseconds because several of our other systems do.  “ALL\_JOINT\_XYZ/feature/floats”: [0.1, 0.2, …] # the x,y,z locations of each joint after updating the skeleton with the amc angles.  “ALL\_JOINT\_XYZ/feature/timestamp”: [0, 1/120\*1000000, 2/120\*1000000] # the time, in microseconds of each frame. It’s redundant, but low cost to store it twice.  } |

References for [Protos](https://developers.google.com/protocol-buffers/docs/reference/python-generated) and [TF.SequenceExamples](https://github.com/tensorflow/tensorflow/blob/2b96f3662bd776e277f86997659e61046b56c315/tensorflow/core/example/example.proto#L300) (available as tf.train.SequenceExample)

s = tf.train.SequenceExample()

s.context.feature["example/id"].bytes\_list.value.append(b"an\_example")

s.feature\_lists.feature\_list["ALL\_JOINT\_ANGLES/feature/floats"].feature.add().float\_list.value[:] = [0.1, 0.2, 0.3]

s.feature\_lists.feature\_list["ALL\_JOINT\_ANGLES/feature/timestamp"].feature.add().int64\_list.value.append(1)

I confirmed that the above code all runs. One thing to note is that the .feature.add() call should happen once per timestep. There should be only one value in the “.../timestamp” .value per timestep and one row of floats in the corresponding “.../floats” .value. You’ll repeat these feature.add() calls like you’re creating new rows in a CSV.

“a string”.encode(“utf8”)

b”some bytes”.decode(“utf8”)

str(sequence) # write this to a file somewhere for inspection and debugging.

# Other projects

Most of the other projects that are suitable for doing in an open source world are similar to step 1. If you’re interested in adding multiple data sets, it would definitely be appreciated. (Video models are typically require a lot of compute to train, and I don’t know how much you have access to given the COVID19 internship changes.)

The exception is that we’re trying to open source a very cool bit of research we’ve done on [how to understand time](https://arxiv.org/abs/1910.09588). Unfortunately, it’s not quite ready to be used yet, but it may in a few weeks. The model is a fancy VAE, so working with VAEs would still be helpful. A piece of the CMU Mocap data set is used in our paper, but it would be good to expand it.

# Weekly progress

2020-06-30

* Status:
  + Some visualizations complete! (see this [Colab](https://colab.research.google.com/drive/1zuzOMOwJAB9n7VNQQ0RcbmSbbqWDDryG?usp=sharing))
* Discussion:
  + Nina has access to GCP and has started some downloading and visualization.
* Questions:
  + What’s the right output format to save on disk?
    - **Big question**
    - Initial impression:
      * the amc files has some degrees, but variable per joint.
      * the asf file is the skeleton, but very complex
    - What do we want:
      * All of the poses/frame for one sequence are in one SequenceExample.
      * For each sequence, want any labels that exist (e.g. “walking”, “running”, “dancing”)
      * For each sequence, we want the 3D XYZ location of each joint (possibly from github library)
      * For each sequence, we want the angles between joints. -- Start with the angles from the ACM file?
        + Future direction will be to think about angles further. [Check out the 6D paper](https://openaccess.thecvf.com/content_CVPR_2019/html/Zhou_On_the_Continuity_of_Rotation_Representations_in_Neural_Networks_CVPR_2019_paper.html).
  + What are our goals and timeline. -- Added to the doc above.