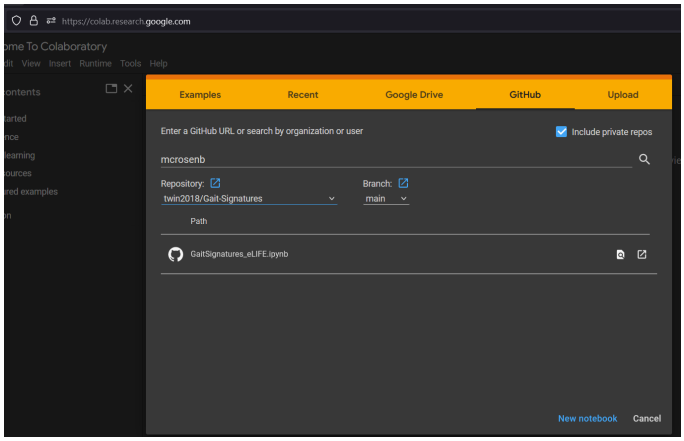
Gait_ signatures Code Execution Instructions

1. **Review manuscript:** Before running the code, please read our manuscript, particularly the methods section that outlines important definitions. [link here]
2. **Setting up Google Colab:**

   General notes

   I. **Upload** our entire Gait Signature repository folder: https://github.com/bermanlabemory/gait_signatures with the Colab scripts, data and functions onto your Google Drive (‘My Drive’) and **rename** the folder as/if desired.

   II. To open IPYNB files from GitHub in Colab, go to *colab.research.google.com* and click on the 'GitHub tab. You can then navigate to your desired repository and select the file you want to open.



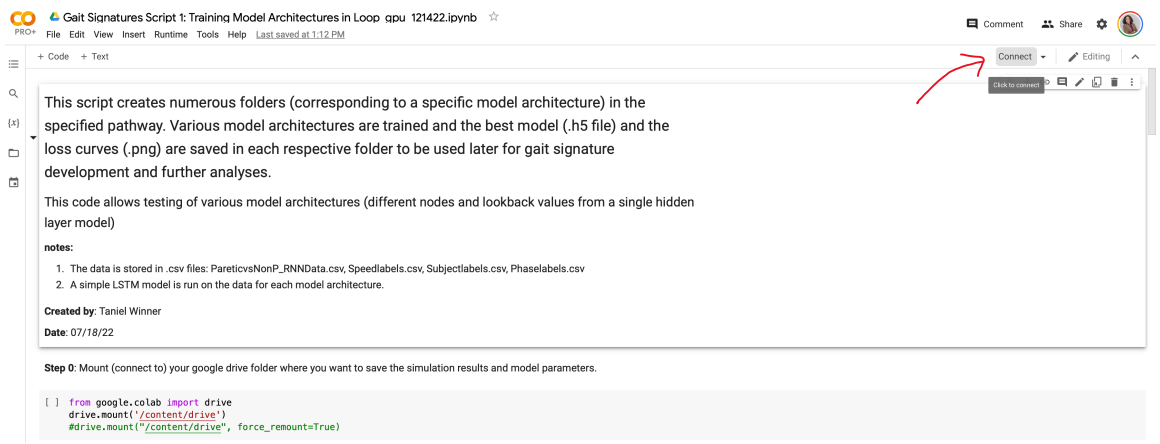
3. **Locate .ipynb files:** There are 3 main scripts named:

i) Gait Signatures Script 1: Training Model Architectures in Loop_gpu_121422.ipynb
ii) Gait Signatures Script 2: Extract and Save Externally and Self-Driven HCs_gpu_121422.ipynb
iii) Gait Signatures Script 3: Load Externally and Self-Driven HCs to Generate Signatures_121422.ipynb

These files should be **run in chronological order**: Script 1, Script 2 then Script 3

The notebooks are set up in such a way that one can specify multiple models to train in a single loop. The model, results and figures would be saved in respective generated folders for each model (denoted by the specific model architecture LSTM units and lookback parameter).

4. **Open** ‘Gait Signatures Script 1: Train Model Architectures.ipynb’
5. On the top right of the notebook click ‘connect’



6. Scroll down '**Step 2**: Load module in Google Drive' and **update the folder name** to the folder that you uploaded from GitHub (if you renamed this folder then this is the name that you would specify). If folder was uploaded to 'My Drive' then there is no need to change the path variable below, otherwise change the path variable to the path that matches where the folder was uploaded.

Step 2: Create folder in your drive and load module in Google Drive

```
# The path to save the models and read data from

### !!! please create and name the main folder that you would like to save all results -- ensure that the functions and data files are in this folder
folder = 'TestScripts_121422/'


path = '/content/drive/My Drive/'+ folder

# Insert the directory
import sys
sys.path.insert(0,path)
```

7. Scroll down to '**Step 3:** Load in data and specify variables/parameters'. If you are using the provided gait signature data file 'PareticvsNonP_RNNData.csv', then the contents of this cell do not need to be altered. Our dataset is a 2D matrix of 6 rows and 108000 columns. Each of the 72 trials in our dataset is 1500 samples long with 6-time varying features (paretic hip flexion, paretic knee flexion, paretic ankle flexion, non-paretic hip flexion, non-paretic knee flexion and non-paretic ankle flexion). The data was concatenated or strung end to end e.g. [trial 1 trial 2 trial 3...trial n] where each trial is a 6x1500 matrix.

8. Scroll down to '**Step 4**: Develop list of model architectures...' Variables in this cell can be changed.

Step 4: Develop list of model architectures and corresponding variables to train. This step also generates a list of folder names and pathways where the models will be saved and accessed later.

```
# generate a list of models and corresponding parameters to test
test_model_nodes = [512]
seqs = [249,499,749] #lookback parameter

# run multiple model architechtures many times to test stability of cost function outputs
runs = 1 # stability analysis - repeat each model architecture this many times
test_model_seq = np.repeat(seqs, runs)

All_nodes = np.empty([0,1], dtype='int')
All_seq = np.empty([0,1],dtype='int')
All_valseg = np.empty([0,1],dtype='int')
All_trainseg = np.empty([0,1],dtype='int')
All_modelname = []
All_mod_name = []

count = 0; #initialize model run -- this serves as the model run ID number
for a in test_model_nodes:
  for b in test_model_seq:
    if count < runs:
      count = count + 1
    else:
      count = 1 # reset counter when all runs of certain model attained
    #if statement for valseg, trainseg based on sequence length
    if int(b) == 249:
      trainseg = 4
      valseg = 2
    elif int(b) == 499:
      trainseg = 2
      valseg = 1
    elif int(b) == 749:
      trainseg = 1
      valseg = 1

    All_nodes = np.append(All_nodes, a)
    All_seq = np.append(All_seq, int(b))
    All_valseg = np.append(All_valseg, valseg)
    All_trainseg = np.append(All_trainseg, trainseg)
    All_modelname = np.append(All_modelname, 'run_' + str(count) + '_UNIT_' + str(a) + '_LB_' + str(b) + '/' )
    All_mod_name = np.append(All_mod_name, 'run_' + str(count) + '_UNIT_' + str(a) + '_LB_' + str(b) )
```

i. **Specify the number of nodes** you would like to run for your model(s). If you wish to run one model for e.g., set *test_model_nodes* = [125] for 125 nodes, otherwise make a vector of nodes to test e.g., testing 3 types of nodes; *test_model_nodes* = [1, 200, 500].

ii. **Specify the lookback parameters** for each of the model nodes specified above. E.g., *seqs* = [249,499,749]. Note that each model node specified will run each through these three lookback parameters as separate models I.e., total 9 models. (See comment for how to select the lookback parameters)

iii. **Specify the number of model runs** for each model architecture. Set the '*runs*' variable to a value greater than 1 if you would like to run a single model architecture multiple times. This can be useful to conduct a stability analysis to determine the variability of results under the different, random model initializations on each run.

iv. **Setting up training and validation sets** are described in the 'Training and Validation Set-up' comment of this cell. The variables *trainseg* and *valseg* needs to be set according to the length of the trial data and the lookback parameter. The *trainseg* variable describes how many minibatches of data would be used for training and *valseg* specifies how many minibatches would be used for validation. For example, a model with lookback parameter of 249 and data with trial length of 1500 samples, would produce 6 minibatches of input and output data sets for training (see comment explanation in cell). Of these 6 minibatches one can choose 4 for training (*trainseg* = 4) and 2 for validation (*valseg* = 2). NB. The current code is hard coded to specify the *trainseg* and *valseg* values for 3 different lookback parameters (249, 499, 749) according to our trial length of 1500 samples and preference. If you are utilizing another dataset this logic would need to be changed.

v. **Run the entire script**: Once these variables are set according to user specifications, the entire script can be run.

9. Once **Script 1 execution has been completed**, you may check the generated folders for each model in your main folder. In each model folder you will find the best saved model (.h5 file), training and validation loss parameters (.npy files) and a training and validation loss curve plot (.png file).
10. **Open 'Gait Signatures Script 2: Extract Internal Parameters.ipynb'**
11. **Ensure steps 5-8 above are repeated for this script 2, as in script 1:** paths and variables should be the same.
12. Once **Script 2 execution has been completed**, you may check each of the model folders in your main folder. In each folder you will find plots of the external and self-driven kinematics for each trial (.png files) [reference kinematics in blue, externally driven predicted kinematics before red vertical line time point and self-driven kinematics after the red line timepoint in green]. The predicted external and self-driven kinematics '*yourmodelname* _PredictiveKinematics.npy' and extracted external and self-driven corresponding internal parameter traces (Hs and Cs): '*yourmodelname*_extdriveHC.npy' and '*yourmodelname* _selfdriveHCs.npy' are also output from this script.
13. **Open 'Gait Signatures Script 3: Generate Gait Signatures.ipynb'**
14. **Ensure steps 5-8 above are repeated for this script 3, as in script 1 and 2:** paths and variables should be the same.
15. Once **Script 3 execution has been completed**, you may check each of the model folders in your main folder. In each folder you will find the generated gait signatures for all trials in file '*yourmodelname*_Gaitsignatures.npy' (full dimensional gait signatures) or 'modelname_Gaitsignatures_trunc6.npy' (6-dimensional gait signatures) along with other pre-processing outputs such as '*yourmodelname*_PhaseAvgPcs.npy', '*yourmodelname*_PhaseAvgPcs_shift.npy' and ' *yourmodelname*_PhaseVariables.npy'. The output gait signature file which was used to generate the results for our manuscript is 'modelname_Gaitsignatures_trunc6.npy' The file is a 3D array of shape (number of trials, 6 principal components, 100 phase samples).