David Ravnsborg

Supervised Learning of Vertical Ground Reaction Force for Predicting Gait Sequence

This paper is an exploration of using predictive modeling to predict gait patterns using a rolling/iterative forecast. Generating an accurate predictive model from gait data is a challenging task. While several channels of kinetic gait data were available for use - for instance, the vertical, lateral, and anteroposterior dimensions - only single leg vertical ground reaction force (VGRF) was used in this analysis. A VGRF time series is necessarily highly variable between individuals (like all measured gait parameters), has somewhat irregular rather than precise periodic cycles, and shows nonlinear behaviour between timesteps. The consequences of these properties are discussed in greater detail in Table 1. A Long Short Term Memory (LSTM) network was used for training and forecasting.

Table 1. Properties of VGRF.

Individually variable	Due to anthropomorphic differences between individuals (e.g. limb length, overall mass, neural and muscular factors underlying the generation of force, symmetry) the shape and cycles of the vertical ground reaction force will be exhibited differentially in a way which is difficult to correct for with a limited set of input parameters (e.g. age, height, weight). This means that any model trained on different individuals will necessarily introduce higher error than a model trained on the same individual. For simplicity, this motivated the use of one participant for model training and comparison purposes.
Non-periodic cycles	Cycles are irregular, meaning that models which rely on regular periodicity cannot accurately capture the temporal variability in the data. This limits some of the techniques that can be applied to transforming the data prior to training a model.
Nonlinear	Factors such as body position - particularly the position of center of mass in the anteroposterior or lateral dimensions relative to the position of the ground reaction force - have strong effects on future data points. This means that future vertical ground reaction force states are dependent upon more than the past states of vertical ground reaction force or the timestep.

August 28, 2017

The primary rationale for using an LSTM network was to capture the nonlinear correlations between time steps, and to easily account for the non-periodic cyclical pattern in VGRF data. LSTM networks excel at this kind of prediction. The use of logistic activation functions within the neurons responsible for integrating previous staters allows for the neural networks to account for some of the nonlinear behaviour. Backpropagation through time is used to continually tune neuron weights which allows for effective short term forecasting, while long term memory blocks allow for the preservation of the state of existing blocks as opposed to classic recurrent neural networks whose memory neurons undergo gradual extinction of previously learned state. Another model which was considered was ARIMA, although early tests using LSTM networks and ARIMA favored LSTM networks significantly for rolling forecasts, so subsequent work on model tuning and analysis was done using LSTM networks.

This document will present the process of selecting the best parameters for the model, and present the final results of the model. The gait data was split, using the first 80% for training and the subsequent 20% for testing. The python code used for this is available on github, as is the training data which was measured using the author of this paper as the participant and is available for re-use.

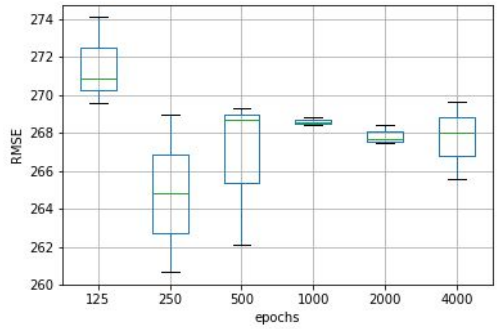
Tuning the Model

Root mean square error (RMSE) was used as the error function when comparing predicted to actual results on trained models. Multiple instances of the LSTM model were trained with varying values for data resampling rate, number of epochs to train the model on each set of inputs, batch size, and number of neurons. For each set of parameters, a model was trained 3 times, as the training of an LSTM relies on random processes and therefore is non-deterministic.

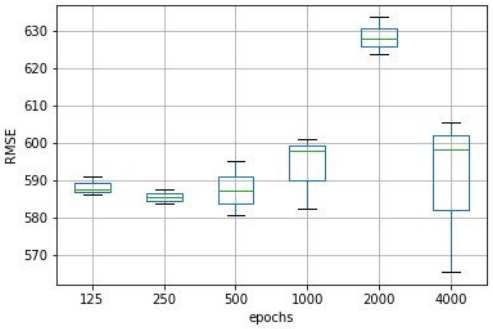
First, resampling rate and number of epochs were tested over the following ranges. Resampling rate: every 5th, 10th, 20th, 30th, and 40th value from the original dataset were

used. This was repeated using 125, 250, 500, 1000, 2000, and 4000 epochs at each resampling rate. The batch size was 4 and the number of neurons used for training was 1. The results are presented below, with each figure representing models trained and analyzed at a particular resampling rate under each of 5 epochs:

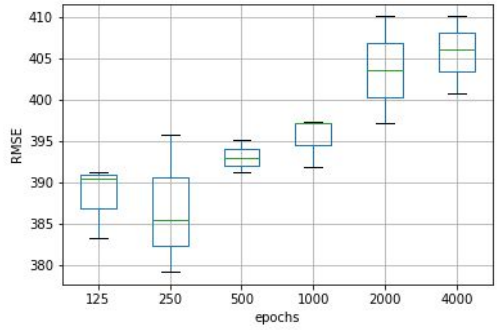
Resampling rate: every 5th



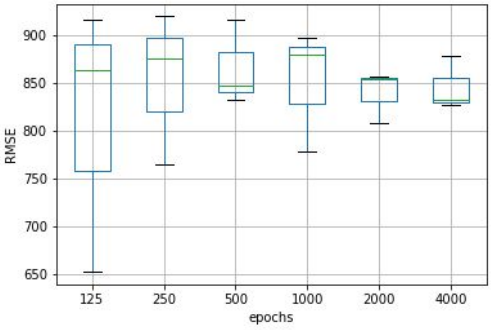
Resampling rate: every 30th



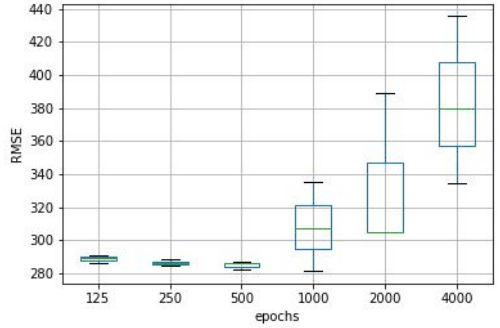
Resampling rate: every 10th



Resampling rate: every 40th

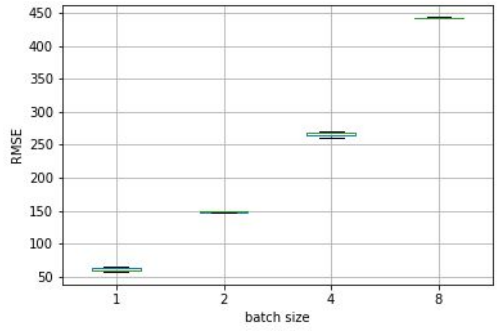


Resampling rate: every 20th

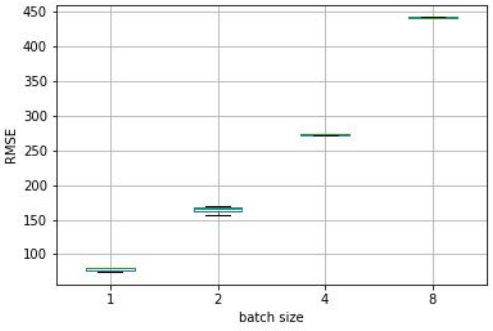


Next, number of neurons and batch size were tested over the following ranges. Number of neurons: 1, 2, 4, 8. This was repeated using batch sizes of 1, 2, 4, and 8 data points. Based upon the lowest RMSE from the previous set of tests, the resampling rate was set to every 5th value, and the epoch size was set to 250. The results are presented below, with each figure representing models trained and analyzed using a particular number of neurons under each of 5 batch sizes:

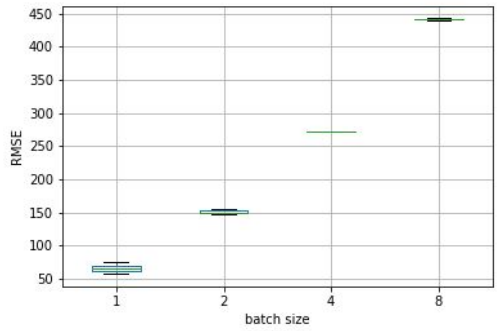
Neurons: 1



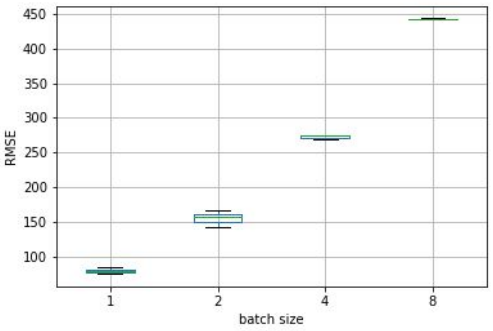
Neurons: 4



Neurons: 2



Neurons: 8



These results indicate that, using the tested parameters, the combination with the lowest RMSE (resampling rate=every 5th, epochs=250, neurons=1, batch size=1). It is worth noting that the time to train each model is increased by a higher number of neurons and a larger epoch size, while being decreased by a higher resampling rate and batch size. The training time of our model with the lowest RMSE was 1027 seconds using an Intel i7-4790k with 16

GB of ram. The model was trained using the keras framework with tensorflow and no additional support for multithreading. By contrast, the quickest model to train took 52 seconds and had the parameters (resampling rate=every 40th, epochs=125, neurons=1, batch size=4), while the longest running model took 4210 seconds and had the parameters (resampling rate=every 5th, epochs=4000, neurons=1, batch size=4).

The results of the best fit model fit using a rolling forecast (resampling rate=every 5th, epochs=250, neurons=1, batch size=1) can be seen below: