# Final Project Report

Yu Xiang

May 2018

## 1 Introduction

Current medical devices are overly sensitive to noise and variations in time series signals and suffer from high false alarm rates – even simple, normal patient movements and sensor/cable artefacts can trigger high-priority clinical alarms. Thus, their ability to accurately and meaningfully alert providers about urgent, clinically significant situations is limited. Our goal is to apply computational methods to analyze large and varied clinical datasets in order to generate models that can accurately identify, differentiate, and possibly predict clinically significant alarms directly from clinical datastreams.

## 2 Overview

This two-semester long research project in healthcare predictive was split into two parts. For the first semester, we worked on the preprocessing of data received from Rhode Island Hospital. Original time-stamped signals are stored as json objects in txt file. Human clinician experts also provided alarm csv files containing high priority alarms filtered in their end (Splunk), along with their human annotations on data sliced from a 10-min time window expanded around alarms. Data preprocessing was done by several python scripts to merge data generated from two consecutive days into a single csv file, extract good quality data with QoS value equal to 1 (qos = 1, 0 , -1 indicates good quality, indeterminate quality, poor quality, respectively), filter data using manual annotations to mark signals, sort and remove duplicates in data, etc. To prepare data for Machine Learning, features are transformed from vertical data to horizontal data, dictionary features are unpacked into separated columns, text features are encoded into numeric values.

For the second semester, we implemented both supervised learning model (SVM as an example) and deep learning models (LSTM and CNN as examples) on the cleaned data set. Our tasks are expanded around learning how machines/devices generate EKG alarms(how to repeat the work of emergency room alarms), learning how to determine whether signals like EKG, PPG are interpretable, and whether the QoS values are correct or not since all these

evaluations have an influence when we are predicting an alarm: if the datastreams are human-interpretable, the resulting alarms are probably real (since the threshold-driven monitor alarms were triggered by an aberrant value), and there is likely to be a quantifiable direct relationship. Clinician experts suggest that an alarm triggered at a given time stamp $t$ won't be related to the signals generated before time stamp $t$ - 5 min, or after time stamp $t$ + 5 min. Their annotations are also done within this time range. Our ultimate goal is to predict true, clinically significant alarms (of emergent/urgent/non-urgent severity).

# 3 Data

There are 3 types of original data:

- Orignal json files consisting of multiple json objects, each json object is an observation of live patient signals at a given timestamp processed by patient bedside monitors and acquired by research "listener" hardware/software devices. Each timestamp is a json object. Based on the data emission characteristics of the patient monitors studied (Philips Intelivue systems; e.g., 1-lead ECG at 250Hz sent out as 64 datapoints / 0.256s, which translates to 256 datapoints / 1.024s = 250Hz), patient physiologic data are exported as .json objects.

- Alarms csv files containing all the red(high priority) ECG, PPG(SpO2) and NIBP(blood pressure) time-stamped alarms of clinical interest for this research program, extracted with Splunk from the original .json files and including only the alarms of interest to the research program.

- Annotations txt files containing human clinician expert annotations of datastream window between [$t$ - 5 min, $t$ + 5 min] around timestamp $t$ of each alarm included in the alarms.csv file. For each subwindow (approx. 60 sec each), a window annotation is applied manually.

However, in the orignal json file, ECG signals (at 250HZ) and Pleth SpO2 signals (at 125 HZ) are sent in separate packages, for example, ECG signals in a 1-second time range is divided into 4 identical packages of data and sent in 4 consecutive time stamps. Not to mention that the these two signals, and other signals like Blood pressure, Heart rate come in a different frequency, which makes it difficult to convert the data into a learnable format for ML.

After discussion with clinician experts, they helped provide a new version of linearized data combined with human annotations, which is a csv file containing time-stamped signals along with signal annotations and alarms annotations. In this file, all ECG and PPG values are unpacked to their corresponding time stamps. Worthy to note is that each single file only contains one alarm, along with the [$t$ - 5 min, $t$ + 5 min] window of data expanded around the alarm.

# 4 Method

The first step is to prepare the waveform data set for our Machine Learning models. This involves framing the data set as a supervised learning problem and normalizing our input features. We know that alarms triggered at a given time stamp $t$ is usually a reflection of a time-window of signals prior to time stamp $t$. In order to reproduce machine-generated alarms, we frame the supervised learning problem as predicting the alarm at the current time stamp $t$ given the clinical signals measured at the prior time window $\big[t$ - *window_size,* $t\big)$. Thus we're given a Multi-variate Time Series Predicting problem. Unlike regression predictive modeling, time series also adds the complexity of a sequence dependence among the input variables, which makes it a difficult type of predictive modeling problem.

Recurrent neural networks is a powerful type of neural network designed to handle sequence dependence. The Long Short-Term Memory network or LSTM network is a type of recurrent neural network used in deep learning because very large architectures can be successfully trained. In this research project, I choose LSTM to tackle our predicting problems because it has the capability of remembering inputs from up to 1000 time steps in the past, which is an advantage for learning long sequences with long time lags. In our linearized post-processed data set, a 5-min window covers near 1500 records/time stamps, which also requires the use of LSTM.

Table4.1 Example supervised learning data format

| Time step | X | Y |
|---|---|---|
| 0 | 3 | 0 |
| 1 | 1 | 0 |
| 2 | 0 | 1 |
| 3 | 5 | 0 |
| 4 | 2 | 1 |

Before machine learning can be used, time series forecasting problems must be re-framed as supervised learning problems. From a sequence to pairs of input and output sequences. A time series is a sequence of numbers that are ordered by a time stamp. A supervised learning problem is comprised of input features X and output labels Y (see Table 4.1), such that an algorithm can learn how to predict the output labels from the input features. The *shift()* function can be used to create copies of column that are pushed forward or pulled back. This is the behavior required to create columns of lag observations as well as columns of forecast observations for a time series data set in a supervised learning format. Table 4.2 shows the example where we want to predict Y for time step $t$ based on observations of two previous time steps $t$-2 and $t$-1 (*window_size* = 2). We use the *shift()* function in Pandas to automatically create new framings of our

time series problem given the desired length (*window_size*) of input and output sequences. This is helpful because it allows us to explore different time window of our time series problem to see which might result in better performance.

| Table4.2 Data format after reframing | | | | | |
|---|---|---|---|---|---|
| | X(t-2) | Y(t-2) | X(t-1) | Y(t-1) | Y |
| 2 | 3 | 0 | 1 | 0 | 1 |
| 3 | 1 | 0 | 0 | 1 | 0 |
| 4 | 0 | 1 | 5 | 0 | 1 |

Taking both the capability of LSTM and our problem setting into consideration, we would explore a time window of up to 1500 input sequences for alarm predictions. Giving so many input features, it is essential for us to learn the importance of contribution of each input to the output labels. Convolutional neural networks excel at learning the spatial structure in input data. Time series signal data does have a one-dimensional spatial structure in the sequence of signals and the CNN may be able to pick out invariant features for true and false alarms. This learned spatial features may then be learned as sequences by an LSTM layer. We can easily add a one-dimensional CNN then feed the consolidated features to the LSTM. The combination of these two Deep learning models turns out to work perfectly with the machine-generated alarms.

Human clinician experts make annotations on EKG and SpO2/Pleth by looking at quality/noisiness and interpretability of alarm-generating signal (at resolution approx. 1 - 2 seconds), and review, correlation with remainder of datastreams within the 10-min window. For QoS values, expert review by looking at correctness of QoS for SpO2/Pleth relative to quality/noisiness and interpretability of waveform (at resolution approx. 1 - 2 seconds), serves as an examination of a single, experimental svd-based algorithm implementation. To define which signals are interpretable vs. not interpretable and represent analyses for clinical significance at the signal level, we also preprocessed the data into a window-series supervised learning format before applying LSTM to reproduce annotations on waveform data and QoS values.

# 5 Conclusion and Future Work

To learn how machines/devices generate ECG alarms, a combination of CNN and LSTM is used for our multivariate time series binary classification problem. A Convolutional layer along with a Max Pooling layer is added to the Neural Network before the consolidated features are fed into LSTM. Our results show that accuracy for this combined Deep Learning model is above 95.5% and it increases slighlty when we increase the *window_size* (number of time steps used for prediction). Since the proportion of of alarms and no-alarms in our dataset

is highly skewed to no-alarms (no-alarms : alarms = 27714 : 1942 = 100 : 7), we need to pay attention to the false rates (false positives and false negatives) as well. The confusion matrix in Table 5.1 indicates that the accuracy of predicting alarms is much lower than accuracy of predicting non-alarms.

| Table5.1 Confusion matrix (*window_size* = 200) | | |
|---|---|---|
| | Predict No-alarm | Predict Yes-alarm |
| Actual No-alarm | 27207 | 507 |
| Actual Yes-alarm | 436 | 1506 |

To determine what represents interpretable and meaningful EKG/PPG signals, and what represents correct QoS analysis, I implemented LSTM on EKG waveform values (250Hz) with interpretability annotations, SpO2 waveform values (125Hz) with interpretability annotations and QoS values (2 - 6 Hz) with correctness annotations, with *window_size* set to 1000, the testing accuracies are all above 98.05%. Table 5.2 shows how accuracies on QoS value correctness prediction changes with various window sizes. One can easily see that the peak accuracy is reached when *window_size* is 900.

| Table5.2 Accuracy vs window size on QoS correctness Prediction | |
|---|---|
| Window size<br>(# of time steps used for prediction) | Accuracy |
| 800 | 97.41% |
| 900 | 98.35% |
| 1000 | 98.05% |
| 1200 | 97.99% |
| 1300 | 97.59% |
| 1400 | 97.38% |

To predict true, clinically significant alarms (of emergent/urgent/non-urgent severity), one first need to detect the alarm first, and then can deal with the significance and severity. Thus I first tried to predict PPG alarms with the sliced data. The combined model of LSTM and CNN achieves a 100% accuracy since the dataset is small and pure(it only contains data from a [$t$ - 5 min, $t$ + 5 min] window around a PPG alarm triggered at timestamp $t$). We cannot merge 10-min data slices from current data folder generated by machine x00-08 because there exist huge gaps between each two 10-min windows sliced around alarms triggered at different time stamps, which makes the records discontinuous in time stamp if we concatenate them into one. Using data collected 2 hours ago along with some recent data to predict alarm at the current time stamp is definitely not a good idea.

Future work should focus on differentiating non-significant alarms vs significant alarms, and how to distinguish clinical significance vs severity. In order to differentiate significant alarms from non-significant alarms, we need more positive examples and false examples to make it sufficient for training. It would be better if these examples are timestamp-continuous to each other so that these data can be merged into a single dataframe to be fed into our Neural Network model. More effort should be made on clinical significance and severity comparisons. We might need to use more features like Heart Rate, SpO2, Blood Pressure as additional inputs to learn to distinguish these two terms. However, these signals come in a much less frequency compared to ECG/PPG waveforms, creating many NaN fields when the data is stored according to the frequency of ECG/PPG. We need to figure out if it is possible to do an interpolation between two time stamps' Blood pressure value to fill in the holes in between. In addition, the number of rows between two records with these fields having a number is not fixed, which makes the interpolating  filling method even harder.