Robustness of Multivariate Time Series Classification Models to Missing Data and Noise

Obradovic, Stefan; Yeh, Ian

Abstract

Dynamic multivariate time-dependent systems often rely on sensors to collect data. With multiple variables being recorded with time, we can better classify the state of the system at each time step. However, sensors are often unreliable and can result in missing or noisy data which can affect the robustness and sensitivity of supervised classification models. In this work we conduct a comparative analysis of three recurrent neural network based models namely, the simple recurrent neural network, the long short-term memory network, and the gated recurrent unit network while studying their performance under synthetically missing or corrupted data at incremental severities. We run experiments on a 13 label classification problem involving human activity recorded by mechanical sensors. It is demonstrated that the gated recurrent unit is more robust to changes to the size of the training set, the simple recurrent neural network is highly sensitive to noise and the long short-term memory network is robust to missing observations.

Section I. Introduction

Motivation. There are many potential applications of learning from multivariate temporal sequences. While considerable research has been conducted in univariate temporal learning **[1],** there is an abundance of multivariate temporal systems in nature and knowing how to learn from multivariate signals for the classification of a single temporal sequence can help produce more accurate predictions. For instance, the ability to reliably and accurately classify the state of a dynamic system over time using multiple time-dependent signals has significant implications for many fields such as disease progression and diagnosis **[2]**, mechanical systems **[3]**, and identifying stock market regimes **[4]** amongst other applications. However, with sensor-related temporal data collection, it is not uncommon for mechanical sensors to fail or break leading to the recording of significant noise, missing entries, and time step observations to be missing **[5]**. As such, it is important for temporal modeling systems to be robust to account for noisy and missing measurements and still draw functional classifications.

This paper focuses on answering the question of whether or not we can use recurrent neural network based supervised learning techniques to reliably classify a multivariate time dependent sequence. Specifically, we compare the accuracy of three proposed methods to classify 13 different types of physical activities for an individual wearing three inertial measurement units and a heart rate monitor. These proposed methods are the simple recurrent neural network (RNN), the long short-term memory (LSTM) RNN, and the gated recurrent unit (GRU) RNN. Additionally, we address the question of which of the three approaches perform best at classifying data under varying severity and types of hypothetical sensor malfunctions. To do so, we designed several experiments involving the random removal of data or addition of noise to conduct a comparative model analysis of classification performance under each of the modeled circumstances.

Related Work. There are many existing methods for the supervised classification designed for temporal sequences proposed in literature. The most common of which includes the recurrent neural network technique which can be further categorized into more specific models such as the simple RNN, the long short-term memory, and the gated recurrent unit. Each of these techniques include layers in a feed forward neural network that are adjacent connected to each other within the layer.

As proposed in **[6]**, the simple RNN model extends the basic neural network architecture by capturing sequential dependencies between inputs of the same layer through the addition of connections between adjacent nodes in a layer as well as nodes in the successive layer. The simple RNN technique attempts to address temporal limitations of feed-forward neural networks which struggle with data that is sequentially dependent **[7]**.

Long short-term memory models proposed in **[8]** extend the number and complexity gates included within each cell of the RNN. General LSTM architecture includes three gates: input, output, and forget gates. These gates function to retain long term dependencies in data through memory while updating with short term information. Using this memory mechanism, LSTMs improve upon simple RNN models by capturing not only short term dependencies but also long term ones **[7]**.

The LSTM was modified in **[9]** by reducing the number of gates and thereby model complexity. In GRU models, each node in the network contains two gates: the update and reset gates which work similarly to the gates in the LSTM model as the update gate decides what information is pertinent to be kept and the reset gate is utilized to decide what information can be forgotten. However, while GRU is less complex in order to avoid the problem of overfitting, due to this reduction it can suffer on tasks with very long sequential dependencies **[7]**.

Previous work has been done in identifying GRU limitations on different types of missing data and proposing new architectures to counteract these shortcomings as with the GRU-D model proposed in **[10]**. Instead, our study uniquely compares the robustness of different types of gated RNN cells against potential sensor data malfunctions. Other works have compared the performance of gated RNNs on different multivariate time-series classification tasks as in **[11]**. However, they do not compare model accuracy on synthetically missing data for these tasks. In this work we hope to understand the strengths and weaknesses of applying each cell type for different types and severity of data loss or corruption.

Hypothesis. Recurrent neural network based methods can function as reliable multivariable temporal classification models. We expect that the LSTM model will outperform GRU or simple RNN on uncorrupted data as it is a more complex model that can potentially learn longer distance dependencies on a larger dataset. Additionally, it will be more robust to noise and missing data because the memory gate will be able to retain previous uncorrupted signals. Comparatively, the GRU and simple RNN techniques will perform better on larger percentages of data corruption as they have less tendency to overfit.

Data. In order to conduct experiments with multivariate time-dependent data, we chose a dataset called PAMAP2: Physical Activity Monitoring. This dataset, originally compiled by Attila Reisss and Dider Stricker, was made to monitor different types of physical activity with sensors and fulfill a need for this type of data. The dataset has 18 different physical activities represented as classes and the different activities were performed at different times as well as for varying lengths of time. These activities included: lying, sitting, standing, walking, running, cycling, Nordic walking, watching TV, computer work, car driving, ascending stairs, descending stairs, vacuum cleaning, ironing, folding laundry, house cleaning, playing soccer, rope jumping, and transient activities [12]. Measurements of individuals were taken during a wide range of activities at various times for over ten hours worth of activity per individual.

In order to obtain the data, Reiss and Stricker relied on multiple Colibri Wireless Inertial Measurement Units (IMUs), which is a device that carries 3 sensors to measure acceleration, angular rate, and magnetic field. Orientation data was also recorded but reported to be invalid within the dataset. Three devices were placed on each participant, with one each on their chest,

ankle, and hand. The data collected by these devices for each individual was then merged with the data from the individual's heart rate monitor into one data file per subject per session while missing data due to wireless data dropping was also specifically annotated within the dataset using NaN. This dataset lends itself to the construction of a multivariate multilabel temporal classification problem where sensor data is used to predict the type of activity.

Objective. The objective of our analysis was to temporally classify the actions of a single individual for 13 possible action labels. Overall, the individual had 376,417 observations recorded, each with 41 separate sensor-based input features.

Section II. Experimental Design

Methods. For this paper, we decided to compare the performance of several different echo-state networks previously proposed in literature discussed above. While we initially attempted to also include an LSTM model with attention mechanism, we decided to reduce the scope of this paper to comparing a simple RNN model, a GRU RNN model, and an LSTM RNN model as it would be easier to control for biases in model architecture complexity.

Data Preparation. We began preparing the PAMAP2 dataset for multiclass temporal classification first by one-hot encoding the target variable, activityID, into 13 binary variables consisting of whether or not an observation corresponded to that class label. Then, to account for sensors recorded at different frequencies, we forward filled all NaN values present in the dataset. For instance, while all acceleration monitors recorded data each second, the heart rate monitor recorded beats in ten second intervals. We decided upon forward filling as a technique to not bias our results by interpolating from future data as well as because the simplicity of the method makes it feasibly implemented each second in real-time in case a sensor is unable to make a recording. Next we convert the data into a time-series classification problem by shifting all 40 input variables to the previous time step. Additionally, we shift each of the 13 one-hot encoded targets to the previous time step to be used as inputs in predicting each of the 13 targets in the current time step. We also make sure to normalize the inputs by scaling all features to a range between 0 and 1. From there we split the dataset into a training, validation, and testing set by selecting the last contiguous 110,000 observations for testing, the 40,000 previous contiguous observations for validation, and all remaining previous data as the training set depending on the size remaining due to each experimental procedure outlined below. We chose to use contiguous chunks for training, validation, and testing rather than random selection in order to preserve the temporal dependency of the data.

Model Parameters. In order to minimize model architecture bias, we decided to use the same architecture for each model across experiments, changing only the type of RNN nodes in the input layer. Each model contained an input layer accepting a tensor of all features for each observation for all previous time-steps and would output to 30 nodes (either simple RNN, GRU, or LSTM cells). The following layer consists of a single fully-connected dense layer with 13 output nodes where each corresponds to the predicted probability using softmax activation of the next time step corresponding to each class. Each model was trained for 10 epochs using categorical cross entropy loss, adam optimizer, and a batch size of 150. We also compared each of these methods to a benchmark naive model that would simply predict the previous time-step's class for the following time-step.

Computational Environment. All data models and experiments were implemented in Python 3.7, TensorFlow 2.2, Keras 2.3 and run on a MacOS v11.6 machine with 16 GB of memory and a 3.3 GHz Dual-Core Intel Core i7 processor.

Evaluation Metrics. We used two separate metrics to evaluate the performance of each model on out-of-sample testing data: classification accuracy and Root Mean Squared Error (RMSE). To test classification accuracy, we take the highest likelihood output from the 13 output nodes in the model as the predicted class and compare it to the class of the ground truth. To calculate RMSE we compare the output of the probability distribution computed over each class as an output of the softmax activation function to the ground truth. The intuition behind using this approach is that we want to not only predict the correct classification label but also predict it with high confidence such that all other class labels are given near-zero likelihood.

Experiment I. In order to simulate latency or mechanical errors in sensors that would result in missing time-steps, we generated a list of n unique random integers between 0 and the length of the entire dataset, where n corresponds to the amount of rows that would needed to be dropped for p% of data to be corrupted and then drop the corresponding rows before processing for the training, validation, and testing sets.

Experiment II. In order to simulate faulty sensors for individual recordings, we randomly generated a list of n unique integers encoding row and column indices, where n corresponds to the number of entries that would need to be dropped for p% of entries to be replaced by NaNs. Before splitting the dataset for training, validation, and testing, we forward fill these values to avoid issues in training.

Experiment III. In order to simulate poorly manufactured or faulty sensors that would produce noisy data recordings, we randomly generate a list of n unique integers encoding row and column indices, where n corresponds to the number of entries that would needed to be adjusted for p% of entries to be corrupted and add noise to the entry from a gaussian distribution with mean 0 and standard deviation 0.1.

For each experiment above, we evaluated the performance of the models under 0% to 40% corruption in increments of 5% and graph the training and testing accuracy as well as loss.

Section III. Results and Discussion

We present testing sample results of the chosen models in each of the three experiments.

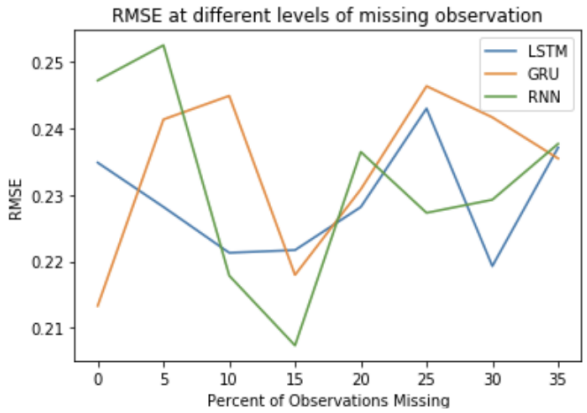


Exhibit 1a. Testing Set RMSE on experiment 1

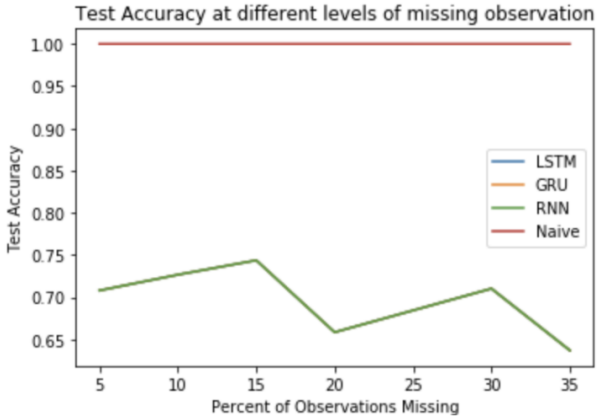
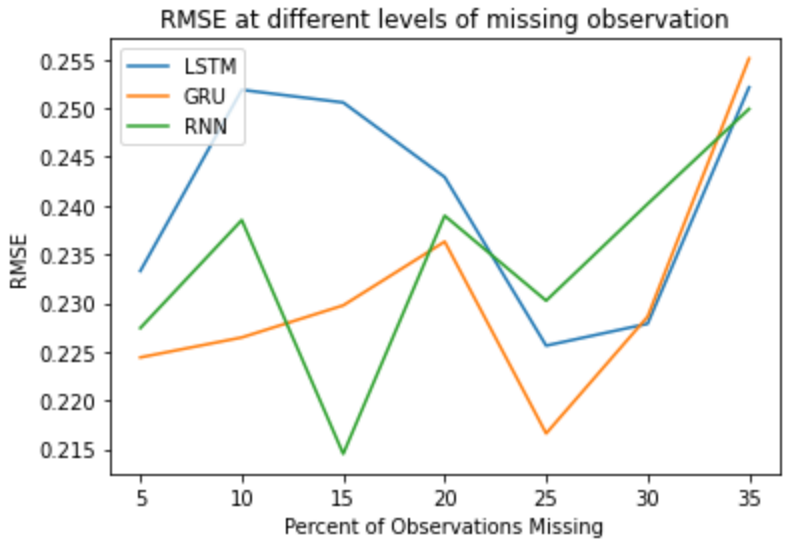


Exhibit 1b. Testing Set Classification Accuracy on Experiment 1



When comparing testing set RMSE performance for the first experiment of simulating latency or mechanical errors by dropping rows of data, we conclude that while the LSTM model performs most consistently across varying degrees of data corruption, the GRU model performs better than all 3 models in most trials. Upon analyzing the loss plots of the GRU compared to the other 2 models, we postulate that this is likely because GRU avoids overfitting to the training set better than the other two models. GRU cells are not as complicated as LSTM cells but just complex enough to capture temporal dependencies better than simple RNN.

In analyzing the RMSE performance for the second experiment in which we replace random data values with NaNs, we conclude that the LSTM model outperforms other methods in most trials. This is likely because simple RNN and GRU cells are more susceptible to missing observations as they lack memory mechanisms to retain long-term accurate data entries like that included in the LSTM cell.

For the third experiment where we added gaussian noise to random values in the dataset, we concluded that the LSTM model is most consistent at dealing with noisy inputs by retaining a relatively stable RMSE. On the other hand, we found that simple RNN is incredibly sensitive to noise which causes large fluctuations in the RMSE. This follows our intuition in terms of LSTM being able to store long term memory while the simple RNN has no such mechanism.

Upon doing further research, we realized that comparing classification models using accuracy has higher interpretability over using RMSE and we decided to conduct a secondary set of experiments in order to determine how many class predictions align with the ground truth. Additionally, we also decided to compare model performance for all three methods against a naive classifier to reason about the predictive benefits of highly complex models like RNNs over simple methods like the naive classifier. After evaluating model performance against the naive baseline classifier using classification accuracy instead of RMSE, we were able to identify potential difficulties with learning activity classifications in the PAMAP2 dataset. In each experiment comparing classification accuracy against the baseline, we concluded that the naive classifier was able to greatly outperform each of the echo-state networks with near perfect accuracy. After investigating this anomaly, we concluded that these results are largely due to the fact that the class labels do not fluctuate much temporally. Rather, the individual's activity

remains the same over longer periods of time, suddenly switches to another activity and then proceeds to remain that label for a similarly long period. As such, the baseline naive classifier results in an incredibly high classification accuracy (99.99%) as it accurately predicts all classes when they remain stable and only misclassified transitions between classes. This led to results where our more complicated learning models were severely overfitting the data and resulted in the LSTM and GRU RNN performing much worse than originally anticipated. We also attributed a decrease in training accuracy of each of the three models to an increase in the ratio of class switches to overall samples as the dataset decreased due to corruption.

After identifying this learning issue, we attempted to run experiments where instead of predicting the activity class, we reconstructed the classification problem to be the binary classification of whether or not the activity class changed in a given time-step; however, we were not able to collect enough non-zero labels to train models on this problem. We discuss potential solutions to this issue in the future work section below.

Section IV. Future Work

With respect to the experiments conducted in this paper, we have identified many potential avenues of future research to develop upon the ideas presented. First of all, the work outlined in this paper primarily compares the multivariate temporal classification performance of echo-state neural network based models such as simple RNN, GRU RNN, and LSTM RNN. However, more research can be done to compare the robustness of echo state networks against different types of time-series classification models. As an example, we could fit a multivariate support vector machine or decision tree classifier on a moving window of temporal observations.

Additionally, we only analyzed the robustness of each of the proposed models to varying degrees of random gaussian noise; however, there are many other different types of noise as outlined in [**13**] that are possible from sensor-related inputs that could affect not only the predictive attributes but also the target attributes as well.

Similarly, we only simulated mechanical sensor defects using two different types of missing data while multiple other types of possible missingness may occur from sensors due to communication failure, hardware damage, or power loss as mentioned in [**14**].

In order to handle the missing data entries, we also assumed that all individual missing entries would be forward filled by the previous observable entry. While this is the most likely mechanism by which a sensor would be able to deal with missing values in real time, there are several other imputation methods proposed in [**15**] that reduce information loss.

Furthermore, sensors may also break entirely resulting in a temporal signal ending abruptly throughout a stream of data. In order to simulate this potential behavior we could try to learn on a distribution where a feature or set of features stops transmitting part-way through the data stream. We could also test the severity of such a situation by changing the time at which a sensor stops recording data as well as the number of sensors that malfunction in this way.

Due to activation layer casualties with Keras, it would be worthwhile to also discuss how the activation layer could be implemented differently in order to avoid running into issues with Python implementation. This can be solved in many ways, including implementing this in different languages to see the effect that this could have, as well as creating our own implementation of neural networks. In addition, many of these issues with the activation layer came about due to the fact that the machines that everybody was working on were not standardized. Next time around, it is important to optimize those machines/maybe even have an AWS Cloud9 setup to ensure that everyone has a familiar experience with editing the code.

We also identified multiple potential improvements upon the results and methodology presented in this paper. First of all, as mentioned in our conclusion, we had issues obtaining meaningful echo-state classification models on the PAMAP2 dataset due to the low frequency at which the activity class changes over time. In order to properly evaluate the robustness of each model to the conducted experiments we would repeat the same experiments on a dataset where class labels change frequently over time.

Finally, in future work we would also consider expanding the evaluation metrics used to include area under the ROC curve as a measure of model performance which as noted in [**16**] has several superior properties over accuracy that make it a more favorable metric for the evaluation of machine learning models. Alternatively, we would consider using the Akaike Information Criterion (AIC) proposed by [**17**] to determine the trade-off between the goodness of fit of each model and the complexity of it.

Section V. References

[1] Fawaz, H. I., Forestier, G., Weber, J., Idoumghar, L., & Muller, P.-A. (2019). Deep learning for time series classification: A review. Data Mining and Knowledge Discovery, 33(4), 917–963. https://doi.org/10.1007/s10618-019-00619-1

[2] Wang, T., Qiu, R.G. & Yu, M. Predictive Modeling of the Progression of Alzheimer's Disease with Recurrent Neural Networks. Sci Rep 8, 9161 (2018). https://doi.org/10.1038/s41598-018-27337-w

[3] Li, C. J., & Yan, L. (1995). Mechanical system modelling using recurrent neural networks via quasi-Newton learning methods. Applied Mathematical Modelling, 19(7), 421–428. https://doi.org/10.1016/0307-904X(95)00015-C

[4] Ilhan, F., Karaahmetoglu, O., Balaban, I., & Kozat, S. S. (2020). Markovian rnn: An adaptive time series prediction network with hmm-based switching for nonstationary environments. ArXiv:2006.10119 [Cs, Stat]. http://arxiv.org/abs/2006.10119

[5] Teh, H.Y., Kempa-Liehr, A.W. & Wang, K.IK. Sensor data quality: a systematic review. J Big Data 7, 11 (2020). https://doi.org/10.1186/s40537-020-0285-1

[6] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1985). Learning internal representations by error propagation. CALIFORNIA UNIV SAN DIEGO LA JOLLA INST FOR COGNITIVE SCIENCE. https://apps.dtic.mil/sti/citations/ADA164453

[7] Salehinejad, H., Sankar, S., Barfett, J., Colak, E., & Valaee, S. (2018). Recent advances in recurrent neural networks. ArXiv:1801.01078 [Cs]. http://arxiv.org/abs/1801.01078

[8] Sepp Hochreiter, Jürgen Schmidhuber; Long Short-Term Memory. Neural Comput 1997; 9 (8): 1735–1780. doi: https://doi.org/10.1162/neco.1997.9.8.1735

[9] Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. ArXiv:1406.1078 [Cs, Stat]. http://arxiv.org/abs/1406.1078

[10] Che, Z., Purushotham, S., Cho, K., Sontag, D., & Liu, Y. (2018). Recurrent neural networks for multivariate time series with missing values. Scientific Reports, 8(1), 6085. https://doi.org/10.1038/s41598-018-24271-9

[11] Gallicchio, C., Micheli, A., & Pedrelli, L. (2019). Comparison between DeepESNs and gated RNNs on multivariate time-series prediction. ArXiv:1812.11527 [Cs, Stat]. http://arxiv.org/abs/1812.11527

[12] A. Reiss and D. Stricker. Introducing a New Benchmarked Dataset for Activity Monitoring. The 16th IEEE International Symposium on Wearable Computers (ISWC), 2012.
[13] Arslan, M., Guzel, M., Demirci, M., & Ozdemir, S. (2019, September). SMOTE and Gaussian Noise Based Sensor Data Augmentation. *2019 4th International Conference on Computer Science and Engineering (UBMK)*, 1–5. doi:10.1109/UBMK.2019.8907003
[14] Chen, S., & Yang, N. (2006). Congestion Avoidance Based on Lightweight Buffer Management in Sensor Networks. *IEEE Transactions on Parallel and Distributed Systems*, *17*(9), 934–946. doi:10.1109/TPDS.2006.115
[15] Jaques, N., Taylor, S., Sano, A., & Picard, R. (2017, Oktober). Multimodal autoencoder: A deep learning approach to filling in missing sensor data and enabling better mood prediction. *2017 Seventh International Conference on Affective Computing and Intelligent Interaction (ACII)*, 202–208. doi:10.1109/ACII.2017.8273601
[16] Herschtal, A., & Raskutti, B. (2004). Optimising Area under the ROC Curve Using Gradient Descent. *Proceedings of the Twenty-First International Conference on Machine Learning*, 49. Presented at the Banff, Alberta, Canada. doi:10.1145/1015330.1015366
[17] Cavanaugh, J. E., & Neath, A. A. (2019). The Akaike information criterion: Background, derivation, properties, application, interpretation, and refinements. *WIREs Computational Statistics*, *11*(3), e1460. doi:10.1002/wics.1460

Contribution Statement:
Stefan Obradovic (Approximate project contribution 75%) - For this project, I implemented and debugged all multivariate time series classification code including data preprocessing, train-test-splitting, and model evaluation as well as fully implementing the Simple RNN classifier, the GRU RNN classifier, the LSTM RNN classifier, and the Naive classifier. Additionally, I wrote code to set up all 3 experiments to produce testing set performance plots over varying amounts of data corruption and helped in running several of the experiments as they take well over 12 hours to run locally. I also conducted extensive research and literature review for the papers referenced in our paper as well as wrote significant portions of text for section II. Experimental Design, section III. Results and Discussion, and section IV. Future Work.

Ian Yeh (Approximate project contribution 25%) - For this project, I implemented the functionality of the three experiments to be run including removing rows, adding NaN values, and adding noise. I helped set up the experiments and ran most of them for upwards of 12+ hours per experiment. I helped Stefan in coding and debugging some of the model implementation. I contributed to finding and researching papers for the literature review and citations as well as wrote sizable portions of the final paper.