ANALYSIS OF WTTE-RNN VARIANTS THAT IMPROVE PERFORMANCE

Rory Cawley and Dr.John Burns

Department of Computing, Institute of Technology Tallaght, Ireland

ABSTRACT

Businesses typically have assets such as machinery, electronics or their customers. These assets share a common trait in that at some stage they will fail or, in the case of customers, they will churn. Knowing when and where to focus limited resources is a key area of concern for businesses. A prediction model called the WTTE-RNN was shown to be effective for predicting the time to event for topics such as machine failure. The purpose of this research is to identify neural network architecture variants of the WTTE-RNN model that have improved performance. The research results on these WTTE-RNN model variant would be useful in the application of the model.

1. INTRODUCTION

Predicting events such as machine failure or customer churn is a common business problem since the impact of being unable to do it is costly for businesses. The ability to predict events based on historical data means that a team of resources can be efficiently directed to maintain the resources that need to the most attention at any point in time.

The model called Weibull Time To Event - Recurrent Neural Network (or WTTE-RNN) (Martinsson, 2017) was designed to provide the predictions of the likelihood of events for the problems just described where its useful to get a predicted estimate of the time to an event such as machine failure. The model (see figure 1) is effective even when there is some gap or blind-spot in the data, i.e. its censored. Censored data is data which we are missing information on. For example we don't know when someone will die so we right-censor data. The WTTE-RNN model is a neural network that outputs a weibull distribution of the likelihood of an event occurring. The model allows for decision making about whether to do something about the events that are predicted in the near future or not.

In a real-life setting, businesses want to know which ma-chines are at risk of failure or which customer has the least probability of buying things. The WTTE-RNN model (Mar-tinsson, 2017) provides predictions of the time to an event. The purpose of the WTTE-RNN is to produce a prediction of"What will happen when". In the case of a business, the WTTE-RNN predictions can answer a question like"which customers/airplane engines should we be concerned about". Answering such a question will allow resources to be orga-nized to do engine maintenance or have customer success consultants seek to actively stop a customer from churning (i.e. customer churn prediction).

DOI:10.5121/mlaij.2019.6103

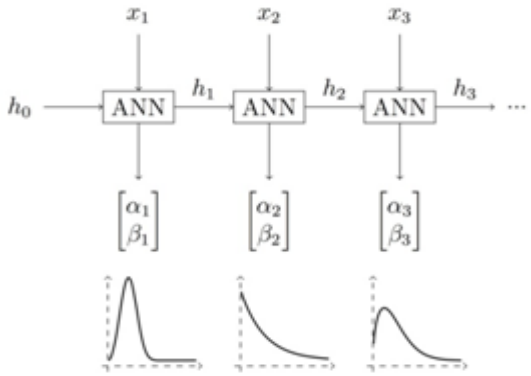

Fig. 1. The WTTE-RNN model - an RNN with a time to event prediction weibull distribution output

The original research (Martinsson, 2017) explored the ef-fectiveness of the WTTE-RNN but decided not to explore the effectiveness of different neural network architectures in the WTTE-RNN. The results presented here demonstrate several architectures that have improved performance for the WTTE-RNN, which is the objective of the research.

A. RESEARCH QUESTION

The objective of this research is to identify performance improvement opportunities to the original WTTE-RNN neural network architecture.

B. A TIME TO EVENT PREDICTION MODEL

Predicting the time to an event such as a machine failure or a customer churn is a difficult and important problem to solve. Martinsson (2017) defines these problems as equivalent to the problem of predicting the time between earthquakes or economic recessions, predicting the time a patient has left to live given treatment history, setting the premium for a life insurance or estimating the time-frame in which a customer is expected to make a repeat purchase.

2. LITERATURE REVIEW

A. UNDERSTANDING CONSUMER BEHAVIOR WITH RECURRENT NEURAL NETWORKS

Zalando, the online fashion retailer, has applied an RNN based model to the time series of customer interactions with their website to predict the probability that a user will place an order with the next time window of 7 days (Lang and Rettenmeier, 2017). Their system replaces a previous system based on hand crafted features.

B. CUSTOMER LIFETIME VALUE PREDICTION USING EMBEDDINGS

The research by (Chamberlain et al., 2017) describes"a customer lifetime value (CLTV) prediction system deployed at ASOS.com, a global fashion retailer. CLTV is an important problem in e-commerce where an accurate estimate of future value allows retailers to allocate marketing budget and focus on high-value customers." The recognised state of the art CLTV prediction system uses handcrafted features that are highly manual and need subject matter experts to create. The new system that learns feature representations automatically and the prediction model is a feed forward deep neural network based sliding window model.

C. PREDICTING CUSTOMER CHURN USING RECURRENT NEURAL NET-WORKS

Ljungehed (2017) says that there are established models to predict churn but the use of RNN is a largely unexplored approach. The advantages of an RNN in this problem is that they are robust to temporal noise and are suitable for sequential data. The focus of the research explores using an RNN for predicting churn based on customer lifetime value (CLV) time series.

D. DEEP LEARNING IN CUSTOMER CHURN PREDICTION: UNSUPER-VISED FEATURE LEARNING ON ABSTRACT COMPANY INDEPENDENT FEATURE VECTORS

Their tests have shown that the deep model was able to generate features using the hidden layers that are necessary for classification. They were able to increase their prediction accuracy from 73.2% to 77.9% so their conclusion was the multi-layer feed-forward neural network was effective in churn prediction (Spanoudes and Nguyen, 2017). Note that this solution doesn't apply to non-subscription companies.

E. WTTE-RNN: WEIBULL TIME TO EVENT RECURRENT NEURAL NETWORK

(Martinsson, 2017) created a time to event prediction model by combining survival analysis with a recurrent neural net-work. Survival analysis methods handle censored data, which means that if some of the observations in the data weren't available for the time period needed e.g. if you want to look at a year of churn for customers but some of them have only been signed up for a couple of months. Using survival analysis Martinsson (2017) was able to avoid the typical sliding time window technique for prediction where only the most recent data is factored into the prediction.

3. PERFORMANCE ANALYSIS OF WTTE-RNN MODEL VARIANTS

A. THE VARIANTS OF THE WTTE-RNN MODEL BEING ANALYZED

Model Variant	Neural Network Architecture						
Original WTTE-RNN	Layer Type	LSTM	FC	FC			
	Layer Width	100	10	2			
Wide LSTM	Layer Type	LSTM	FC				
	Layer Width	150	2				
Wide GRU	Layer Type	GRU	FC				
	Layer Width	150	2				
Stacked LSTM 30	Layer Type	LSTM	LSTM	LSTM	FC		
	Layer Width	30	30	30	2		
Stacked GRU 30	Layer Type	GRU	GRU	GRU	FC		
	Layer Width	30	30	30	2		
Stacked LSTM 50	Layer Type	LSTM	LSTM	LSTM	FC		
	Layer Width	50	50	50	2		
Stacked GRU 50	Layer Type	GRU	GRU	GRU	FC		
	Layer Width	50	50	50	2		
Stacked LSTM 100	Layer Type	LSTM	LSTM	LSTM	FC	FC	
	Layer Width	100	100	100	10	2	
Stacked LSTM Stacked Dense	Layer Type	LSTM	LSTM	LSTM	FC	FC	FC
	Layer Width	30	30	30	30	30	2
Stacked GRU Stacked Dense	Layer Type	GRU	GRU	GRU	FC	FC	FC
	Layer Width	30	30	30	30	30	2
Phased LSTM	Layer Type	Phased LSTM	FC				
	Layer Width	2	2				

Table I Wtte-Rnn Neural Network Architecture Model Variants

The original research on the WTTE-RNN (Martinsson, 2017) didn't examine different architectures for the neural network. This research will propose variants of the WTTE-RNN model with different neural network architectures. The table I provides the list of 10 neural network architectures that were assessed along with the architecture from the original research that was tested as a baseline to compare with. In the table I, LSTM is a Long Short-Term Memory layer type, GRU is a Gated Recurrent Unit layer type and FC is a Fully Connected layer type or a plain node layer type.

B. METHODOLOGY

This research follows the CRISP data mining methodology (Wirth and Hipp, 2000), which is a good match for the iterative nature of working with predictive models based on neural networks since they need a lot of experimental tuning in order to achieve the desired results.

C. DATASET USED FOR THE ANALYSIS

In the comparisons between the proposed variants of the WTTE-RNN model, the data that was used in the original research by Martinsson (2017) was used. The original WTTE-RNN model was reproduced in order to compare the predictive performance with the proposed WTTE-RNN model variants.

The data used in this research is from the same NASA data set of simulated jet engines running to failure (Martinsson, 2017). The data is in the time-series form, which involves a sequence of data points placed in temporal order. Each data point in the sequence has 26 features, see table II. The data sets consist of multiple multivariate time series. The data set is divided into training and test subsets. Each time series is from a different engine, i.e. the data can be considered to be from a fleet of engines of the same type. Each engine starts with different degrees of initial wear and manufacturing variation which is unknown to the user. This wear and variation is considered normal, i.e. it is not considered a fault condition.

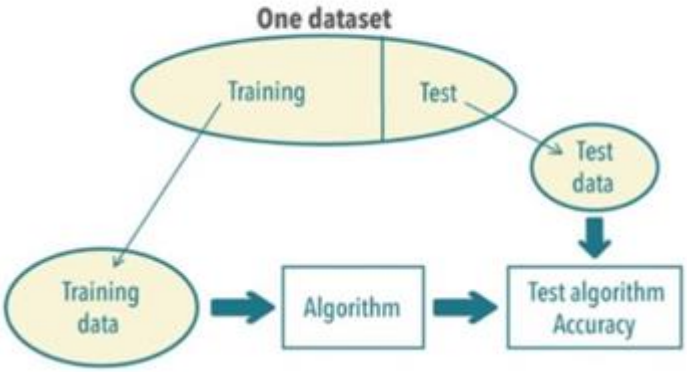

Fig. 2. Model is trained on training data and tested for prediction accuracy using test data

Once the model has been created it needs to be validated, using data it hasn't seen before, i.e. test data, and it should perform as accurately on the test data as on the training data (see figure 2) if the model hasn't been trained to over-fit on the training data.

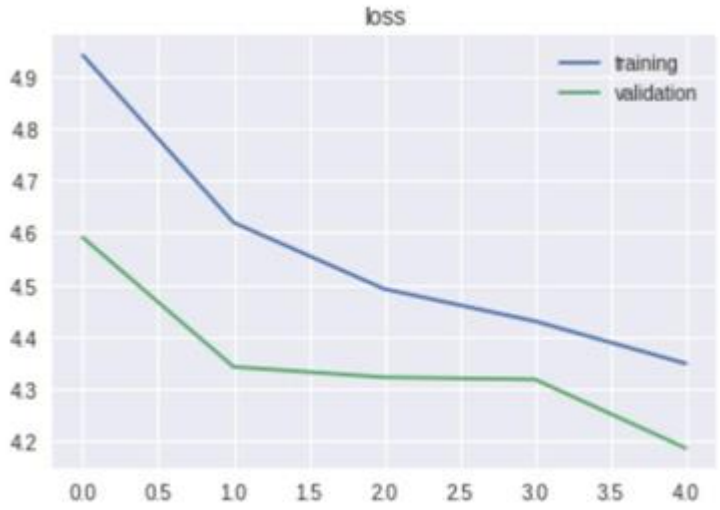

Fig. 3. Loss training vs. validation

The training set contains 20k rows of data from 100 unique engines that have run to failure. Each engine starts with different degrees of initial wear and manufacturing variation which is unknown to the user. This wear and variation is considered normal, i.e. it is not considered a fault condition.

The engine is operating normally at the start of each time series, and develops a fault at some point during the series. In the training set, the fault grows in magnitude until system failure. In the test set, the time series ends at some time prior to system failure. The objective is to predict the number of remaining operational cycles before failure in the test set, i.e. the number of operational cycles after the last cycle that the engine will continue to operate. A vector of the true remaining useful life (RUL) cycles for the test data is provided.

Index	Data Field	Data Type	Data Description
1	Id	Integer	Aircraft engine identifier, range [1,100]
2	Cycle	Integer	Time, in cycles
3	Setting1	Double	Operational setting 1
4	Setting2	Double	Operational setting 2
5	Setting3	Double	Operational setting 3
6	S1	Double	Sensor measurement 1
7	S2	Double	Sensor measurement 2
...	...	...	...
26	S21	Double	Sensor measurement 21

Table II Structure of The Data For The Engine Observations (Both Fortraining And Test Data Sets)

The implicit assumption of modeling data is that the asset of interest has a progressing degradation pattern, which is reflected in the asset's sensor measurements. By examining the asset's sensor values over time, the machine learning algorithm can learn the relationship between the sensor values and changes in sensor values to the historical failures in order to predict when failures will occur in the future.

Fig. 4. Example of the time series of a single sensor in a single engine during the full cycle duration

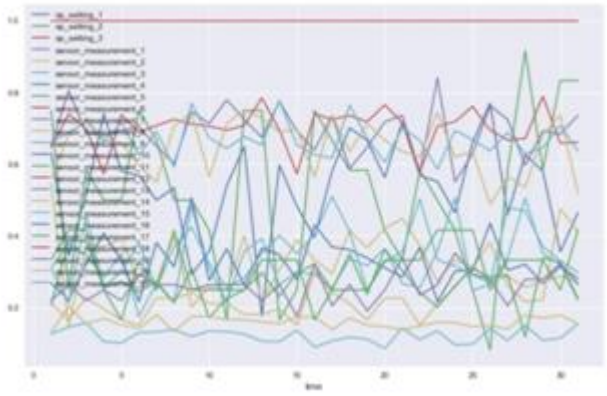

Fig. 5. Illustration of the combined set of sensor readings for a single engine

Each of the sensors has its own independent time series. An example time series of one sensor for one engine is in figure 4 while the exact type of sensor is unknown and the time is measured in an abstract 'cycles' quantity, the data doesn't seem to have any humanly comprehensible pattern that could be used in prediction. The problem of prediction becomes exponentially difficult when several of these independent time series need to be factored together to make a prediction. The chaotic looking figure 5 is an illustration of the enormity of the task of simultaneously finding patterns in all of the sensors at once that indicate a progression towards failure.

The problem of predicting events with a mass of separate time series is impossible for a human and is an extraordinary accomplishment for a machine.

The data was updated before being processed by the model. Any constant features were removed and the data was normal-ized.

D. EXPERIMENT ENVIRONMENT

Google Collaboratory 1 was the environment used for this research with the software setup in table III. The environment provides high specification hardware CPU (Intel Xeon), GPU (Nvidia Tesla K80) and 10GB RAM. Ive included a link to the notebooks that were used during my analysis 2.

Software Name	Usage	Version
Python	Programming language for scripting	3.6.3
Keras	Deep Learning API	2.1.6
Tensorflow	Tensor processing API	1.7.0
NumPy	Array processing for numbers, strings, records, and objects	1.14.3
Pandas	Powerful data structures for data analysis, time series,and statistics	0.22.0
GNU/Linux	Operating system	4.4.111

Table III Software Used To Conduct The Analysis (On Google Collaboratory Environment)

E. NEURAL NETWORK ARCHITECTURE EXPERIMENTS

Finding the best architectures for neural networks is largely done through trial and error. There is no sure way to instantly find the right number of layers or the right size for each layer. This research will evaluate a set of different architectures to create WTTE-RNN model variants in order to find architec-tures that improve upon the original WTTE-RNN (Martinsson, 2017).

RNN based neural networks are time dependent and they extract meaning from the order of the data for their representa-tion so they are suitable for use with the data in this research, which consists of sequences of time-stamped inputs. The most common variety of RNNs used are the LSTM and GRU. An additional experimental RNN model called ”Phased LSTM” was also considered (Neil, Pfeiffer, and Liu, 2016).

The original model used a single LSTM layer with 100 nodes so that is taken as the baseline architecture for this research. That model was reproduced and tested with all of the other architectures. Overall, 10 deep learning architectures were tested including one architecture with an exotic ”Phased LSTM” layer that specializes in very long time steps called (Neil, Pfeiffer, and Liu, 2016). Table I shows the architecture of the different model variants. Each layer is made up of Phased LTSM nodes, LSTM (Long Short Term Memory) nodes, GRU (Gated Recurrent Unit) nodes or FC (Fully Connected Neuron) nodes. The number of nodes configured for each layer is shown.

A sufficiently large single hidden layer multilayer percep-tron can be used to approximate most functions (Hornik, Stinchcombe, and White, 1989). According to (Hermans and Schrauwen, 2013) each layer processes some part of the task we wish to solve and passes it to the next layer like a processing pipeline. According to (Pascanu et al., 2013) a hierarchical model can be exponentially more efficient at representing some functions than a shallow one.

The approach used for defining the architectures for the WTTE-RNN model variants was to use permutations of layer type, number of layers and nodes per layer. The list of architectures is not an exhaustive list since that is an infinite set but its a good representation of architectures that were iterated upon and tested within the time constraints of the research. Once an architecture was created it was trained for 100 epochs on the data and monitored to establish the point (i.e. the epoch) when it converges without over-fitting on the training data (see figure 3), which would reduce the accuracy of generalizing its predictions using data it hadn’t seen before.

[1]https://colab.research.google.com
[2]https://github.com/rorycawley/MSc-Dissertation

When considering the number of nodes in a layer for the architectures, advice was taken from Heaton (2010), which suggests a number between the number of inputs and the number of outputs for hidden layers. When considering the number of layers to be used for the architectures successive attempts were made to train the different variations with a number of layers that was the same for each of the variants and a figure of 3 layers was arrived at for the RNN layers.

The number of training parameters increases enormously for layers with more nodes. Overall the stacked narrow ar-chitectures are found to have the least number of trainable parameters and the architectures with wider layers have the greater number of trainable parameters, see figure 6.

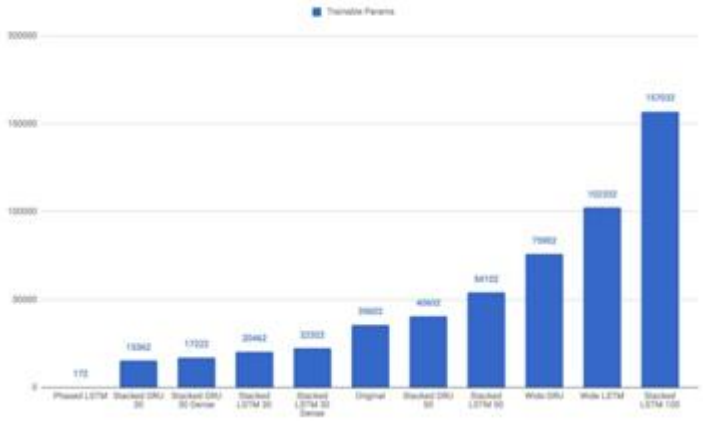

Fig. 6. Number of trainable parameters for each of the WTTE-RNN model variants

When making the choice between the different node types, primarily the LSTM and GRU, it can depend on the data set. The GRU node has several advantages over the LSTM node, it uses less training parameters, less memory and it trains and executes faster. The research by Rana (2016) points to using LSTMs if the data uses really long and complex sequences or accuracy is very critical to the project and use GRUs if theres a concern about training time, execution time, and memory consumption.

Fig. 7. Number of trainable parameters for each of the WTTE-RNN model variants

F. MODEL TRAINING

The model variants were initially trained for 100 epochs (epoch is a single pass over the entire training set) with a batch size of 128 samples (after each mini-batch the network weights are updated and state reset). The training rate wasnt altered from the default Keras library value. Training was monitored to see the when the model variant converged and if there was an earlier point of convergence in the training then that epoch number was chosen and the model was

retrained. The training of a model was iterated upon until a satisfactory training, testing, prediction, evaluation pipeline completed, see figure 6.
The loss function for the model is the original WTTE-RNN one, which has a formula for calculating the ”log-likelihood” for censored time-to-event data. The model’s cost function’s error is high when the model is predicting high probabilities of events during the known event-free lifetime.

LSTMs can quickly converge and even over-fit on some sequence prediction problems. To counter this, regularization methods can be used. Dropout randomly skips neurons during training, forcing others in the layer to engage. The recurrent layers (i.e. GRU or LSTM) were configured with the ’tanh’ activation and a recurrent dropout values of 0.25.

G. METHOD FOR THE EVALUATION OF WTTE-RNN MODEL VARIANTS

Because the WTTE-RNN is neither a classification nor a regression problem the mode value (see figure 8) of the asymmetrical output distribution (see figure 9) is used to convert the output into a regression output, which can then be used in a simple RMSE comparison.

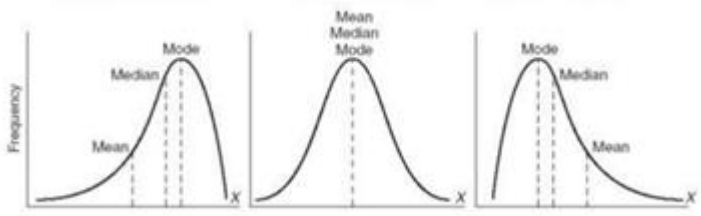

Fig. 8. Illustration showing that mode is a best approximation of predicted value since it’s closest to the weighted peak for asymmetrical distributions

Separate testing data was used to validate the trained models, it has the same data schema as the training data but the only difference is that the data does not indicate when the failure occurs (in other words, the last time period does not represent the failure point unlike in the training data). The number of remaining working cycles for the engines in the testing data is provided in a separate file so it can be compared with the model’s time to event predictions.

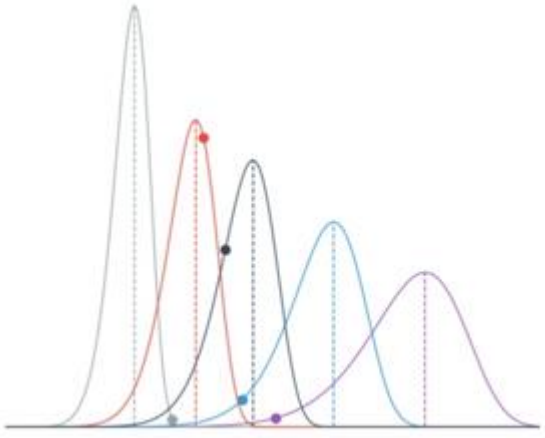

Fig. 9. Visualizations of weibull distributions generated by the WTTE-RNN showing the mode as a vertical dotted line

4. RESULTS OF PERFORMANCE ANALYSIS

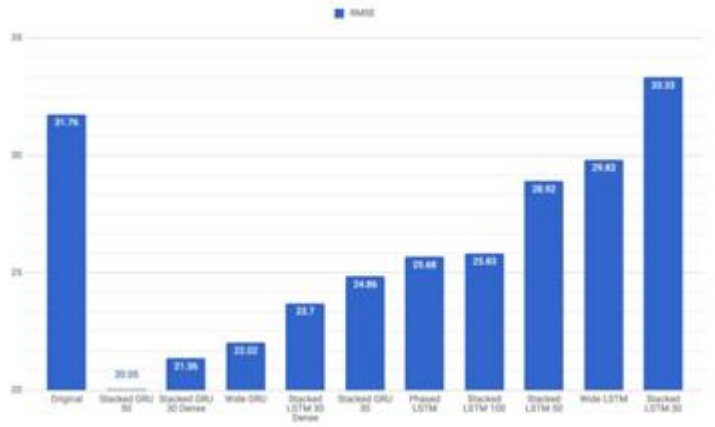

Fig. 10. RMSE based performance comparison of original WTTE-RNN with the model variants analyzed in this research (lower RMSE means more accurate prediction)

This chart, figure 10, compares the RMSE of the WTTE-RNN model variants, taking as input the mode of the predicted engine failure events and the actual engine failure events of the test data. The"Stacked GRU 50" model variant performed best since it had the lowest RMSE score. Importantly the chart shows clearly that every single model variant except one performed better than the original model.

Almost all of the GRU based architectures had a better pre-diction performance than the LSTM based architectures. This is an interesting finding since GRUs are deemed to be faster to train, faster to execute but not necessarily more accurate than LSTMs (Rana, 2016). Amongst both the stacked GRU and stacked architectures LSTM, having a wider layer resulted in improved performance, probably because of the increased representational power of the layer due to the extra nodes. Additionally the "Phased LSTM", which showed reasonable mid-pack performance, experienced the smalled training time due at least in part to the very small number of training parameters.

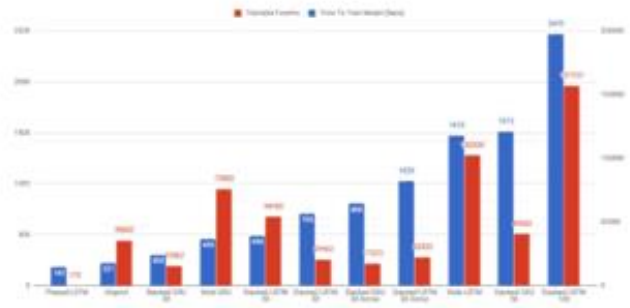

Fig. 11. Time to train shows with the number of trainable parameters for WTTE-RNN model variants

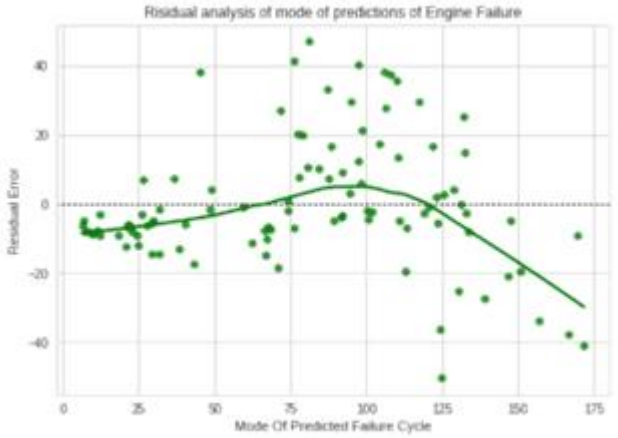

Fig. 12. Residual diagnostic test for the ”Stacked GRU 50” based model (straight line better)

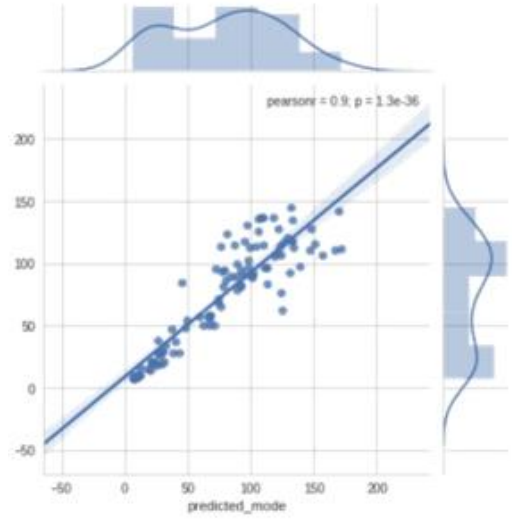

Fig. 13. Plot of correlation between the mode of the prediction and true value (T) for the”Stacked GRU 50” based model

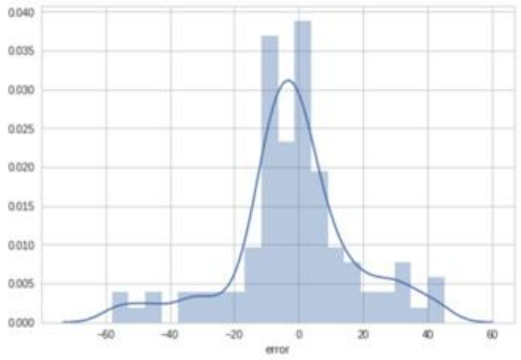

Fig. 14. Plot of the distribution of error values as calculated by subtracting the mode values of the predicted distributions from the true values for the ”Stacked GRU 50” based model

By increasing the width of the layers (by adding additional nodes) additional connections are made between the nodes in the different layers. Each connection has a weight and is involved in the training calculations. Wider model variants take longer to train as a result of the number of additional trainable parameters, figure 11. If training time is a concern then favoring a stacked architecture with less nodes is preferred. The highest performing ”Stacked GRU 50” based model did experience one of the longest training time, taking the second place after the ”Stacked LSTM 100” based model. In a scenario where very large datasets are used then this information on the length of the training time should be given careful consideration when considering the use of the ”Stacked GRU 50” based WTTE-RNN model.

Another interesting observation is that the ”Stacked GRU 30” model variant has practically the same RMSE but 20% of the training time of the ”Stacked GRU 50”. Unless there’s a call to squeeze the last piece of performance out of the models then ”Stacked GRU 30” is preferable to use due to its very short training time. Note that since each of the models converged at a different number of epochs this is also a factor in the training time.

This chart, figure 12, is a residual analysis of the predicted values and the actual failure cycle times for the ”Stacked GRU 50” based model. The central thick line should follow the 0 residual horizontal. The values have some clustering close to the line. There is a small parabola effect that is more pronounced for predictions that have higher failure cycle times i.e. greater than 150. The residual plot shows a nonlinear relationship and that predictions for events that occur farthest in the future are less reliable. This is likely related to the neural network look-back period of 100 steps that was set based on the assumption that electronic components work well for a long time and then suddenly they start decaying rapidly over a short time frame.

The chart, figure 13, shows the predicted mode values against the actual true values for the ”Stacked GRU 50” based model test, they have a close alignment to the diagonal line, which is the optimal prediction. The Pearson correlation coefficient value is close to 1 showing an almost exact linear relationship. The p-value is close to 0, this indicates that its extremely low probability that the Pearson’s r value was produced by an uncorrelated dataset. Also, there is a similar effect to what the residual plot shows, the further away the predicted failure cycle is from the present the less accurate it is. This makes sense since it’s more accurate to predict nearer term events and with electronic sensors, they tend to fail fast rather than over a protracted time period. In addition the histogram for the predicted values, at the top of the box matches well with the histogram on the right of the true values, this means that the prediction were of good quality.

This chart, figure 14, shows the distribution of prediction errors for the ”Stacked GRU 50” based model (the true value minus the predicted value). The histogram shows that most predictions are centered around 0 meaning the predictions were close to the real values.

5. CONCLUSION

Models based on GRU layers were best. The results also showed that model training time, if that is a factor in the decision making to choose a prediction model, is dependent not only on the number of training parameters but also on the time it takes for the model to converge without experiencing over-fitting on the training data.

The contribution this research makes is to assist with the practical applicability of this model by showcasing a set of proven to be effective neural network architectures. The experiments showed a wide distribution of the time needed for the model variants to converge.

The GRU based models outperformed the LSTM based models and the wider stacked GRU performed best. GRU based models werent the fasted to converge in training as expected but that may have been something specific to the training set. Attention should be given to the Phased LSTM node type since it provided high levels of accuracy for the relatively small training time.

This information is useful for engineers or scientists looking to apply the WTTE-RNN to a problem and need to consider what architectures to use. This research will inform on the trade-offs and also provide a method to compare the perfor-mance of architecture variations.

6. FUTURE WORK

Further work in using different architecures such as a having convolutional layers before the recurrent layer since they've been show to facilitate the recognition of patterns (Wu et al., 2018). Additionally, using a deep learning introspection tool such as LIME (Ribeiro, Singh, and Guestrin, 2016) could help examine the issues with exploding gradient.

REFERENCES

[1] Chamberlain, Benjamin Paul et al. (2017). "Customer lifetime value prediction using embeddings". In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowl-edge Discovery and Data Mining. ACM, pp. 1753–1762.

[2] Heaton, Jeff (2010). Programming neural networks with Encog 2 in Java. Heaton Research, Inc.

[3] Hermans, Michiel and Benjamin Schrauwen (2013). "Training and analysing deep recurrent neural networks". In: Advances in neural information processing systems, pp. 190–198.

[4] Hornik, Kurt, Maxwell Stinchcombe, and Halbert White (1989). "Multilayer feedforward networks are universal ap-proximators". In: Neural networks 2.5, pp. 359–366.

[5] Lang, Tobias and Matthias Rettenmeier (2017). "Under-standing Consumer Behavior with Recurrent Neural Net-works". In: Proceedings of the 3rd Workshop on Machine Learning Methods for Recommender Systems. http://mlrec. org/2017/papers/paper2. pdf.

[6] Ljungehed, Jesper (2017). "Predicting Customer Churn Using Recurrent Neural Networks". MA thesis.

[7] Martinsson, Egil (2017). "WTTE-RNN: Weibull Time To Event Recurrent Neural Network". In:

[8] Neil, Daniel, Michael Pfeiffer, and Shih-Chii Liu (2016). "Phased lstm: Accelerating recurrent network training for long or event-based sequences". In: Advances in Neural Information Processing Systems, pp. 3882–3890.

[9] Pascanu, Razvan et al. (2013). "How to construct deep recur-rent neural networks". In: arXiv preprint arXiv:1312.6026.

[10] Rana, Rajib (2016). "Gated Recurrent Unit (GRU) for Emo-tion Classification from Noisy Speech". In: arXiv preprint arXiv:1612.07778.

[11] Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin (2016). "Why should i trust you?: Explaining the predic-tions of any classifier". In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discov-ery and Data Mining. ACM, pp. 1135–1144.

[12] Spanoudes, Philip and Thomson Nguyen (2017). "Deep Learn-ing in Customer Churn Prediction: Unsupervised Feature Learning on Abstract Company Independent Feature Vec-tors". In: arXiv preprint arXiv:1703.03869.

[13] Wirth, Rudiger¨ and Jochen Hipp (2000). "CRISP-DM: To-wards a standard process model for data mining". In: Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining. Citeseer, pp. 29–39.

[14] Wu, Yuankai et al. (2018). "A hybrid deep learning based traffic flow prediction method and its understanding". In: Transportation Research Part C: Emerging Technologies 90, pp. 166–180.