Recurrent Neural Network-based Fault Detector for Aileron Failures of Aircraft

Nobuyuki Yoshikawa, Nacim Belkhir, Sinji Suzuki

Recurrent Neural Network-based Fault Detector for Aileron Failures of Aircraft

Nobuyuki Yoshikawa[1], Nacim Belkhir[2] and Sinji Suzuki[3]

***Abstract*— This paper empirically investigate the design of a fault detection mechanism based on Long Short Term Memory (LSTM) neural network. Given an equation based model that approximate the behavior of aircraft ailerons, the fault detector aims at predicting the state of aircraft: the normal state for which no failure are observed, or four different failure states, e.g. a delay changes. This is achieved by collecting a limited amount of command and responses data by varying the parameters of the aileron model, such that a LSTM network is used to predict the state of the aircraft of sequence of the pair commands/responses. In this empirical study we empirically demonstrated LSTM networks can be a promising approach for fault detection, and achieve reasonable performances despite a limited amount of data, in particular avoiding overfitting of the model.**

I. INTRODUCTION

In recent year, the air traffic is growing by more 5 percent per year [1]. At same time, the number of flight is increasing and it causes the shortage of the pilots. It is not easy to get new additional pilots. It is estimated that the rate will keep higher because globalization and the developing countries growth. Pilot shortage is a severe problem for the airlines. Therefore the single pilot flight is required. On the other hand, the accident rate of the air transportation is reduced. The rate reduction is caused by not only the advance of the aviation technology but also the operation by the multiple pilots, especially in the passenger flights. In the multiple pilot flight, each pilots monitor each other handling to avoid miss understand and handling while all the flight procedure.

When severe failure happens on flight, the aircraft characteristics can be changed. The pilots in the flight such the sever failure are stressed harder than normal flight without failure. The large stress may cause mistake in the operation and the handling. The multiple pilot flights tries to keep the low accident rate by monitoring each other.

Accordingly some operation support system is required to realize the single pilot flight. Therefore we propose the fault tolerant control system. In this paper, the fault detector by the recurrent neural network for the aileron is developed as a first step to develop the fault tolerant control system.

Fault detection (FD) using Neural Network (NN) has been proposed in some studies [2], [3]. By using NN, it is easy to handle nonlinear model. Aileron model is a nonlinear model which has dead time and variable time constant, and there is a time dependent relationship between commands and responses of ailerons.

While NN, has been widely investigated in the control system community, recurrent neural networks [3], [4], [5] remains under explored. Nonetheless, if we consider a failure state as a state that is dependent on the time, recurrent neural network seems the most appropriate approach to use, as they are particularly suited to deal with temporal/sequential data. After several successes in different research fields, where input data can also be limited, recurrent neural network and in particular LSTM [6], a variant of RNN are a method to be explored.

For this reason, a recurrent neural network was adapted as an aileron fault detector. In this paper, the idea of time dependent fault detection and isolation of ailerons that strong nonlinear model using recurrent neural network is empirically shown.

Section II introduces the aileron model that will be used to generate input data for learning a RNN. Next in Section III we give a brief overview of fault detector in the control system community. In Section IV we defines recurrent neural networks and their mechanisms. Next, we present the experimental setting and protocol of our empirical study in Section V. In Section VI, we present typical results of our empirical study. This is followed in Section VII by discussion on the results and highlights the main limitation. Finally, we conclude and give a suggestion of further works to extend this empirical study in Section VIII.

II. AILERON MODEL

A control target of our fault tolerant control system is the Multiple Aviation Laboratory α (MuPAL-α) that belongs to the Japan Aerospace Exploration Agency (JAXA).We planned to divide the development into 2 stage. This study is a first step of the development. Therefore MuPAL-α is used as a fault detection target. The differential equation of the aileron of MuPAL-α is written as bellow:

$$\dot{\delta_a}(t) = \frac{1}{\tau}(K\delta_{a_c}(t-T) - \delta_a(t)) \tag{1}$$

where $\delta_a(t)$ is the aileron angle, $\delta_{a_c}(t)$ is the aileron angle command, $\tau = 0.12$ sec is the time constant, $K = 0.67$ is the gain, T is the lag time, respectively [7]. This model is developed from commands and actual responses of the aileron angle that is observed by using MuPAL-α.

[1]Nobuyuki Yoshikawa is with the Department of Aeronautics and Astronautics, The University of Tokyo, Bunkyo, Tokyo, 1138656 Japan email: nyoshikawa@g.ecc.u-tokyo.ac.jp

[2]Nacim Belkhir is with the TAU team, INRIA Saclay, Orsay,91400 France. email: nacim.belkhir@inria.fr

[3]Sinji Suzuki is with the Department of Aeronautics and Astronautics, The University of Tokyo, Bunkyo, Tokyo, 1138656 Japan email:tshinji@mail.ecc.u-tokyo.ac.jp

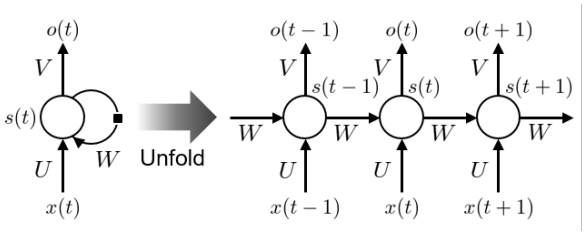


Fig. 1. Schema describing a RNN and the unfolding in time of the computation involved in the forward computation.

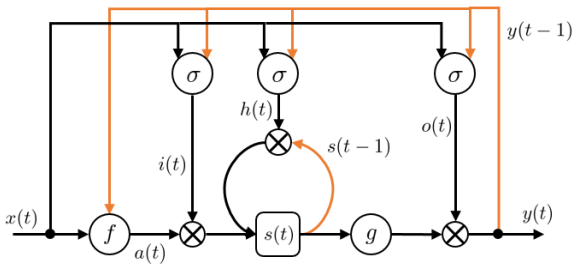


Fig. 2. Example of LSTM neurons in a hidden layer and description of the inner architecture of cells. Orange arrows show data from previous step.

III. FAULT DETECTOR IN CONTROL SYSTEMS

Fault detection (FD) is to monitor the system and detect the occurrence of a failure. One basic approach of FD is a method which uses hardware redundancy for comparing values measured by duplicated sensors. Model-based methods are based on the comparison between measured values and predicted values using a mathematical model or a knowledge-based model[8].

Fault detection and isolation (FDI) attempts to distinguish not only the presence or absence of a failure but also the location or type of failure. FDI can be divided into model-based FDI and signal processing based FDI. In recent years, the signal processing based FDI research has been actively conducted. Fault detection by neural network is a typical method among signal processing based methods [9], [10]. The NN is trained by using measured values with failure information to learn the relationship between the measured values and the failure. Using NN makes it possible to detect failures in systems that include strong nonlinearity and systems in which relationships between measured values and failures are not known. NN is suitable for FDI for ailerons, because ailerons are also nonlinear models which have dead time and variable time constants [7]. However, ordinary NN estimates outputs from a combination of input values. It is difficult to find failures with a temporal change, therefore recurrent neural network is adopted as the fault detector to detect failures including delay time changes.

IV. RECURRENT NEURAL NETWORKS (RNN)

Sinple feedforward neural network is widely invested in control community to design fault detector system(see [3], [4], [5] among many others). When traditional neural network is used, we assue that input data (and outputs) are independent each other. Traditional NN gives bad result when inputs are temporally dependent. Recurrent neural network addresses this issue by using the state values one step before as inputs to treat sequentially dependent data.. However, thanks to advances in their architectures [6], [11] and ways of training them [12], [13], RNNs have been found to be very good at learning a wide variety of sequential data.

A. Description

The concept of recurrent neural network can be traced back to [14] Fig.1 shows a typical recurrent neural network, such that an unfolded structure. In this study, 10 command/response data collected from the simulation model described in Section [2] are used to estimate the state of the aileron. Therefore we consider an unrolled structure, it would be 10 layer neural network: one layer per input data in the sequence. However typical RNN has the problem in learning, which is backpropagated gradients either grow or shrink at each time step, therefore over many time steps they typically explode or vanish [13], [15], [16].

B. Long Short Term Memory Neural Networks

Long Short Term Memory networks, known by LSTMs , are originally proposed in [6], and are a special kind of RNN. Initially, they has been proposed in order to tackle the long term dependency problem or RNN , with vanishing information through times. Hence, they are specifically designed for remembering information for long period of times.

Unlike RNNs, they have a more complex structure with four inner neural network layer that interact with each others in each neurons. A specific mechanism is necessary and relies on the principle of cell states and cell gates that can add or remove information to the cell state. Related to feedforward networks, while RNN neurons are defined by a neuron with recursive loop with respect to input sequence data; LSTM is similar to RNN but again contains sub-networks within its structures to retain information (see Fig.2 that shows the inner structure of the LSTM cells). Forward calculation of the LSTM is described as follows:

$$a(t) = f(W_c x(t) + U_c h(t-1) + b_c) \quad (2)$$
$$i(t) = \sigma(W_i x(t) + U_i h(t-1) + b_i) \quad (3)$$
$$h(t) = \sigma(W_h x(t) + U_h h(t-1) + b_h) \quad (4)$$
$$o(t) = \sigma(W_o x(t) + U_o h(t-1) + b_o) \quad (5)$$

$$s(t) = i(t) \odot a(t) + f(t) \odot s(t-1) \quad (6)$$
$$y(t) = o(t) \odot g(c(t)) \quad (7)$$

where $x(t)$ is the input vector, $s(t)$ is the cell state, $y(t)$ is the LSTM output, W_k, U_k amd b_k ($k \in \{c, i, h, o\}$) are weight parameters and biases, $f(\cdot) = tanh(\cdot)$ is a hyperbolic tangent, $g(\cdot) = ReUL(\cdot)$ is a rectified linear unit and σ is a sigmoid function. An operator $\odot$ indicates an Hadamard product.

V. EXPERIMENTAL SETTING

A. Parameters of the Aileron Model

The simulation model described by Eq.(1) was used to acquire learning data set. Rectangle waves and sinusoidal waves were inputted as command signals. Each waves period and amplitude were varied in following pairs: [4 sec, 2 deg.], [8 sec, 2 deg], and [4 sec, 5 deg]. The maximum command signal amplitude were assumed to be 5 degrees, thus commands and responses were normalized by 5 degrees when they were used as the inputs of neural network. We are considering using image measurement to acquire the actual aileron angle of MuPAL-α. The sampling rate of image processing is 10 Hz. Therefore the sampling rate of learning data set was 10 Hz.

In this study, 4 failure were assumed as following: delayed, gain changed, rate limited, fixed. These failure were implemented as the followings:

$$\dot{\delta_a}(t) = \frac{1}{\tau}(K\delta_{a_c}(t - T_f) - \delta_a(t)) \tag{8}$$

$$\dot{\delta_a}(t) = \frac{1}{\tau}(K_f\delta_{a_c}(t - T) - \delta_a(t)) \tag{9}$$

$$\dot{\delta_a}(t) = \min\{\dot{\delta_{af}}, \frac{1}{\tau}(K\delta_{a_c}(t - T) - \delta_a(t))\} \tag{10}$$

$$\dot{\delta_a}(t) = 0 \tag{11}$$

where T_f is the delay time on delayed, K_f is the gain on gain changed, $\dot{\delta_{af}}$ is the maximum change amount on rate limited. The aileron responses of normal state and typical failure state for the rectangle wave and sin wave are shown in Figs. 3, 4 respectively.

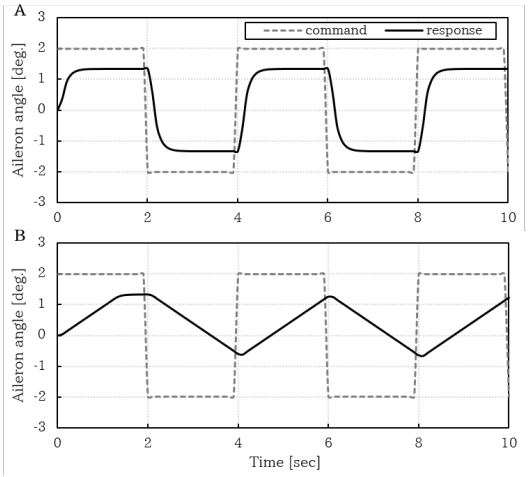


Fig. 3. Aileron simulation with rectangle wave (Period=4 sec, Amplitude=2 deg.) in (A) the normal state and (B) the failure of Rate Limited in 1 degree per second

From the introduced parameters, a limited set of examples are generated in order to have sequences of responses and command that are sliced in period of 1 second. In order to artificially increase the overall number of input data, a sliding window is used.

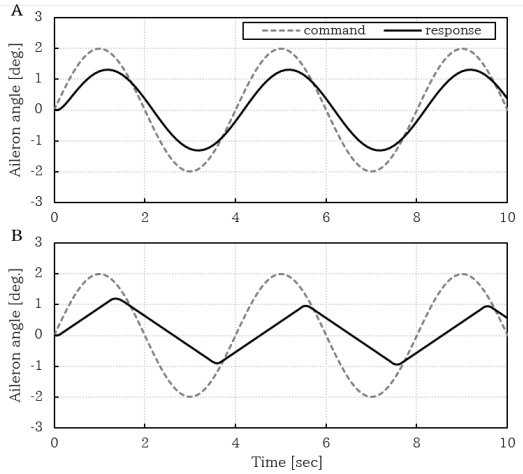


Fig. 4. Aileron simulation with sin wave (Period=4 sec, Amplitude=2 deg.) in (A) the normal state and (B) the failure of Rate Limited in 1 degree per second

B. Long Short Term Memory Network Configuration

Similarly traditional neural networks, the main challenge of LSTM is to find an appropriate configuration of the network and determine the best hyper-parameter setting of the network and the optimizer, so that we can learn at best a model that is robust to overfitting.

Because, there exist no rules or general strategy to configure neural networks, this is often left to the practitioners and trials/errors procedures. Hence, we propose to empirically investigate the configuration of the network and parameter setting that can have a huge impact on the learning capabilities.

In this study, input data is the command and the aileron response collected from simulation model as $x(t) = [\ \delta_a(t) \quad \delta_{ac}(t)\ ]^T$. The output vector is consisted with the estimated failure classes, normal or 4 failure states, $y(t) = [\ p_{normal}(t) \quad p_{delay}(t) \quad p_{gain}(t) \quad p_{limit}(t) \quad p_{fixed}(t)\ ]$.

The number of neuron N per layer is investigated, such that $N \in \{32, 64, 128, 256\}$. This is followed by the choice of the network architecture itself, such that the network L_N has one hidden layer, $L_{N,N}$ is a two hidden layer network with the same number of neuron per layer,then $L_{N,N/2}$ is a two hidden layer network with the second layer with half of the neurons. We limit our experimental study to small architecture, because the number of input variables and the overall number of example of example is rather small (e.g. compared to some image classification tasks,...), hence we aim at avoiding to overfit the fault detection model.

Regarding the optimization algorithm $\mathcal{A}$, several methods has been proposed in the literature[17], and we selected four optimizer with outstanding performances on a wide variety of problems: RMSprop[18], Adam[19], Adagrad[20], Nadam[21]. While neural network can be trained online, with 'streamed' input data, or from the whole dataset, batch processing has revealed to greatly improve the backpropagation of the gradient in the network: the batch process aims at resampling the dataset, and thus iteratively compute the

gradient hence avoiding local optima, that may worsen the performance of the neural network. Because we have few input data, we propose to investigate small batch sizes such that $b \in \{2, 5, 8, 12\}$

C. Empirical Validation

Here we consider two separate experimental cases: when the command signal has a rectangle wave shape, and when it has a sinusoidal wave shape. Then for each of these cases, we explore the different parameter setting and configuration of LSTM.

To proceed, for all experiments described in the previous section, we use a cross-validation procedure such that the input data are split into a training, a validation and a test set such that they input sequences are randomly shuffled, split without replacements, and the proportion of each type of failure is preserved in all sets. For statistical purposes, the cross-validation procedure is performed 25 times.

Regarding the performance measure, the cross entropy Eq.(12) of the training set, and the validation set are computed at each epochs of the learning process. Similarly the accuracy of the predicted classes for both the training and the validation set.

$$E = -\frac{1}{N_y} \sum_k y_k \ln y_k \tag{12}$$

The LSTM neural network and the overall experimental plan has been implemented in python. It relies on two main external libraries known for many successes in the machine learning community: keras (`https://keras.io/`) is used to implement neural networks and scikit-learn (`http://scikit-learn.org`) is used to compute performance measures.

VI. RESULTS

In this Section, we present the results of our experimental plan for learning a 'robust' fault detector, i.e. that is robust against overfitting. For all displayed figures, the mean and the standard deviation are displayed and computed over 25 trial independent runs. Due to space limitation of the paper, we only present typical results over all experiments: supplementary material are

Fig. 5 shows the loss values (cross-entropy) of the validation set, over the the learning process. It shows the mean and the standard error of the validation for different batch size, with a LSTM network with $L(N)$ and $L(N, N)$ configurations, such that the results are displayed for different size $N \in \{32, 64, 128, 256\}$.

First regarding the batch size, over the whole test data, we observe that using large batch size provide worst performances in particular when we use a small number of neurons, e.g. for N=32 and 64. Over the whole experiments, we observe that the using a batch of size 5 remains the best alternative in order to prevent from an increase of the loss value.

Regarding the number of neurons and different configurations, we observe that using smaller number of neurons, e.g. 32 give best results. As expected, using bigger number of neurons, lead to an increase of the loss. Similarly when the configuration is bigger, with two hidden layers, like $L(32, 32)$.

When comparing the different networks, we observe that the loss values greatly vary when we use 2 hidden layers: there is a huge variance and the loss start to decrease and then increases.

Fig.6 shows typical values of the loss function for the validation dataset when LSTM is configured with $L(32)$, and is trained with batches of examples of size 5. It compares the different optimizers based on stochastic gradient descent.

Without surprise, Adam is the clear winner when it is compared to the end of the learning phase (at the 50^{th} epoch). Although the loss values for Adagrad is higher, this remains a good alternative as it has a log linear behavior with a stable convergence rate.

Fig.7 shows the loss and accuracy of training and validation set w.r.t. the learning process. It shows that the accuracy is limited with an upper bound at 65%. Additionally Fig.8 shows an example with command and response data from the test set. We observe that the LSTM network is able to predict efficiently most cases, but remains not optimal as it only has a 65% accuracy.

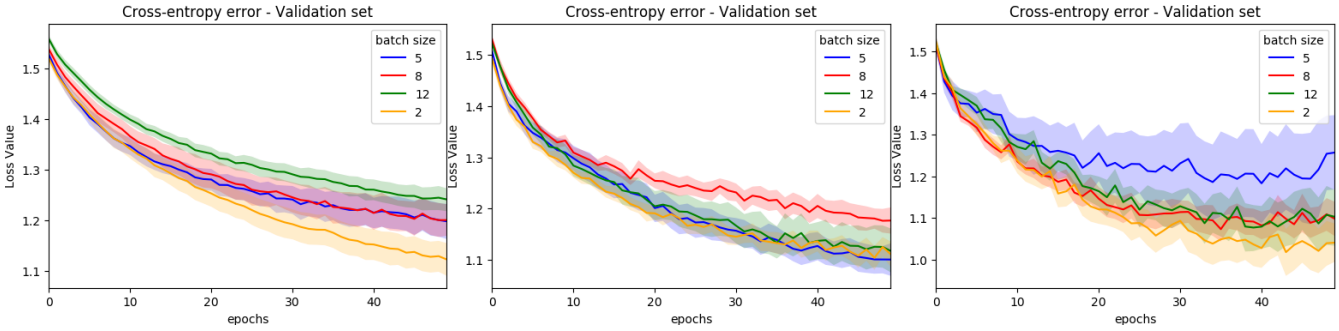
VII. DISCUSSION

Our results suggests that a fault detector can be learned by using a LSTM network. We empirically investigated the implementation of a LSTM network so that it can efficiently be used as a fault detector, hence beeing robust to over fitting. From the results we can say now that, that such task is challenging as in most case the it requires to use a very small configuration but still with open issues. Here, we generated some input from different model parameters, hence limiting the overall number of examples. However, neural networks are known to be methods that requires several input data (thousands, millions,...). Despite such major limitation the LSTM network reach a 60% percent of accuracy. While recent state of the art in the deep learning community, we observe that best optimizers, like adam and adagrad, fail when the configuration of the network is innapropriate. Hence it enforces the need to empirically investigate such approaches when tackling a new problem.

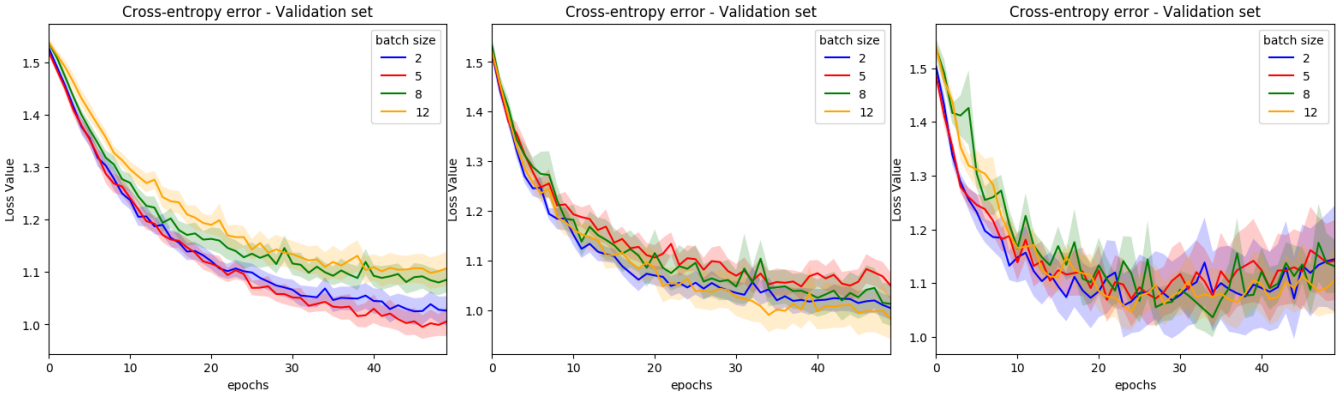
In this paper, we empirically demonstrated that a bad parameter configuration will lead to a increase of the loss, hence it suggests an overfitting on the training data that were used for learning the model. Despite the use of optimizers that are known to be robust to overfitting, this approach is still limited by the lack of data.

VIII. CONCLUSIONS AND FURTHER WORKS

In this paper, we empirically investigated the implementation of a fault detector for aircrafts, in particular for detecting failures related to ailerons. Given an aileron simulation model used for the MuPAL-α, in which failures can be controlled with respect to some parameters, we preliminary observed that a failure can be observed, and is dependent



(a) $L(32)$ (b) $L(64)$ (c) $L(128)$



(d) $L(32, 32)$ (e) $L(64, 64)$ (f) $L(128, 128)$

Fig. 5. Excerpt of performances (cross-entropy loss) for the validation set of data, and computed at each epochs of the learning process. Typical results are displayed for an increasing number of neurons N and increasing size of batch examples used during each epochs

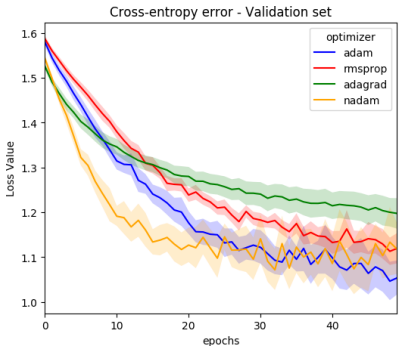


Fig. 6. Typical values of the loss during when the LSTM is with configuration $L(32)$ and a batch size = 5. It compares the the different optimizers.

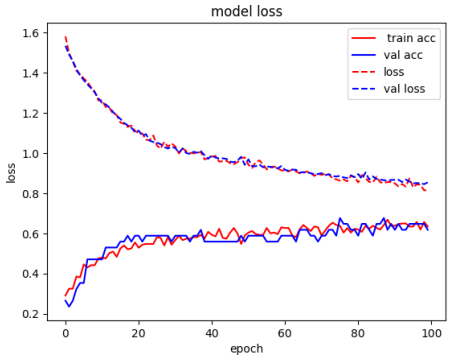


Fig. 7. Excerpt of the loss and accuracy computed from the training set and the validation loss and validation accuracy computed from the validation set.

to the previous command and response measures. In that direction, we proposed to use recurrent neural networks, in particular Long Short Term Memory (LSTM) networks, that are well known to deal with sequential data, where a temporal relationship exists.

We empirically investigated the network configuration of a LSTM network and the major parameter setting that has a key role in the overall performance and generalization of the network. Despite very small set of input data, we demonstrated that LSTM can be used as a fault detector and reach 60 to 65 percent of accuracy on new data. We empirically demonstrated that the parameters and configurations choice of neural networks and recurrent neural networks still remain an open issue, but that this empirical study is a first step toward a better understand of such mechanism.

Although, recurrent neural network are promising methods for tackling multivariate temporal data, as traditional neural network, they are cursed by the required number of examples to learn a model: few input data may lead to a bad learning of the model. Therefore, we suggests in the further work to investigate on new methods that deals with few example data in order to improve our approach, but more importantly it should be investigated with more data, that will help to

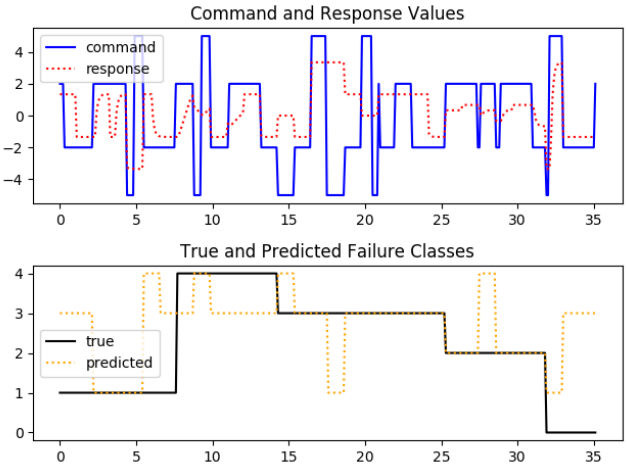


Fig. 8. Example of Command and Response values (on the top) and the plotted with respect to the true failure type and predicted failure type (on the bottom).

generalize and improve the LSTM model. In addition, a fault detector is first step toward an a Fault tolerant controller that new adaptive commands. Regarding such topic, reinforcement learning, as been widely proved in the literature (see [22], [23], [24], [25] for an overview of reinforcement for feedback control and adaptive controllers) to be a promising path to explore.

REFERENCES

[1] statista, "Annual growth in global air traffic passenger demand from 2005 to 2017," *https://www.statista.com/statistics/193533/growth-of-global-air-traffic-passenger-demand/*, 2017.

[2] A. Fekih, H. Xu, and F. N. Chowdhury, "Neural networks based system identification techniques for model based fault detection of nonlinear systems," *International Journal of Innovative Computing, Information and Control*, vol. 3, no. 5, pp. 1073–1085, 2007.

[3] K. J. Hunt, D. Sbarbaro, R. Żbikowski, and P. J. Gawthrop, "Neural networks for control systemsa survey," *Automatica*, vol. 28, no. 6, pp. 1083–1112, 1992.

[4] Y. Zhang and J. Jiang, "Bibliographical review on reconfigurable fault-tolerant control systems," *Annual reviews in control*, vol. 32, no. 2, pp. 229–252, 2008.

[5] R. T. Rysdyk and A. J. Calise, "Fault tolerant flight control via adaptive neural network augmentation," 1998.

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[7] M. Sato and A. Satoh, "Flight control experiment of multipurpose-aviation-laboratory in-flight simulator," *J. Guidance, Control, and Dynamics*, vol. 34, no. 6, pp. 1081–1096, 2011.

[8] J. Ghidella and P. J. Mosterman, "Requirements-based testing in aircraft control design," in *Proceedings of the AIAA Modeling and Simulation Technologies Conference and Exhibit 2005*, 2005, pp. 2005–5886.

[9] B. Samanta and K. Al-Balushi, "Artificial neural network based fault diagnostics of rolling element bearings using time-domain features," *Mechanical systems and signal processing*, vol. 17, no. 2, pp. 317–328, 2003.

[10] I. Eski, S. Erkaya, S. Savas, and S. Yildirim, "Fault detection on robot manipulators using artificial neural networks," *Robotics and Computer-Integrated Manufacturing*, vol. 27, no. 1, pp. 115–123, 2011.

[11] S. El Hihi and Y. Bengio, "Hierarchical recurrent neural networks for long-term dependencies," in *Advances in neural information processing systems*, 1996, pp. 493–499.

[12] I. Sutskever, "Training recurrent neural networks," *University of Toronto, Toronto, Ont., Canada*, 2013.

[13] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International Conference on Machine Learning*, 2013, pp. 1310–1318.

[14] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the national academy of sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.

[15] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[16] S. Hochreiter, "Untersuchungen zu dynamischen neuronalen netzen," *Diploma, Technische Universität München*, vol. 91, 1991.

[17] S. Ruder, "An overview of gradient descent optimization algorithms," *CoRR*, vol. abs/1609.04747, 2016. [Online]. Available: http://arxiv.org/abs/1609.04747

[18] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.

[19] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[20] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[21] T. Dozat, "Incorporating nesterov momentum into adam," 2016.

[22] R. S. Sutton, A. G. Barto, and R. J. Williams, "Reinforcement learning is direct adaptive optimal control," *IEEE Control Systems*, vol. 12, no. 2, pp. 19–22, 1992.

[23] F. L. Lewis and D. Vrabie, "Reinforcement learning and adaptive dynamic programming for feedback control," *IEEE circuits and systems magazine*, vol. 9, no. 3, 2009.

[24] F. L. Lewis, D. Vrabie, and K. G. Vamvoudakis, "Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers," *IEEE Control Systems*, vol. 32, no. 6, pp. 76–105, 2012.

[25] F. L. Lewis and D. Liu, *Reinforcement learning and approximate dynamic programming for feedback control*. John Wiley & Sons, 2013, vol. 17.