Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 55-6 (2022) 643–648

Fault Detection and Isolation for UAVs using Neural Ordinary Differential Equations

Luis Enciso-Salas * **Gustavo Pérez-Zuñiga** *
Javier Sotomayor-Moriano *

* *Engineering Department, Pontifical Catholic University of Peru, Av. Universitaria 1801, San Miguel, Lima 15088, Perú (e-mail: {lenciso, gustavo.perez, jsotom}@pucp.edu.pe).*

Abstract: In recent years, the increasing complexity and diversity of data-based fault detection and isolation (FDI) methods usually require high computational efforts in the pre-processing stage, large amounts of data, and, most of the time, some feature extraction to obtain relevant information for the data-based algorithms. This paper proposes using the Neural Ordinary Differential Equations (NODE) framework to represent the dynamics of the studied plant and later employ such representation in FDI system design. Such an approach enables loss optimization to be performed jointly in the plant dynamics and external inputs without previous use of complex pre-processing and is useful for working with nonlinear systems. The approach is first validated using a simulated Unmanned Aerial Vehicle (UAV) and later applied to a data-set that contains actuators and sensors faults. Ultimately, the proposed approach is compared with other usual machine learning techniques, showing better performance metrics.

Keywords: Fault diagnosis, Neural Ordinary Differential Equations, Machine learning, UAV

1. INTRODUCTION

Nowadays, the applications of UAVs are rapidly increasing as engineering improves and becomes more accessible. UAVs use includes working under limiting factors; and it is inevitable that its devices onboard will fail at some point in time (Huaman and Pérez Zuñiga, 2019). Usually, UAVs FDI system design approaches are divided into model-based, data-based, or a combination of them. The model-based approach in (Han et al., 2018) is a combined parity-space and least-square method, and in (Keipour et al., 2019) the method of least square is employed to find an online model and then be used in the diagnosis system. In the data-based approach, the data-set is constructed and then used to train a fault diagnosis model, albeit decision logic can be added in some cases such as in (Sun et al., 2017) where an ANFIS (Artificial Neural Fuzzy Inference System) is used, better performances have been obtained recently in the field of machine learning (Lei et al., 2020), such as in (Abbaspour et al., 2017) where a hybrid solution based in neural networks with backpropagation and extended Kalman Filter is presented to work in an online setup. However, improved performances with the use of deep learning potential are yet to be well developed for UAVs FDI systems.

Within deep learning techniques, NODE (Chen et al., 2018) solves an ODE initial value problem for a function represented in the form of a neural network using ODE solvers, thus representing a continuous transition in the evolution of the network, and is helpful for the analysis of faults by learning a continuous representation of their dynamics. Other solutions inspired in NODE have been developed, making its implementation a well-established framework, including an augmentation to be able to represent a wider variety of dynamics (Dupont et al., 2019). Works for irregular time series can be found in (Rubanova et al., 2019) and more recently in (Kidger et al., 2020), these works show the potential of NODE for predictive tasks; other works such as in (Deng et al., 2019) use NODE for continuous graph flows in generative tasks, in (Tzen and Raginsky, 2019) for an extension of stochastic processes, in (Liu et al., 2019) for inclusion of noise injection mechanisms for a framework in stochastic differential equations; and also in (Poli et al., 2020) for inclusion in Graph Neural Networks as a tool to have a continuous flow of node features, all these works show the richness of the representations that can be achieved with this approach. In (Enciso-Salas et al., 2021) NODE is used in combination with model-based fault diagnosis methods rendering a good performance in FDI; however, the resultant system required linearity and previous modeling in the input relations.

In this work, the FDI system is designed and trained to be wholly data-based and works on external inputs with non-linear relations; therefore, instead of the work in (Enciso-Salas et al., 2021) no further modeling of the studied plant is required. Besides, the pre-processing employed is kept minimal, only requiring the normalization of the data; furthermore, if no critical information is lost in the process, sub-sampling for faster training of the NODE is also possible. Another potential advantage of the approach is its possible use in predictive tasks since the system can be utilized to predict trajectories in state space using inputs and previous distributions.

10.1016/j.ifacol.2022.07.200

This article is structured in the following manner: Section 2 deals with the NODE presentation and handling of external inputs, Section 3 explain the modeling and architecture of the employed NODE employed, Section 4 is dedicated to the training procedure for the NODE model, Section 5 is dedicated to the fault diagnosis training and evaluation, and Section 6 give the conclusions.

2. NEURAL ORDINARY DIFFERENTIAL EQUATIONS AND EXTERNAL INPUTS

NODE (Chen et al., 2018) represents continuous transitions of the network; therefore can be expressed as a dynamical transition and define the solution to an initial value problem:

$$\begin{cases} \dot{\mathbf{H}}(t) = \mathbf{F}(t, \mathbf{H}(t), \theta) \\ \mathbf{H}(0) = \mathbf{X}_0 \end{cases}, \tag{1}$$

where $\mathbf{F}$ represents the neural network, $\mathbf{H}$ represents the hidden states, θ the parameters of the network and $\mathbf{X}_0$ the initial hidden state.

2.1 NODE training using adjoint state

The solution to (1) is performed through training using ODE solvers. The loss L is calculated using the forward propagation and a first ODE solver as (2).

$$\begin{aligned} L(z(t_1)) &= L\left(z(t_0) + \int_{t_0}^{t_1} \mathbf{F}(z(t), t, \theta)\, dt\right) \\ &= L(ODESolve(z(t_0), \mathbf{F}, t_0, t_1, \theta))\,, \end{aligned} \tag{2}$$

where $z(t)$ is the hidden state at each instant, t_0 and t_1 are the initial and final time respectively.

In the backward propagation, the adjoint state, $a(t) = \partial L/\partial z(t)$, is used for the calculation of the propagation gradients and relation (3) can be prove (Chen et al., 2018),

$$\frac{\partial a(t)}{\partial t} = -a(t)^\top \frac{\partial \mathbf{F}(z(t), t, \theta)}{\partial t}. \tag{3}$$

Also, computation of the gradient change with respect to the parameters requires to solve relation (4) backward in time,

$$\frac{\partial L}{\partial \theta} = -\int_{t_1}^{t_0} a(t)^\top \frac{\partial \mathbf{F}(z(t), t, \theta)}{\partial t}. \tag{4}$$

Equations (2)–(4) are implemented as augmented dynamics that permit the evaluation of the respective integrals for z, a, and $\frac{\partial L}{\partial \mathbf{\Theta}}$ jointly in a single call to the ODE solver.

2.2 Considering External Inputs

In order to handle nonlinear relations in the input, which is an improvement from previous work (Enciso-Salas et al., 2021), the following nonlinear state-space representation of a process is considered:

$$\dot{x}(t) = f(x) + g(u)\,, \tag{5}$$

$$y(t) = h(x)\,, \tag{6}$$

where $x(t) \in \mathbb{R}^n$ is the state vector, $f(.)$ is a function $\mathbb{R}^n \to \mathbb{R}^n$, $u(t) \in \mathbb{R}^m$ the input vector, $g(.)$ is a function $\mathbb{R}^m \to \mathbb{R}^n$, $y(t) \in \mathbb{R}^p$ the output vector, and $h(.)$ is an invertible function $\mathbb{R}^n \to \mathbb{R}^p$.

Then using a network U to represent the external inputs relationship, as (7)

$$U = g(\bar{u}, t, \beta), \tag{7}$$

where $\bar{u}$ represent the known set of inputs and β are the parameters of this network.

Therefore using equation (2) and equation (7), the loss relation is given by (8),

$$\begin{aligned} &L(z(t_1)) = \\ &= L\left(z(t_0) + \int_{t_0}^{t_1} f(z(t), t, \theta)\, dt + \int_{t_0}^{t_1} g(\bar{u}, t, \beta)\, \mathrm{dt}\right) \\ &= L(ODESolve(z(t_0), \mathbf{f}, \mathbf{g}, t_0, t_1, \theta, \beta))\,, \end{aligned} \tag{8}$$

Moreover, it is required to calculate derivatives for parameters θ, β, and t. For that, the augmented dynamics are denoted by:

$$\frac{d}{dt}\begin{bmatrix} z \\ \theta \\ \beta \\ t \end{bmatrix} = f_{aug}([z, \theta, \beta, t]) = \begin{bmatrix} f(z(t), t, \theta) + g(\bar{u}, t, \beta) \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{9}$$

And it is furthermore possible to use the augmented adjoint,

$$a_{aug} = \begin{bmatrix} a \\ a_\theta \\ a_\beta \\ a_t \end{bmatrix}, \quad a_\theta(t) = \frac{\partial L}{\partial \theta}, \quad a_\theta(t) = \frac{\partial L}{\partial \beta}, \quad a_\theta(t) = \frac{\partial L}{\partial t}, \tag{10}$$

such that, due to the independence of the terms of $\bar{u}$, the following Jacobian can be obtained,

$$\frac{\partial f_{aug}}{[z, \theta, \beta, t]} = \begin{bmatrix} \frac{\partial f}{\partial z} & \frac{\partial f}{\partial \theta} & \frac{\partial g}{\partial \beta} & \frac{\partial f + \partial g}{\partial t} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{11}$$

and by the same principle in (3), the relation for the augmented adjoint could be use for (12)

$$\frac{da_{aug}}{dt} = -\left[a(t)\ a_\theta(t)\ a_\beta(t)\ a_t(t)\right] \frac{\partial f_{aug}}{\partial([z, \theta, \beta, t])}(t) \tag{12}$$

It can be shown that relation (3) can also be derived from (12), and on the other hand, for the gradients with respect to β the relation (13) is to be computed; meanwhile, relation for adjoint a_t can be used for computation of the changes with respect to the initial and end of the

integration interval. Furthermore, it can be appreciated that parameters θ and β can be handled together during the training.

$$\frac{\partial L}{\partial \beta} = -\int_{t_1}^{t_0} a(t)^\top \frac{\partial \mathbf{g}(\bar{u}, t, \beta)}{\partial t}. \tag{13}$$

3. MODEL REPRESENTATION

As the NODE could be used to represent dynamics, we relate the trajectories generated by its outputs with a state-space representation as given in (5) and (6); here, for a UAV device, a more general setup is considered with the effect of previous states in such dynamics. These considerations justify the architecture of the NODE that is used in the rest of the paper.

3.1 UAV modeling

The data-set employed for this demonstration has been obtained using a UAV model (see Figure 1), for which the system can be modeled by the following simplified dynamics relations(Zhang et al., 2014):

$$\ddot{x} = \frac{s\phi s\psi + c\phi c\psi s\theta}{m} F \tag{14}$$

$$\ddot{y} = \frac{c\phi s\psi s\theta - c\psi s\theta}{m} F \tag{15}$$

$$\ddot{z} = -g + \frac{c\phi c\theta}{m} F \tag{16}$$

$$\ddot{\phi} = \dot{\theta}\dot{\psi}\frac{I_y - I_z}{I_x} + \frac{1}{I_x}\tau_\phi \tag{17}$$

$$\ddot{\theta} = \dot{\phi}\dot{\psi}\frac{I_z - I_x}{I_y} + \frac{1}{I_y}\tau_\theta \tag{18}$$

$$\ddot{\psi} = \dot{\phi}\dot{\theta}\frac{I_x - I_y}{I_z} + \frac{1}{I_z}\tau_\psi, \tag{19}$$

where the states in the system are the positions x, y and z, the angular positions row (ϕ), pitch (θ) and yaw (ψ) and their respective derivatives, inputs are the total thrust F, and torques in each angle axis τ_ϕ, τ_θ and τ_ψ for roll, pitch, and yaw respectively. Moreover, the system is tested with the following parameters, $M = 10Kg$, $I_x = 0.4Kg - m^2$, $I_y = 0.3Kg - m^2$, $I_z = 0.3Kg - m^2$, $L = 0.5m$, and $g = 9.81m/s^2$.

Inputs equations are also defined with respect to velocities of the engines(Bresciani, 2008), Ω_1, Ω_2, Ω_3, and Ω_4 respectively, in the following manner:

$$F = b\left(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2\right), \tag{20}$$

$$\tau_\phi = Lb\left(-\Omega_2^2 + \Omega_4^2\right), \tag{21}$$

$$\tau_\theta = Lb\left(-\Omega_1^2 + \Omega_3^2\right), \tag{22}$$

$$\tau_\psi = d\left(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2\right). \tag{23}$$

A dynamic feedback linearization controller controls the UAV position, and a PD controller regulates the yaw angle. The system has a total of four actuators or engines. Also, measurements for each one of the states: x, y, z, ϕ, θ, and ψ, are considered, minimally this means GPS sensor for positions, and accelerometers plus gyro-meters for measuring angle displacements are installed in the device.

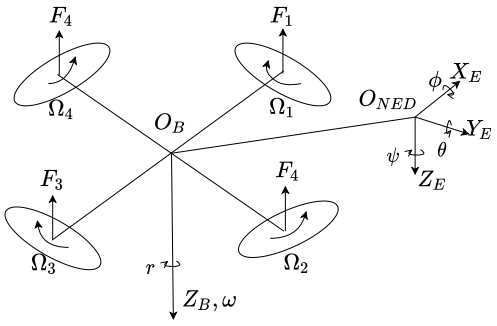

Fig. 1. UAV model scheme.

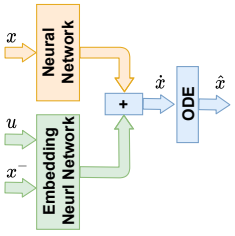

Fig. 2. NODE model used to represent the UAV dynamics.

The following faults are considered:

- Sudden shutdowns of engines.
- Sensor bias varying in the range between $\pi/36$ rad and $\pi/10$ rad.

3.2 NODE structure

For the UAV representation, the NODE has to handle nonlinear relationships in the inputs, then given a state-space system such as

$$\dot{x}(t) = f(x) + g(x, u)\,, \tag{24}$$

$$y(t) = h(x)\,. \tag{25}$$

The architecture given in Figure 2 is proposed to represent the UAV dynamics, where the embedding of the input comprises the proper inputs and past measurements of the system x^-, then to aim the following plant dynamics,

$$\dot{\hat{x}}(t) = \hat{f}(\hat{x}) + g(u, x^-)\,, \tag{26}$$

Notice that x^- can be treated as constant for the NODE system since they are known estimated values that can be obtained from $h^{-1}(y)$, thus not affecting the gradients of the training mechanism and respecting the equations in section 2.2.

4. TRAINING

4.1 NODE training

The data-set is composed by noisy samples of inputs and outputs generated from the model in section 3.1, and then

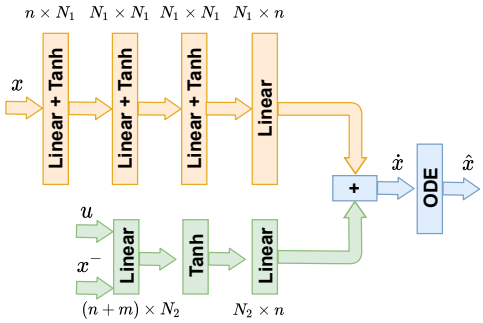

Fig. 3. NODE used for the model training, N_1 and N_2 are adjusted accordingly.

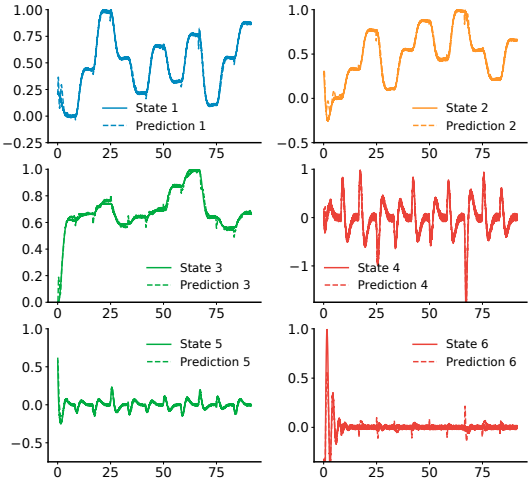

Fig. 4. Training performance of the NODE.

the NODE model (Figure 3) is trained with the following setup:

- A total of 4000 samples of the non-faulty UAV are used to train and validate the NODE system; such data is normalized to the range between 0 and 1 and sub-sampled to reduce its size and facilitate the training; no other pre-processing procedure is required.
- The data is then divided into 80% for training and 20% for validation.
- A total of 1000 periods of training (epochs) are performed in training to minimize $\hat{x} - h^{-1}(y)$.
- Training of the input embedding components, the fully connected layers A and B, are performed in independent steps every 100 epochs.
- Batches of 500 samples are used in each epoch.
- Learning rate starts at 10^{-3} and is updated at epochs 500 to 10^{-4} for the rest of the training.
- Since the data-set is regularly sampled and increases the speed of training; the NODE is trained with an Euler method.

The obtained model gives the results in Figure 4 for the training set and the results in Figure 5 for the validation set. It can be appreciated that renders the right behavior in the generation of the predicted states. Besides, the total MSE loss is 0.002 and 0.0009 in each case, respectively.

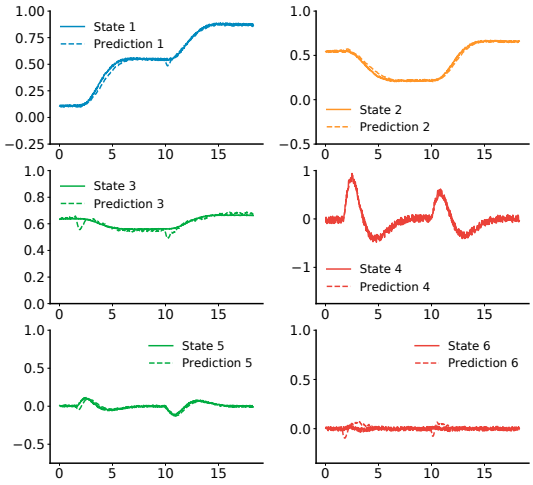

Fig. 5. Validation performance of the NODE.

5. TRAINING AND EVALUATION OF THE FDI SYSTEM

5.1 Data-set and pre-processing

For the UAV data-set we use the model in section 3.1 to generate the noisy samples of trajectories in the vehicle and sudden engine faults; we model the faults as shut-downs of one of the engines during flight and save its duration and timestamp for later recognition; furthermore sensor and process faults are added for the training of the FDI system. In our generated data-set we also include ground-truth for the fault conditions status and the command for the engines. Also, in each case the data is normalized using the same normalization constants used for the NODE training, and no other pre-processing procedure is performed.

5.2 Fault Detection training

For the fault detection, two fully connected layers are added to the NODE system, as presented in Figure 6, then the following setup is used for the training:

- The data-set with faults is divided into 70% for training and 30% for validation.
- A total of 1000 epochs are performed for the training with individual batches of 400 samples uniformly selected from the training data-set.
- Since the data-set is imbalanced, weights are used for the training, having at least two classes: no-fault and fault in the engine; the vector of weights used was $[0.01, 100.0]$.
- The learning rate starts at 10^{-3} and is updated at epochs 800 to 10^{-4} for the rest of the training.

Optimization is performed in the following manner:

- Parameters of the NODE are kept constant since the NODE was previously trained.
- The NODE is used to generate an estimate of the states $\hat{x}$ based on the current inputs u and past estimated states $h^{-1}(y^-)$.
- Then the difference of the predicted derivatives are obtained such as $F_{NODE}(\hat{x}) - F_{NODE}(h^{-1}(y))$, where F is the NODE function.

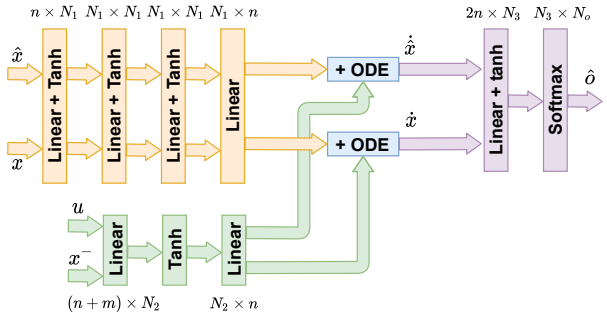

Fig. 6. FDI system, the NODE is kept constant. N_3 is adjusted accordingly.

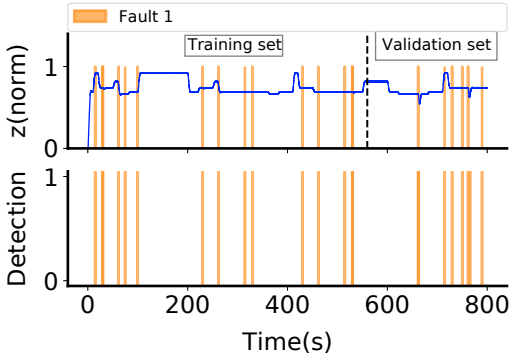

Fig. 7. Fault detection, continuous line in blue represents normalized position z, faults occurring are shown in the figure above, and detections are shown below.

- This difference is passed into the augmented neural network to classify the faulty condition.

Also summarized in the Algorithm 1.

Algorithm 1 Training FDI system

Require: u, y^-, t, *labels* ▷ Initial data
  $N \leftarrow n\_batch$ ▷ number of sub-batches
  $S \leftarrow size\_batch$ ▷ size of sub-batches
  $optimizer = Adam(lr = 0.001)$
  for $i \in epochs$ **do**
    for $batch(N, S) \in training\_DS$ **do**
      $\hat{x} \leftarrow NODE(u, h^{-1}(y^-))$
      $d_{\hat{y}} \leftarrow NODE(\hat{x})$
      $d_y \leftarrow NODE(h^{-1}(y))$
      $pred \leftarrow AUG\_NN((d, d_y))$ ▷ Prediction
      $loss \leftarrow CrossEntropyLoss(pred, label)$
      $loss.backward()$
    end for
  end for

The FDI system results are presented in Figure 7 for training and validation sets, later these results are used for section 5.4.

5.3 FDI training

Similarly, we can evaluate the system to isolate different engine faults and sensor faults. Thus, a new data-set including the effects of different fault scenarios in actuators and sensors is used. The tested scenarios shown in the results are: no-fault, fault in engine 1 (Fault 1), fault in engine 2 (Fault 2), and fault in roll sensor (Fault 3). Then, performing training using the Algorithm 1 give us results shown in Figure 8 which are obtained for training and validation data-sets. Also, for untested fault scenarios, visualization of the continuity of the states as given in (Enciso-Salas et al., 2021) is a helpful tool for fault detection.

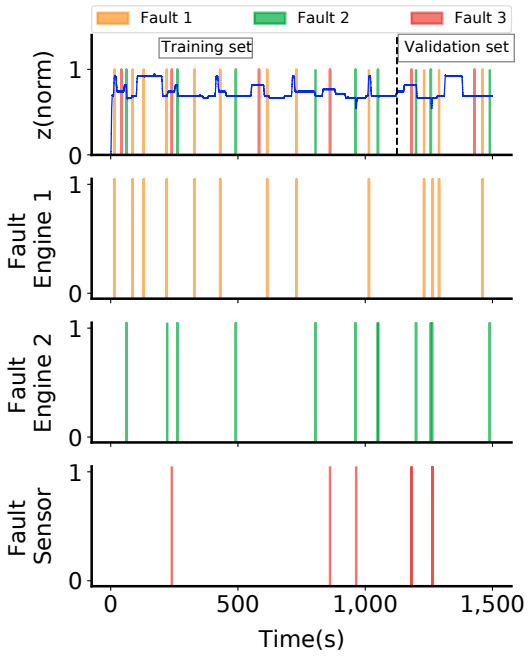

Fig. 8. Complete FDI system. In the figure above, different bar colors represent various sudden faults occurring; and the detections are shown below correspondingly.

5.4 Evaluation

For the evaluation phase, we tested the Fault diagnosis for binary classification (Section 5.1) using different algorithms:

- **Support Vector Machine (SVM)**: The SVM is one of the main tools in machine learning for time-series(Lara-Benítez et al., 2021). Here, it was used for the classification of a fault or non-fault status; for such reason, a total of 10 features (6 measurements and 4 inputs) were employed. The SVM kernel was a polynomial with degree 4, and the class weights were 1000 and 1 for fault and non-fault status, respectively.
- **Long-Short Term Memory (LSTM)**: The LSTM is a recurrent network with a lot of success in recent applications for time-series (Lara-Benítez et al., 2021). In this application, it was used in combination with a softmax layer to make classification of a fault or non-fault status. LSTM required special tuning of its parameters, and the best results were obtained using 30 hidden states, 10 LSTM layers, dropping ratio of 0.5, and class weights of 100 and 0.02 for fault and non-fault status, respectively. Also, in this case, a total of 10 features (6 measurements and 4 inputs) were used.

Evaluation is performed in each model with two main metrics, given the imbalanced distribution of the faults

Table 1. Metrics for different Machine learning models.

S.No.	Training set		Validation set	
	F1-score	AUC	F1-score	AUC
SVM	0.98	0.99	0.024	0.19
LSTM	0.14	0.93	0.21	0.93
FDI-NODE	0.56	1.0	0.45	0.93

with respect to the whole data-set. Therefore, the F1-score and the AUC (Area under the curve) are used, F1-score obtains a weighted relation using precision and recall, thus considering the false and true negatives, and AUC values the capacity of the network in distinguishing both classes, a higher value indicates a better metric in both cases.

Results for each case are presented in Table 1. As shown, the FDI-NODE obtained a better overall performance over the methods considered in this particular data-set. SVM is clearly over-fitting the training set; meanwhile, LSTM, although it does a better job, has a low F1-score. Moreover, results in Figure 7 show that FDI-NODE does a proper detection in every case, albeit having some false positive samples in the detected periods, which decreases its metrics in the presented table.

6. CONCLUSIONS

The proposed approach provides more adaptability to work with nonlinear relations in the description of the plant dynamics using NODE models; the adaptation is achieved by the optimization of the training phase and improves the capability for handling nonlinear plants with nonlinear relationships in its inputs with the form $g(x, u)$. Thus, training the NODE model does not require elaborated schemes for pre-processing the data that are commonly used in other approaches, such as feature extraction and data augmentation and therefore less computational effort. Furthermore, using the NODE framework is an appealing manner to obtain a continuous model of the system that can be adapted for further predictive tasks. To validate this approach, a fault detection system was developed and compared with other usual machine learning approaches applied to time-series data, always showing improved performance. And later, an FDI system for the complete set of actuators and sensors in the UAV was designed, offering great performance.

ACKNOWLEDGEMENTS

This research was funded by *Proyecto de Mejoramiento y Ampliación de los Servicios del Sistema Nacional de Ciencia Tecnología e Innovación Tecnológica 8682-PE, Banco Mundial, CONCYTEC and PROCIENCIA* through grant E041-01[N48-2018-FONDECYT-BM-IADT-MU].

The authors are also grateful for the finantial support from Contrato de Adjudicación de fondos N10-2018-FONDECYT/BM-Programas de Doctorados en Áreas Estratégicas y Generales.

REFERENCES

Abbaspour, A., Aboutalebi, P., Yen, K.K., and Sargolzaei, A. (2017). Neural adaptive observer-based sensor and actuator fault detection in nonlinear systems: Application in UAV. *ISA Transactions*, 67, 317–329.

Bresciani, T. (2008). *Modelling, identification and control of a quadrotor UAV*. Ph.D. thesis, Lund University.

Chen, R.T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. (2018). Neural Ordinary Differential Equations. In *NeurIPS 2018*, volume 109, 31–60.

Deng, Z., Nawhal, M., Meng, L., and Mori, G. (2019). Continuous Graph Flow. *arXiv preprint arXiv:1908.02436*.

Dupont, E., Doucet, A., and Teh, Y.W. (2019). Augmented neural odes. *arXiv preprint arXiv:1904.01681*.

Enciso-Salas, L., Pérez-Zuñiga, G., and Sotomayor-Moriano, J. (2021). applied sciences Fault Diagnosis via Neural Ordinary Differential Equations. *Applied Sciences (Switzerland)*, 11(3776).

Han, W., Wang, Z., and Shen, Y. (2018). Fault estimation for a quadrotor unmanned aerial vehicle by integrating the parity space approach with recursive least squares. *Proc.Brescian of the IME, Part G: Journal of Aerospace Engineering*, 232(4), 783–796.

Huaman, A.S. and Pérez Zuñiga, C.G. (2019). Design of a fuzzy sliding mode controller for the autonomous path-following of a quadrotor. *IEEE Latin America Transactions*, 17(6), 962–971.

Keipour, A., Mousaei, M., and Scherer, S. (2019). Automatic Real-time Anomaly Detection for Autonomous Aerial Vehicles. *arXiv*, 5679–5685.

Kidger, P., Morrill, J., Foster, J., and Lyons, T. (2020). Neural controlled differential equations for irregular time series. *Advances in Neural Information Processing Systems*.

Lara-Benítez, P., Carranza-García, M., and Riquelme, J.C. (2021). An Experimental Review on Deep Learning Architectures for Time Series Forecasting. *International Journal of Neural Systems*, 31(3), 1–25.

Lei, Y., Yang, B., Jiang, X., Jia, F., Li, N., and Nandi, A.K. (2020). Applications of machine learning to machine fault diagnosis: A review and roadmap. *Mechanical Systems and Signal Processing*, 138, 106587.

Liu, X., Xiao, T., Si, S., Cao, Q., Kumar, S., and Hsieh, C.J. (2019). Neural SDE: Stabilizing neural ODE networks with stochastic noise. *arXiv preprint arXiv:1906.02355*.

Poli, M., Massaroli, S., Park, J., Yamashita, A., Asama, H., and Park, J. (2020). Graph neural ordinary differential equations. *Advances in Neural Information Processing Systems*, 2018-Decem, 6571–6583.

Rubanova, Y., Chen, R.T., and Duvenaud, D. (2019). Latent ODEs for irregularly-sampled time series. *Advances in Neural Information Processing Systems*, 32.

Sun, R., Cheng, Q., Wang, G., and Ochieng, W.Y. (2017). A novel online data-driven algorithm for detecting UAV navigation sensor faults. *Sensors (Switzerland)*, 17(10).

Tzen, B. and Raginsky, M. (2019). Neural Stochastic Differential Equations : Deep Latent Gaussian Models in the Diffusion Limit. *arXiv preprint arXiv:1905.09883*.

Zhang, X., Li, X., Wang, K., and Lu, Y. (2014). A survey of modelling and identification of quadrotor robot. *Abstract and Applied Analysis*, Vol.2014.