Safe Design of Stable Neural Networks for Fault Detection in Small UAVs

Kavya Gupta, Fateh Kaakai, Béatrice Pesquet-Popescu, Jean-Christophe Pesquet

Safe Design of Stable Neural Networks for Fault Detection in Small UAVs

Kavya Gupta[1,2⋆], Fateh Kaakai[2], Béatrice Pesquet-Popescu[2], and Jean-Christophe Pesquet[1]

[1] Université Paris-Saclay, CentraleSupélec, Inria
Centre de Vision Numérique, Gif-sur-Yvette, France
[2] Air Mobility Solutions BL, Thales LAS France

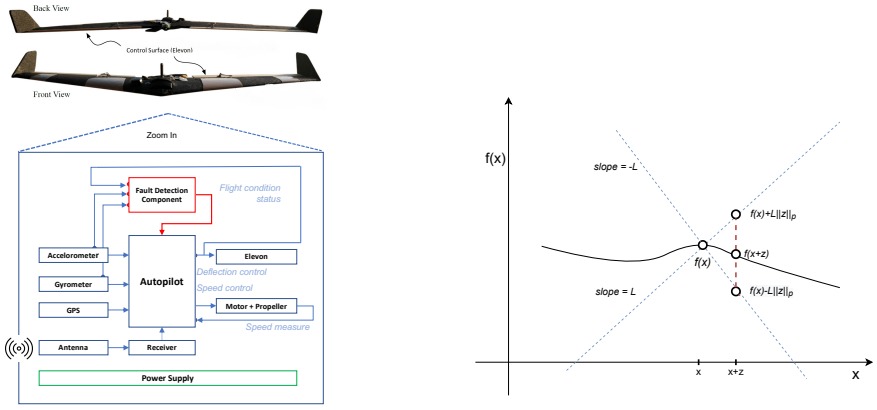
Abstract. Stability of a machine learning model is the extent to which a model can continue to operate correctly despite small perturbations in its inputs. A formal method to measure stability is the Lipschitz constant of the model which allows to evaluate how small perturbations in the inputs impact the output variations. Variations in the outputs may lead to high errors for regression tasks or unintended changes in the classes for classification tasks. Verification of the stability of ML models is crucial in many industrial domains such as aeronautics, space, automotive etc. It has been recognized that data-driven models are intrinsically extremely sensitive to small perturbation of the inputs. Therefore, the need to design methods for verifying the stability of ML models is of importance for manufacturers developing safety critical products.
In this work, we focus on Small Unmanned Aerial Vehicles (UAVs) which are in the frontage of new technology solutions for intelligent systems. However, real-time fault detection/diagnosis in such UAVs remains a challenge from data collection to prediction tasks. This work presents application of neural networks to detect in real-time elevon positioning faults. We show the efficiency of a formal method based on the Lipschitz constant for quantifying the stability of neural network models. We also present how this method can be coupled with spectral normalization constraints at the design phase to control the internal parameters of the model and make it more stable while keeping a high level of performance (accuracy-stability trade-off).

Keywords: safety · stability · Lipschitz constant· verification · machine learning · neural networks · UAV · tabular data · adversarial attacks.

1 Introduction

Machine Learning (ML) methods and in particular neural networks are rapidly paving their way in various mission-critical and safety-critical domains such as aeronautics, aerospace, automotive, railways, and nuclear plants. In particular, UAVs are currently being used in various applications such as surveillance of critical infrastructures, delivery of goods in densely populated urban areas, disaster

⋆ Corresponding author : kavya.gupta100@gmail.com



(a) (b)

Fig. 1: (a) Fault Detection in drones/UAVs :The component called "Fault Detection Component" computes a flight condition status (nominal or faulty) using the outputs of the inertial sensors and the elevon deflection control variable produced by the autopilot. (b) Intuition of using Lipschitz constant as a stability property of neural network [24].

management, emergency services, etc. UAVs are capable of replacing humans in potentially dangerous situations, but also reduce the cost of operations in some businesses. But these kind of systems need to be continuously monitored to detect in real-time any problem ("fault") that could lead to a failure of the UAV and with a potential interruption of the mission, a potential collision with a manned aircraft (air risk), a potential collision with people on ground or with a critical infrastructure like a nuclear plant (ground risk). The need is to develop fault-tolerant systems. Formally, fault tolerance is the ability of a system to perform the intended function despite the presence or occurrence of faults, whether it is a physical damage to the hardware, software defects, or malicious attacks. In this paper, we deal with a Machine Learning based UAV fault detection (depicted in the red component of Figure 1a) that implements a Deep Neural Network (DNN) to detect mechanical faults like an elevon stucked or locked in a given position. This real-time fault detection mechanism allows the remote pilot to take necessary actions (e.g. remote triggering of the drone emergency parachute) to avoid the loss of control of the UAV which could induce an air risk or a ground risk.

This Machine Learning based fault detection function should be safe to claim compliance with the applicable regulations. In particular, it should be stable with respect to noise perturbations in the input data coming from the inertial sensors and the autopilot i.e. the model continues to operate correctly despite small perturbations in its inputs within the Operational Design Domain (ODD). In other words, small perturbations that are considered as part of the real operating conditions should be considered in the designing phase of the model making the model 'stable-by-design'. However, the concept of adversarial attacks intro-

duced in [21] shows that adding a designed imperceptible noise to the sample is able to fool a neural network even if the model is trained to achieve high accuracy and performance. There have been studies that design different attacks [16, 10] on different model architectures and applications. Such design weakness in DNNs questions the desired stability property of neural networks and hence its deployment in mission-critical or safety-critical products like UAVs. Many works providing counter measures to adversarial attacks such as adversarial training [7] and defensive distillation have appeared in literature. But it is hard to demonstrate the completeness of these counter measures, i.e. their ability to address all types of foreseeable perturbations in the given operational context.

An example of formal measure of stability of neural network is the Lipschitz constant of the model which allows to evaluate how small perturbations in the inputs impact the output variations as illustrated in Figure 1b. A sensitivity analysis of inputs using Lipschitz constant for neural networks in presented in [9]. The contributions of this work is to demonstrate the efficiency of formal guarantees based on the Lipschitz constant for checking the stability of DNNs. We present a stability control loop grounded on spectral normalisation constraints to achieve an accuracy-stability trade-off. We show the results in various experimental designs on a small UAV dataset for classification tasks. We start with presenting generalisation capabilities of neural networks over other machine learning methods such as SVM and logistic regression, and then give Lipschitz constant estimates of different neural network configurations. We show that it is possible to train a neural network satisfying both Lipschitz and accuracy targets using our stability control loop. We also demonstrate the effectiveness of spectral normalised models in handling adversarial tasks.

The next section discusses related work in the field of fault detection using neural networks and stability analysis of these architectures. Following this, a spectral normalisation based stability control loop is described. Then the real flight data used to evaluate the fault detection model is presented. In the following section, results and observations for various scenarios related to binary and multi-class classification tasks are presented. The last section summarizes the main contributions of this work.

2 Related Works

2.1 Fault Detection and Diagnosis(FDD)

FDD strategies can be primarily divided into two categories model-based and data-driven. Model-based approaches require an accurate model of the aircraft for successful diagnosis of the faults. But a small UAV system is susceptible to various perturbations and lacks an accurate model. Hence, model based approaches fail in such scenarios and studies are then concentrated on using data-driven methods.

AI based data-driven techniques are gaining more and more interest in the field of fault detection. Fault detection in UAVs has been studied by using different sensors and/or for detecting faults in different fault prone parts of the

vehicle. Freeman et al.[6] compares model-based and data-driven fault detection methods by using data from small, low-cost UAVs. Both approaches were shown to be capable of detecting different elevon faults in real data during maneuvers and in the presence of environmental variations. Guo et al.[8] proposed a hybrid feature model and deep learning-based fault diagnosis for UAV sensors. The model detects different sensor faults, such as GPS, IMU and ADS. The authors of [25] modelled fault detection using Generalized Regression Neural Network with a particle swarm optimization algorithm. Eroglu et al.[4] proposed a data-driven fault estimation method for estimating actuator faults from aircraft state trajectories. The authors propose to use an architecture with 1-D CNN followed by LSTM layers for learning state trajectories on simulated data of fault injected aircraft. Sadhu et al. [19] proposed a deep learning based real-time fault detection and identification on UAV IMU sensor data using CNN and Bi-LSTM. Iannace et al. [12] presented a single layered neural network for detecting unbalanced blades in a UAV propeller. Bronz et al.[1] uses an SVM classifier with RBF kernel to classify the different faults for small fixed-wing UAVs. We use the same real flight dataset for our analysis and make comparisons with the reported results for SVM classifier.

2.2 Stability Quantification through Lipschitz Constant

The number of works studying the stability issue of neural networks has increased in manifolds with those defining and handling the stability of these networks in various manners. A way of quantifying the stability of a trained neural network model consists of studying the Lipschitz properties of the neural network. A Lipschitz constant of a multi-input multi-output function f is an upper bound on the ratio between the variations in the outputs and the variations in inputs of f, and thus it constitutes a measure of sensitivity of the function with respect to input perturbations.

For a feed-forward neural network represented by f where m is the number of hidden layers, N_i the number of neurons in layer $i \in \{1, \ldots, m\}$, let $x \in \mathbb{R}^{N_0}$ be the input and $f(x) \in \mathbb{R}^{N_m}$ be the associated output. If $L \in [0, +\infty[$ is such that, for every input $x \in \mathbb{R}^{N_0}$ and perturbation $z \in \mathbb{R}^{N_0}$, we have

$$\|f(x+z) - f(x)\| \leq L\|z\|, \tag{1}$$

then L is a Lipschitz constant of the neural network f. The smaller this constant value, the more stable the neural network.

A *trivial upper bound* on the Lipschitz constant of a neural network was defined in [7] as the product of spectral norms (S) of each layer independently. Mathematically it can be written as

$$\overline{L}_m = \|W_m\|_{\mathrm{S}} \|W_{m-1}\|_{\mathrm{S}} \cdots \|W_1\|_{\mathrm{S}}. \tag{2}$$

where W_i are the weights/kernels at each layer $i \in \{1, \ldots, m\}$. This bound was found to be too loose for some practical applications.

In [23], it was proved that the problem of computing the exact Lipschitz constant of a neural network with differentiable activation functions is NP-hard. In [3], the authors proposed various bounds on the Lipschitz constant of a feed-forward network under the assumption that the activation operator at each layer is α_i-averaged with $\alpha_i \in]0, 1]$. This assumption is satisfied by most of the widely used activation functions, which are 1/2-averaged [2]. If the activation function at each layer is separable, a general Lipschitz constant estimate, CPLip, is defined as

$$L_m = \sup_{\Lambda_1 \in \mathcal{D}_1, \ldots, \Lambda_{m-1} \in \mathcal{D}_{m-1}} \|W_m \Lambda_{m-1} \cdots \Lambda_1 W_1\|_{\mathrm{S}}, \tag{3}$$

where, for every $i \in \{1, \ldots, m-1\}$, $\mathcal{D}_i$ is the set of diagonal matrices of size $N_i \times N_i$ with diagonal elements equal to $2\alpha_i - 1$ or 1. This estimate remains true even with other norms than the Euclidean, applied to the input and output perturbation spaces. In [13] authors proposed a polynomial constrained optimization (LipOpt) based technique for estimating Lipschitz constant. This estimate can be used for any ℓ_p norm for input and output perturbations, but it is valid for neural network with single output. Solving such an optimization problem can be achieved by solving a hierarchy of convex problems such as Lasserre's hierarchy.

SDP based Lipschitz Constant estimates were explored in [5]. This work focuses on neural networks using separable activation operators. It assumes that the activation functions at all layers are slope-bounded. By using the non-expansiveness property of the activation function, the optimization problem of estimating a Lipschitz constant is recast as solving a semi-definite positive programming (SDP) problem. This estimation can only be used when the Euclidean norm is employed at both input and output spaces. Also, the most optimized technique for estimating the Lipschitz constant in [5] is erroneous, as mentioned and proved in [18]. Hence, for all our results we use the valid version LipSDP-neuron for all our Lipschitz constant estimates.

3 Spectral Normalization Control of the Model Stability

For safety critical tasks, Lipschitz constant and performance targets can be specified as engineering requirements, prior to network training. A Lipschitz constant target can be defined by a safety analysis of the acceptable perturbations for each output knowing the input ranges and it constitutes a current practice in many industrial processes. The Lipschitz constant of a neural network can be arbitrarily high when unconstrained, making it sensitive to adversarial attacks. Imposing this Lipschitz target can be done either by controlling the Lipschitz constant for each layer or for the whole network, depending on the application at hand. Such a work for controlling the Lipschitz constant has been presented in [20] using Hinge regularization. Spectral normalization techniques [15, 17] have been proved to be very effective in controlling stability properties of DNNs and counter adversarial attacks. But they require an exact computation of the spectral norm in (3), which is computationally expensive, and therefore it becomes intractable for deep neural networks. On the other hand, it has been shown

that constraining the Lipschitz constant of DNNs enhances their stability, but it makes the minimization of the loss function harder. Hence, a trade-off between stability and prediction performance needs to be reached [26]. We subsequently propose an iterative procedure allowing us to find the best trade-off.

Given an m-layer fully connected NN architecture (FCN) denoted by T with a Lipschitz constant L_{target} and an accuracy target $\text{acc}_{\text{target}}$. We can constrain the spectral norm of each layer to be less than $\sqrt[m]{L_{\text{target}}}$. According to (2), this ensures that the upper bound on the global Lipschitz constant is less than L_{target}. This bound is however overpessimistic, which means that, by monitoring accurately the value of the actual Lipschitz constant of the network, we can relax the bound on individual linear layers so as to get an improved accuracy. In our proposed approach, this process is done in a closed loop with two targets : good performance ($\text{acc}_{\text{target}}$) and desired stability L_{target}.

The approach is summarized in Algorithm 1. Module $\mathsf{Train}_{\text{SN}}(L, \ldots)$ is spectral normalisation training module which trains the defined neural network model with the Lipschitz constant constraint, i.e. each weight matrices W_i with $i \in \{1, \ldots, m\}$ of the neural network is divided by $(\sigma_i / \sqrt[m]{L})$, where σ_i is the largest singular value of W_i. Module LipEst estimates accurately the Lipschitz constant of the trained neural network using LipSDP-Neuron[5]. Module Predict checks the performance (here classification performance) of the trained model on the validation dataset. If L is small, training will be over-constrained and the accuracy of the model will suffer. To relax this constraint we iteratively increase the value of L through small α factor to find the best value of L_{opt} which would be less than L_{target} and ensure an increase in the performance on the validation dataset to achieve acc_{opt} greater than acc_{target}. We stop the iterations when both conditions are met i.e. accuracy of the trained model is greater than acc_{target} and Lipschitz constant is less than L_{target}.

4 Validation on Drone Flights Data

4.1 Dataset Description

For our fault detection and stability analysis, we use the real-flight dataset provided in [1]. The authors use Paparazzi Autopilot system [11] for possible flight trajectories. The flight test data is captured in outdoor setting on different days with varying environments especially wind speeds and also imperfections in the geometry due to manufacturing of the UAV.

The actuator fault model (F) as proposed in [22] is given by

$$u_{\text{app}} = F u_{\text{com}} + E \tag{4}$$

where u_{com} is the commanded control deflection from the autopilot to the actuators and u_{app} is the applied control deflection (i.e. final movement of the elevon) and E is the offset. In vector form, it can be expressed as

$$\begin{bmatrix} u_{\text{app}_\text{r}} \\ u_{\text{app}_\text{l}} \end{bmatrix} = \begin{bmatrix} f_r & 0 \\ 0 & f_l \end{bmatrix} \begin{bmatrix} u_{\text{com}_\text{r}} \\ u_{\text{com}_\text{l}} \end{bmatrix} + \begin{bmatrix} e_r \\ e_l \end{bmatrix}, \tag{5}$$

Algorithm 1 Spectral Normalisation-based Stability Control Loop

Inputs : training dataset $(X_{\text{train}}, Y_{\text{train}})$, validation dataset $(X_{\text{valid}}, Y_{\text{valid}})$, target Lipschitz constant L_{target}, target accurcay $\text{acc}_{\text{target}}$, neural network T, multiplicative factor $\alpha > 1$, maximum iteration number $n_{\max}$
Output : optimal neural network T_{opt}, Lipschitz constant L_{opt}, and accuracy acc_{opt}
Algo :
$L \leftarrow L_{\text{target}}$, $n \leftarrow 0$
$T \leftarrow \mathsf{Train}_{\text{SN}}(L, X_{\text{train}}, Y_{\text{train}})$
$L_{\text{est}} \leftarrow \text{LipEst}(T)$
$\text{acc} \leftarrow \mathsf{Predict}(T, X_{\text{valid}}, Y_{\text{valid}})$
while $(n < n_{\max})$ and $(L_{\text{est}} < L_{\text{target}})$ and $(\text{acc} < \text{acc}_{\text{target}})$ **do**
    $L \leftarrow \alpha\, L$
    $T \leftarrow \mathsf{Train}_{\text{SN}}(L, X_{\text{train}}, Y_{\text{train}})$
    $L_{\text{est}} \leftarrow \mathsf{LipEst}(T)$
    $\text{acc} \leftarrow \mathsf{Predict}(T, X_{\text{valid}}, Y_{\text{valid}})$
    $n \leftarrow n + 1$
end while
$T_{\text{opt}} \leftarrow T$
$L_{\text{opt}} \leftarrow L_{\text{est}}$
$\text{acc}_{\text{opt}} \leftarrow \text{acc}$
if $n = n_{,\max}$ **then**
    Notification: No trade-off found between Stability and Accuracy
else
    Notification: Trade-off found between Stability and Accuracy
end if

where $()_r$ and and $()_l$ stands for the right and left elevon respectively. The feature set is a vector of length 8 consisting of linear accelerations at three coordinates (a_x, a_y, a_z), angular rates $(\omega_x, \omega_y, \omega_z)$, and auto-pilot commanded controls $(u_{\text{com}_\text{r}}, u_{\text{com}_\text{l}})$ for the two aerodynamic actuators. Depending on the values of f_r and f_l, the fault detection problem can be translated into two kinds of classification problems: 1) Binary Class Classification and 2) Multi-class Classification.

We use three performance measures: classification accuracy (%), F1 score, and Matthews Correlation coefficient (MCC). For stability measure we use Lipschitz constant estimate as calculated using LipSDP-Neuron [5]. We recall that we do not use LipSDP most tight bound LipSDP-Network since it is erroneous as mentioned in the section 2.2.

4.2 Binary Classification Results

We train the data on 12^{th} July and test the trained model on 12^{th} july unseen and 13^{th} July data. It is interesting to use 12^{th} July flight data for training and predict fault for the next day, i.e 13^{th} July since 12^{th} and 13^{th} July had different atmospheric conditions at the time of data collection. For further checking the efficacy and generalization capabilities of the methods, we checked the trained

models on flight data for 21st July. The faults were injected in the right elevon $f_r = 0.3$. Fault code 0 implies 'Nominal fault' and fault code 1 implies fault in the right elevon.

Model	12th July			13th July			21st July		
	Acc. (%)	**F1**	**MCC**	**Acc. (%)**	**F1**	**MCC**	**Acc. (%)**	**F1**	**MCC**
SVM	**98.7**	**0.99**	**0.98**	60.5	0.59	0.38	85.3	0.85	0.72
LogReg	89.8	0.90	0.79	58.0	0.59	0.31	56.6	0.45	0.18
Baseline	98.6	0.99	0.97	**73.3**	**0.74**	**0.52**	**93.1**	**0.93**	**0.86**

Table 1: Binary Classification : Comparison between SVM, logistic regression, and best NN trained with baseline training on 12th July.

Model	12th July			13th July			L
	Acc. (%)	**F1**	**MCC**	**Acc. (%)**	**F1**	**MCC**	
(10)	98.3	0.983	0.967	66.4	0.67	0.41	**12.26**
(10,6)	98.4	0.985	0.969	71.0	0.71	0.47	27.56
(50,10,6)	**98.7**	**0.987**	**0.974**	**73.3**	**0.74**	**0.52**	37.76

Table 2: Binary Classification : Results on 12th and 13th July using NN trained with baseline training on 12th July for various configurations of FCN.

Implementation details: We propose to use fully connected feed-forward networks (FCN) for solving the problem. We train such neural networks of varying depth and width. We use ReLU as the activation function except for the last layer where it is employed with sigmoid for binary and softmax for multi-class classification. We use the cross entropy losses as the loss function for training the model. The feature set is standardized by removing the mean and scaling to unit variance. While training, the dataset is split into 4:1 ratio for train and validation sets. The models are trained using Keras with Tensorflow backend. The initializers are set to Glorot uniform and we use Adam as the optimizer. To capture the temporal dynamics of the system, feature set of length 8 $(a_x, a_y, a_z, \omega_x, \omega_y, \omega_z, u_{\text{com}_\text{r}}, u_{\text{com}_\text{l}})$ is concatenated T times. Therefore, the input feature vector to the model is of length $8T$. We observe a general increasing trend in the testing performance as we increase the value of T, which suggests that concatenating more feature lists from the time history of the flight data improves the performance of the real-time prediction. Hence, we found the value $T = 20$ as optimal for our experiments, adding time history beyond this level did not contribute to the learning and the neural network starts to overfit.

In the **first** set of experiments for binary classification, we train on 12th July flight data and test on of 12th July, 13th, and 21st July and directly compare our NN best models to SVM and logistic regression. SVM classifier is trained with a RBF kernel and is optimized using grid-search. The results are given in Table 1. The results obtained using SVM classifier are satisfactory, better than

for logistic regression, but they fail to generalize on the unseen data from 13$^{\text{th}}$ and 21$^{\text{st}}$ July. We remark it is possible to train neural networks with much better generalization performance on different days with varying weather conditions.

Model	Baseline		Spectral	
	Acc. (%)	L	Acc. (%)	L
(10)	66.4	12.26	75.1	3.7
(10,6)	71.0	27.56	75.3	3.8
(50,10,6)	73.3	37.76	**76.7**	**3.7**

(a) Binary Classification: Results on 13$^{\text{th}}$ July trained on 12$^{\text{th}}$ July data.

Model	Baseline		Spectral	
	Acc. (%)	L	Acc. (%)	L
(50,10)	69.7	31.44	**72.65**	3.8
(100,10)	69.54	34.17	71.07	**3.6**
(100,75,10)	68.9	69.06	71.93	3.7

(b) Multi-class Classification: Results on 23$^{\text{rd}}$ July trained on 21$^{\text{st}}$ July data.

Table 3: Comparison between models trained with baseline and spectral normalization constraint ($L_{\text{target}} = 4$).

Model	Training	No attack		FGSM ($\epsilon = 0.2$)		PGD($\epsilon = 0.2$)	
		Acc. (%)	MCC	Acc. (%)	MCC	Acc. (%)	MCC
(50,10,6)	**Baseline**	98.6	0.97	77.6	0.62	76.0	0.59
	Spectral	98.7	0.97	**81.0**	**0.66**	**79.2**	**0.62**

Table 4: Binary Classification: Results on 12$^{\text{th}}$ July clean data, FGSM and PGD attacks using model trained on 12$^{\text{th}}$ July.

In **second** set of experiments we discuss the stability performance of neural networks which is our primary concern pertaining to stable model design. The comparison of the classification task accuracy as well as the stability performance in terms of Lipschitz constant for a varying number of hidden layers and hidden neurons is shown in Table 2. Notation (10,6) in Table 2 implies that the FCN has two hidden layers with 10 and 6 hidden neurons and so on. In the baseline training, no constraints are employed while training the NN. Lipschitz constant (L) is computed using LipSDP-neuron [5] on the trained network.

Third, we show the performance and stability effect of constraining the training of the neural network by spectral normalization, as introduced in section 3 with Algorithm 1. We chose the same configuration for the neural networks as baseline and introduced constraints, by setting $L_{\text{target}} = 4$ and $\text{acc}_{\text{target}} = 75\%$. Table 3a shows a comparison of the classification accuracy and Lipschitz constant value for baseline training and the new training procedure described in Algorithm 1. We remark that the proposed approach improves both accuracy and stability of neural network models.

We also test and compare the results of the trained neural networks when they are attacked with widely known adversarial attacks such as FGSM [7] and PGD [14]. The attacks are generated using ART toolbox [3]. An adversarial attack is a special case of addition of noise to the input samples. It is useful to test

[3] https://github.com/Trusted-AI/adversarial-robustness-toolbox

robustness of the neural network since the adversarial noise is crafted in a manner to generate misclassification errors. We show the results on 12^{th} July test dataset in Table 4 [4]. We remark that spectral normalised training is better in handling the adversarial attacks, leading to more stable models.

4.3 Multi-class Classification Results

We use the flight data captured on the days of 21^{st} and 23^{rd} July for detecting the faults in right and left elevon. For injecting the faults, efficiency of the right elevon is reduced ($f_r = 0.3$) and similarly for the left control elevon ($f_l = 0.9/0.3$). Fault codes: '0' - nominal , '1' - $f_r = 0.3$, '2' - $f_l = 0.9$ and '3' - $f_l = 0.3$, making a 4-class classification problem. The emphasis has been put on detecting faults on the left elevon which has the effect of a geometric twist. Identical faults are injected in the flight data for both days. For capturing the temporal dynamics in the prediction performance, the optimal value of T is taken to be 20 as in the previous section.

Model	21^{st} July			23^{rd} July		
	Acc. (%)	**F1**	**MCC**	**Acc. (%)**	**F1**	**MCC**
SVM	94.1	0.94	0.91	60.3	0.60	0.51
LogReg	89.0	0.88	0.84	57.8	0.55	0.43
Baseline	**95.2**	**0.95**	**0.93**	**69.7**	**0.68**	**0.58**

Table 5: Multi-class Classification: Comparison between SVM, logistic regression and best NN trained with baseline training on 21^{st} July and tested on 21^{st} and 23^{rd} July .

Model	21^{st} July			23^{rd} July			L
	Acc. (%)	**F1**	**MCC**	**Acc. (%)**	**F1**	**MCC**	
(50,10)	95.2	0.95	0.93	**69.7**	**0.68**	**0.58**	**31.44**
(100,10)	95.3	0.95	0.93	69.5	0.68	0.57	34.17
(100,75,10)	**95.4**	**0.95**	**0.93**	68.9	0.67	0.57	69.06

Table 6: Multi-class Classification : Results on 21^{st} July and 23^{rd} July using NN trained with baseline training on 21^{st} July for various configurations of FCN.

For the experiments, we train a FCN with ReLU activation function for all layers, except the last one for which softmax is used. We train the neural network on the flight data of 21^{st} July and test on data of 21^{st} July not seen by the model at the time of training, and also on the flight data of 23^{rd} July. We show the prediction and generalization capabilities of the neural networks over

[4] In case of SVM model, classification metric MCC had negative values for FGSM and PGD attacks implying model performs very poorly against adversarial attacks, hence we discard them from comparison.

state-of-the-art SVM and logistic regression. We tabulate the classification metrics in Table 5.Varying configurations of model layers and neurons classification metrics along with the stability metric (L) have been considered in Table 6. We consider $\text{acc}_{\text{target}} = 70\%$ and $L_{\text{target}} = 4$ as the model performance and stability requirements. The results are shown in Table 3b.

Observations of the performance and stability analysis:

- Neural networks perform better than SVM classifier and logistic regression in terms of prediction performance indicating better generalization capabilities of neural network models, when trained and tested on different days.
- We observe the trend of increased Lipschitz constant as the number of layers and neurons in the neural network increases.
- There is a steep decrease in the Lipschitz constant value when the models are trained with a Lipschitz target which satisfies the performance target as well Table 3. We also observe an increase of classification performance with spectral normalization constraint.
- Spectral normalised trained models are better against adversarial attacks implying since they are designed to be more stable.

5 Conclusion

Our work provides a formal method by means of estimating Lipschitz constant to evaluate and control the stability of a neural network, which is necessary for many mission-critical and safety-critical systems. This work analyses a real flight data collected on a small fixed-wing UAV in varying flight conditions to design a fully connected feed-forward neural networks that implements a fault detection function embedded in the UAV. The task is to detect mechanical faults like an elevon stucked or locked in a given position even in the presence of noise in the input data coming from the inertial sensors and the autopilot and to allow the remote pilot to take necessary actions to ensure the safety. The main contributions of this work are the following: Through the evaluation of classification performance we have shown that our approach based on DNN compares favorably with SVM and logistic regression techniques. Unlike many works related to the use of neural networks in industrial applications, we address the stability of our trained neural networks using a formal verification method based on the quantification of the Lipschitz constant of the model. We show the effectiveness of spectral normalization techniques in controlling the Lipschitz constant of the network to reach a Lipschitz safety target which can be defined at system level. Following the insights presented in the paper, we show that it is possible to build neural networks which are "Stable-by-Design" and verified through Lipschitz certificates. Such designed models are more robust to adversarial attacks and present a good accuracy. These results regarding stability are strong developments toward learning assurance artefacts in ML models.

References

1. Bronz, M., Baskaya, E., Delahaye, D., Puechmore, S.: Real-time fault detection on small fixed-wing uavs using machine learning. In: DASC. pp. 1–10 (2020)
2. Combettes, P.L., Pesquet, J.C.: Deep neural network structures solving variational inequalities. Set-Valued and Variational Analysis pp. 1–28 (2020)
3. Combettes, P.L., Pesquet, J.C.: Lipschitz certificates for neural network structures driven by averaged activation operators. SIAM Journal on Mathematics of Data Science **2**, 529–557 (2020)
4. Eroglu, B., Sahin, M.C., Ure, N.K.: Autolanding control system design with deep learning based fault estimation. Aerospace Science and Technology **102**, 105855 (2020)
5. Fazlyab, M., Robey, A., Hassani, H., Morari, M., Pappas, G.J.: Efficient and accurate estimation of lipschitz constants for deep neural networks. In: NeurIPS (2019)
6. Freeman, P., Pandita, R., Srivastava, N., Balas, G.J.: Model-based and data-driven fault detection performance for a small uav. IEEE/ASME Transactions on mechatronics **18**(4), 1300–1309 (2013)
7. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)
8. Guo, D., Zhong, M., Ji, H., Liu, Y., Yang, R.: A hybrid feature model and deep learning based fault diagnosis for unmanned aerial vehicle sensors. Neurocomputing **319**, 155–163 (2018)
9. Gupta, K., Kaakai, F., Pesquet-Popescu, B., Pesquet, J.C., Malliaros, F.: Multivariate lipschitz analysis of the stability of neural networks. Frontiers in Signal Processing (2022)
10. Gupta, K., Pesquet, J.C., Pesquet-Popescu, B., Kaakai, F., Malliaros, F.: An adversarial attacker for neural networks in regression problems. In: IJCAI AI Safety Workshop (2021)
11. Hattenberger, G., Bronz, M., Gorraz, M.: Using the Paparazzi UAV System for Scientific Research. In: IMAV. pp. pp 247–252 (Aug 2014)
12. Iannace, G., Ciaburro, G., Trematerra, A.: Fault diagnosis for uav blades using artificial neural network. Robotics **8**(3), 59 (2019)
13. Latorre, F., Rolland, P., Cevher, V.: Lipschitz constant estimation of neural networks via sparse polynomial optimization. arXiv preprint arXiv:2004.08688 (2020)
14. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083 (2017)
15. Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y.: Spectral normalization for generative adversarial networks. arXiv preprint arXiv:1802.05957 (2018)
16. Moosavi-Dezfooli, S.M., Fawzi, A., Frossard, P.: Deepfool: a simple and accurate method to fool deep neural networks. In: CVPR. pp. 2574–2582 (2016)
17. Pan, Z., Mishra, P.: Fast approximate spectral normalization for robust deep neural networks. arXiv preprint arXiv:2103.13815 (2021)
18. Pauli, P., Koch, A., Berberich, J., Allgöwer, F.: Training robust neural networks using lipschitz bounds. arXiv preprint arXiv:2005.02929 (2020)
19. Sadhu, V., Zonouz, S., Pompili, D.: On-board deep-learning-based unmanned aerial vehicle fault cause detection and identification. In: ICRA. pp. 5255–5261. IEEE (2020)
20. Serrurier, M., Mamalet, F., González-Sanz, A., Boissin, T., Loubes, J.M., del Barrio, E.: Achieving robustness in classification using optimal transport with hinge regularization. In: CVPR. pp. 505–514 (2021)

21. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 (2013)
22. Tandale, M.D., Valasek, J.: Fault-tolerant structured adaptive model inversion control. Journal of Guidance, Control, and Dynamics **29**(3), 635–642 (2006)
23. Virmaux, A., Scaman, K.: Lipschitz regularity of deep neural networks: analysis and efficient estimation. In: NeurIPS. pp. 3835–3844 (2018)
24. Weng, T.W., Zhang, H., Chen, P.Y., Yi, J., Su, D., Gao, Y., Hsieh, C.J., Daniel, L.: Evaluating the robustness of neural networks: An extreme value theory approach. arXiv preprint arXiv:1801.10578 (2018)
25. Xie, X.h., Xu, L., Zhou, L., Tan, Y.: Grnn model for fault diagnosis of unmanned helicopter rotor's unbalance. In: ICEECA. pp. 539–547. Springer (2016)
26. Zhang, H., Yu, Y., Jiao, J., Xing, E., El Ghaoui, L., Jordan, M.: Theoretically principled trade-off between robustness and accuracy. In: ICML. pp. 7472–7482. PMLR (2019)