A Comparative Study of Deep Learning Architectures for Detection of Anomalous ADS-B Messages*

Ralph Karam[1], Michel Salomon[1], and Raphaël Couturier[1]

***Abstract*— Since the 1920's, air traffic is becoming more prevalent by the year which results in a steady increase of the number of aircrafts roaming the airspace. This requires the expansion of the air surveillance systems in order to be able to manage each one of these aircrafts. Such an accommodation is planned to be implemented using different technologies and notably the Automatic Dependent Surveillance Broadcast (ADS-B) system. The ADS-B protocol is based on the idea that aircrafts as well as air traffic controllers communicate with each other using messages. However, for practicality reasons, those messages are not encrypted thus malicious messages can be injected. Hence, these attacks need to be detected to ensure the safety of the protocol. In this paper, we evaluate deep learning architectures for the purpose of detecting anomalous/malicious ADS-B messages, especially LSTM architectures which appear to be the most promising ones.**

I. INTRODUCTION

Air transport has witnessed a continuous growth over the years and that is still ongoing. According to the International Air Transport Association, trends suggest that the number of passengers could reach more than 8 billion in 2037. Obviously, the Air Traffic Control (ATC) will have to be able to accommodate the corresponding growth in the number of flights. Therefore, a boost in the capacity of the surveillance systems is needed while simultaneously maintaining high safety levels. To address this challenging target the International Civil Aviation Organization (ICAO) adopted in the early 2000s the Global Air Navigation Plan (GANP). Since its introduction, the GANP has evolved continuously to serve as a worldwide reference to transform the air navigation system in an evolutionary manner. The final objective for the ICAO is to achieve a global interoperable air navigation system, for all users during all phases of flight, that meets agreed levels of safety, provides for optimum economic operations, and is environmentally sustainable.

Among the surveillance technologies present in the ICAO GANP, the Automatic Dependent Surveillance Broadcast (ADS-B) system is supposed to become a cornerstone technology in air transportation systems [1] like the American one called NextGen. In fact, the purpose of ADS-B is to provide an Air Traffic Management / Control (ATM/ATC) surveillance system with a more accurate and precise representation of the 3D position of an aircraft during its on-departure, en-route, and on-arrival operations. Practically, an aircraft broadcasts messages (thanks to an ADS-B Out transmitter) over time (at a data rate of at most 1 Mbit/sec), with each message which can contain different kinds of information like its altitude, velocity, and so on. To compute the position a Global Navigation Satellite System (GNSS) is used, typically Global Positioning System (GPS) satellites.

ADS-B messages are received by other aircrafts in the immediate vicinity and reachable ATM facilities on the ground, allowing a visualization on a controller's screen like the one that would produce a secondary radar. A main objective of ADS-B is to improve the surveillance coverage in areas which have low or no radar coverage. Some air navigation service providers also wish to deactivate radar facilities in some areas, first to save costs due to maintenance and second to reduce the dependence of ATM/ATC on conventional radar. Concerning information flow, there are two main parts in ADS-B: ADS-B Out and ADS-B In. While the first one refers to an aircraft broadcasting its altitude and other information, allowing to track live flights, the latter refers to an aircraft receiving messages from other aircraft in its neighbourhood and from the ATM/ATC ground network. ADS-B In is more precisely devoted to the reception of traffic and weather datalinks. Thus, in the context of a surveillance system, ADS-B Out is of interest to air traffic controllers, while ADS-B In is rather of interest to aircraft pilots. Fig. 1, issued from Global Aerospace Design Corporation website, illustrates the overall information flow scheme.

The design of the ADS-B system was mainly driven

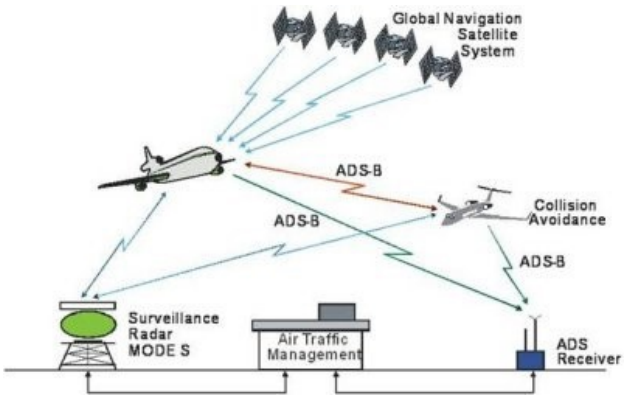

Fig. 1. Information flow in ADS-B system (illustration from Global Aerospace Design Corporation website)

*This work was supported by the DGA (French defence procurement agency) in the context of the GeLeaD project (project number ANR-18-ASTR-0011) related to the ANR ASTRID research program (specific support scheme for research works and innovation defence). It was also partially supported by the EIPHI Graduate School (contract ANR-17-EURE-0002). Computations have been performed on the supercomputer facilities of the "Mésocentre de Franche-Comté".

[1]R. Karam, M. Salomon, and R. Couturier are with FEMTO-ST Institute, Univ. Bourgogne Franche-Comté (UBFC), CNRS, France. e-mail: ralph.karam,michel.salomon,raphael.couturier@univ-fcomte.fr.

by cost, surveillance coverage, and accuracy, with limited security considerations. As a result, ADS-B, which is now widely deployed, is a system prone to vulnerabilities and threats. Typical minimal security services to protect communication protocols are missing since there is no message authentication or encryption and no device authentication. These lacks open major flaws since they allow an attacker to access classical spoofing and replay attacks, to eavesdrop on message, and the ability to inject or alter messages by using an unauthorized ADS-B device [2].

This paper focuses on attacks that alter the semantics of the ADS-B data while preserving their syntactical correctness and logical consistency. This kind of cyberattacks, known as False Data Injection Attacks (FDIAs), can potentially have high impacts on national security and many economic sectors. Indeed, several civil and defense control systems, such as SCADA, the IoT or those of autonomous vehicles [3], etc., might be misled by FDIAs. For example, smart grid hacking [4] in electricity networks attempts to send back to the controller (SCADA system) fake sensor measurements to alter its operation. Hence, the purpose of our study is to mitigate such threats by designing a tool able to detect FDIA attacks that target the ADS-B system.

Over the past years machine learning and particularly its hip branch known as deep learning [5], [6] has received an increasing interest in the cybersecurity field. Malware detection and network intrusion detection are two areas where deep learning-based approaches have produced major improvements over rule-based and classic machine learning-based solutions. In fact, these security threats can be considered as an anomaly detection which is a common problem addressed by machine learning. Obviously, a key point for the success of deep learning is the access to relevant data since it needs extensive datasets. In this work, we will consider a very basic attack scenario on ADS-B messages, namely a rough change of the altitude value in some messages.

The remainder of this paper is organized as follows. Section 2 gives a brief summary of the techniques present in the literature used for anomaly detection. The next section presents the studied machine learning architectures for the case of alteration scenario targeting the altitude value in ADS-B message. Section 4 presents the experimental work done with these models. Finally, we draw some conclusions for the design of a new architecture for FDIAs detection.

II. RELATED WORK

There exists a considerable literature on anomaly detection using machine learning techniques (mainly unsupervised). These techniques belong to two main classes: classical artificial intelligence methods and neural networks based methods. In the context of anomaly detection in the aviation domain, Basora *et al.* [7] have proposed a taxonomy of classical methods that includes neural networks methods. But as they noticed, the recent advances are mainly issued from the field of neural networks and more particularly deep learning. Therefore we think it is more relevant to consider neural networks aside from other classical methods.

A. Classical Artificial Intelligence Methods

These techniques rely on different approaches: local outlier factor computation, clustering, isolation forest, statistical methods, boundary based methods, principal component analysis (PCA) and its variants.

First, the local outlier factor (LOF) is a score used to detect anomalous points. Intuitively, the LOF score compares the local density (point density) of a specific data point relatively to its k nearest neighbours (k is a hyperparameter to be fixed) and if it detects that it has a lower density then it is considered as an outlier [8]. Second, a clustering denotes the act of partitioning a dataset in an unsupervised manner. This partitioning results in multiple clusters which group data points together according to a similarity measure such as a pairwise distance. Many clustering algorithms were used for anomaly detection like: DBSCAN [9], HDBSCAN [10], OPTICS [11] (these three algorithms give a cluster for anomalies and one for normal data), or IMS [12] (which uses the distance of points from the centroid of their associated clusters as an anomaly measure: distant points are anomalous). Third, an isolation forest [13] rely on the idea that outliers are faster to isolate than inliers. Isolation forests use a set of decision trees for classifying each point. The shorter the path used for deciding if the studied point is anomalous, the more likely it is an outlier. Fourth, in statistical methods of anomaly detection (found in Pimentel *et al.*'s review [14]) the data is assumed to follow a normal distribution. These methods rely on the fact that points which are far from the mean of the distribution should be considered as outliers. Fifth, in boundary methods a boundary between normal and anomalous data is learned from training data for the subsequent identification of outliers. In this kind of methods, one notable technique used is the one class support vector machine [15]. This method uses a linear boundary to separate data. To ensure the presence of such a boundary, the data is projected in a linearly separable space (kernel trick). Finally, in PCA methods data points are considered anomalous if their last principal components are relatively large [14].

B. Neural Networks based Methods

Many techniques of anomaly detection in time series belong to neural network methods such as: autoencoders (AE), recurrent neural networks (RNN), convolutional neural networks (CNN), and generative adversarial networks (GAN). Some proposed architectures also combine some of them, typically a combination of RNN and CNN.

Autoencoders [16] are neural networks used to encode input data into a compressed representation in a latent space and then to reconstruct the data such as the output data is the closest possible to the input one. These autoencoders are made of two blocks. First, the encoder maps the input data in a latent space by using several stacked downsampling layers up to the bottleneck layer where the data lies in the latent space. The second block, the decoder, maps back the low-dimensional representation of the data starting from bottleneck layer thanks to upsampling layers. In the case of anomaly detection, autoencoders are usually trained solely

on normal data such that they can identify outliers using the reconstruction error values. More precisely, once trained such an autoencoder cannot properly reconstruct anomalous data and thus it allows to detect them when the similarity between the input and output data is large. In the literature autoencoders are usually combined with other architectures. For example, Malhotra *et al.* [17] combined long short-term memory (LSTM) networks (recurrent neural networks suitable for sequential data) with autoencoders in order to be able to detect outliers in time series. Zhang *et al.* [18] merged convolutional architectures with LSTM and autoencoders for outlier identification. They used a convolutional encoder to extract spatial information, an attention based ConvLSTM to extract temporal patterns, and a convolutional decoder for the reconstruction of the input. A ConvLSTM is a variant of LSTM in which convolution operations replace internal matrix multiplications within LSTM unit. In [19], to detect outliers a multi-modal deep autoencoder approach was used. In this architecture, multiple time series data are input into the network: a sliding window is used on the time series simultaneously. Time series portions undergo a concatenation into one vector which is then entered into the autoencoder. Finally the reconstruction error is used to detect outliers.

Recurrent neural networks (more specifically stacked LSTM networks) can be used to detect anomalies in time series using a regression [20]. These networks learn to obtain the following values of the time series using a training set of normal data. After the learning step, the network can detect outliers by computing the residuals (errors). When a residual is above a certain threshold then an anomaly is detected. In [21], CNN networks and hybrid networks (CNN-RNN, CNN-LSTM, CNN-GRU) were used for network intrusion detection using a supervised learning approach. In their paper, TCP/IP packets are modeled as a time series and 1D convolutional filters are used. Each packet is transformed into a vector which is input into the network for training and then testing. In [22] a GAN is used for time series anomaly detection. A GAN is a network which tries to generate fake data which cannot be distinguished from real data. This is done by training a generator to generate fake data and a discriminator to detect the fake data. When the discriminator cannot do the distinction anymore the training stops. In [22], after training the GAN with normal time series data, the network can detect outliers by using an anomaly score. This anomaly score is equal to residual plus discriminator error. Note that the residual is the error computed when the generator tries to generate the test data.

More specifically, in the context of anomalous ADS-B message detection, Habler and Shabtai have proposed a LSTM autoencoder [23]. This latter models a flight route through the analysis of sequences of normal messages and, as explained above, evaluates the deviation of a received message from the legitimate flight route. Experiments were completed while considering six flight routes and various message alterations showed the relevance of their proposal. Another interesting work is the one due to Cretin *et al.* [24] who studied the utility of DSL-based testing against FDIAs.

III. STUDIED ARCHITECTURES

As noticed previously, in ADS-B protocol, aircrafts broadcast messages over time during their operations. Consequently, the data to process to detect a FDIA targeting an aircraft consist in a set of time series obtained after decoding the received ADS-B messages. The fact that the data have a temporal evolution indicates that machine learning models able to keep information from the past should be the most suited, namely neural networks with a recurrent architecture. Natural candidate deep learning architectures for examination are therefore the LSTM architecture and its variants. Supervised learning will be applied using these architectures since it is rarely investigated in the literature for the case of anomaly detection. Apart from neural networks we will also investigate a Gradient Boosting method, a technique attracting attention for its prediction speed and accuracy.

A. Gradient Boosting (XGBoost)

XGBoost, which stands for eXtreme Gradient Boosting [25], is a recent gradient boosting algorithm designed for speed and performance. Gradient boosting algorithm is a kind of machine learning technique suited for solving regression and classification problems. This kind of algorithm produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees.

B. Long Short-Term Memory (LSTM)

A LSTM [26] is a recurrent network of cells / units with short-term and long-term memory. The idea associated with the LSTM is that each computational unit is linked not only to a hidden state h but also to a state c of the cell that acts as a memory. The transition from c_{t-1} to c_t is done by constant gain transfer equal to 1. In this way, errors are propagated at previous steps (up to 1,000 steps in the past) without gradient vanishing phenomenon. The status of the cell can be modified through a door that allows or blocks the update (input gate). Similarly, a door controls whether the cell state is communicated at the output gate of the unit.

C. Convolutional Neural Network (CNN)

A convolutional neural network [27] is a rather different type of neural network compared to a LSTM. Indeed, a CNN is a well-known feed-forward neural network in the deep learning field, in which the pattern of connection between the neurons is inspired by the visual cortex of the animals. The neurons in this region of the brain are arranged so that they correspond to overlapping regions when the visual field is paved. Even if CNNs are mainly used for image processing, they can also be used for other kinds of input data like audio. In our case study, we have used a CNN before using a LSTM in order to extract features from input data.

IV. EXPERIMENTAL WORK

In order to perform a first evaluation of the ability of different deep learning models to detect FDIAs in the ADS-B case study, we have performed some experiments using data in which the altitude values have been roughly altered.

Therefore we took some individual ADS-B messages of several aircrafts and we changed the altitude by adding to the original value a random number equal to $+$ or $-(800 + \mathrm{random}(200))$ feets. Some altitude values are thus increased by adding a value between 800 and 1,000 ft, while others are decreased with a value in the same range. Overall, for a given aircraft approximately one out of 30 messages has an altered altitude value. The data used for training and testing respectively consist of messages from 10 and 4 aircrafts ($66,172$ messages for training and $19,827$ for testing). Since preexisting deep learning architectures are used for anomaly detection in our study, and there is no need for low level manipulation of neural networks, the Keras python library is used to build deep learning anomaly detection tools. For more general machine learning techniques such as gradient boosting the Sklearn python library is used.

A. Data Acquisition

The data used to evaluate the different models are issued from The OpenSky Network [28], a community-based receiver network which continuously collects air traffic surveillance data. In order to obtain the data needed for the anomaly detection, a data pipe based on several python scripts was developed in order to enable the querying of the OpenSky database. More precisely we downloaded raw ADS-B messages and decoded them using the traffic and pyModeS python libraries. Currently, to build our dataset we have used air traffic surveillance data collected on the 13th of January 2019 between 1PM and 2PM GMT near the swiss border.

B. Data Format

The ADS-B protocol comes in 4 different message types: Aircraft Identification, Airborne Position, Airborne Velocity, and Surface Position. Each type of message contains its own information and is sent by the aircraft transponder in a fixed period of time (5 seconds for Identification messages and 0.5 seconds for the others). For example, an ADS-B message of type Surface Position contains the following information: altitude, ground speed, track, latitude, longitude.

C. Confusion Matrix

To assess the relevance of the different machine learning models we used confusion matrices to evaluate their accuracy results. A confusion matrix sums up the prediction results of a classifier. Such a matrix is constructed as follows: each line corresponds to a real class and each column corresponds to a predicted class. For example, in our case where there are two classes (there is an attack / Positive — the altitude value has been altered — or not / Negative) and the matrix looks like:

$$\begin{bmatrix} TN & FP \\ FN & TP \end{bmatrix},$$

where, in the case of anomalous messages detection,

- TN represents the number of True Negative predictions, in other words the number of correctly detected messages without attack;

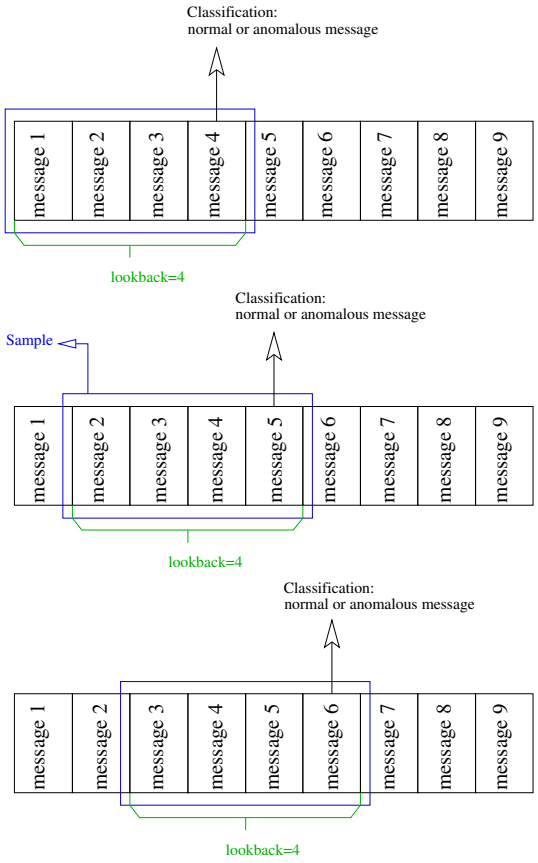

Fig. 2. Classification of the last message of three successive samples

- FN, which stands for False Negative, counts the number of altered messages which are classified as not attacked;
- TP means True Positive and thus represents the number of attacked messages which were correctly detected;
- Finally, FP (for False Positive) corresponds to the number of messages without alterations classified as attacked messages.

The objective is then to find a classifier whose predictions produce a diagonal matrix.

In the context of the classification of messages, we define an input sample as a sequence of messages and call its size lookback. Each sample is then used to classify its last message as normal or anomalous as shown in Fig. 2.

As an illustration of application of the confusion matrix to evaluate the accuracy of a classifier, we present the evaluation of the XGBoost algorithm for anomaly detection. The evaluation of this algorithm is straightforward since XGBoost is directly available in python as a module. Using a lookback value of 50, the obtained confusion matrix is:

$$\begin{bmatrix} 14,400 & 4,557 \\ 17 & 657 \end{bmatrix}.$$

To obtain the number of samples used for testing, the following formula can be used: $S = T - N \times (L - 1)$, where S is the number of test samples effectively used to evaluate a classifier, T is the total number of message in testing set, N is the number of aircrafts whose messages

are used for testing, and L is the lookback value. Using the previous formula, given the experiment setup ($T = 19,827$, $L = 50$, $N = 4$), the number of samples used to evaluate the XGBoost algorithm is equal to $19,631$. Note that the sum of the elements of the confusion matrix is also equal to the number of samples used for testing i.e. $19,631$.

According to the confusion matrix, we can observe that the number of FP is very large implying a low precision (12.6%), which means that XGBoost predictions are quite mitigate. We plan to investigate in future works the optimization of the hyperparameters of the XGBoost algorithm in order to obtain an improved performance. In addition to the XGBoost algorithm, we investigated different deep learning architectures

D. LSTM Architecture Evaluation

A stacked LSTM architecture with two layers has been considered. As we need to start with a fixed number of units in each layer, we have chosen 256 units in the first layer and 128 units in the second layer. Later we will see the influence of the LSTM architecture by considering different settings for the sizes of the two layers.

First, different optimizers have been compared: ADAM, ADAMAX, NADAM, RMSPROP, and the classical SGD. The parameters controlling the training are set as follows: the maximum number of epochs is set to $3,000$, an early stopping condition on the training loss error is set to $5e{-}4$, and a lookback value of 15 is chosen. Table I, which presents the obtained respective Precision and Recall, as well as the F-score and the number of epochs needed to converge, shows that NADAM is the optimizer providing the better detection performance. Indeed it has the highest percentages, which means the best detection ability, and it is the fastest optimizer to converge because it requires only 312 epochs. Note that in this case the number of messages (or samples) used for testing is equal to $19,771$ while it was $19,631$ for XGBoost. This increase in the number of samples is explained by the lower lookback value (15 versus 50).

Focusing on NADAM, we can observe that the size of the lookback influences greatly the number of epochs to reach the convergence and the quality of the prediction. According to our experiments, as shown in Table II, a lookback value of 35 allows to obtain the best results but it requires a larger number of epochs to converge. The sizes of the different layers appear to have also a great impact on the detection ability of the two layer LSTM architecture. Table III, which presents the percentages gained for different layer sizes configurations, highlights a best setting of 64 units for the first layer and 32 for the second one.

TABLE I

EVALUATION OF DIFFERENT OPTIMIZERS ON A STACKED LSTM (LAYERS OF 256 AND 128 UNITS - LOOKBACK VALUE OF 15)

Optimizer	%Precision	%Recall	%F-score	Number of epochs for convergence
ADAM	71.6	96.62	82.25	788
RMSPROP	55.1	84.88	66.82	2,223
SGD				did not converge
NADAM	82.6	97.65	89.50	312
ADAMAX	71.58	97.65	82.61	536

TABLE II

EVALUATION OF DIFFERENT LOOKBACK VALUES ON A STACKED LSTM (LAYERS OF 256 AND 128 UNITS - NADAM OPTIMIZER - *: EARLY STOPPING LOSS SET TO $5e{-}3$)

Lookback	%Precision	%Recall	%F-score	Number of epochs for convergence
5*	91.91	96.63	94.21	685
10	76.48	98.38	86.06	1,635
15	80.05	98.38	88.27	314
20	81.67	97.79	89.01	338
25	76.90	97.05	85.81	598
30	83.83	98.53	90.59	288
35	93.04	98.82	95.84	1,871
40	86.57	98.37	92.09	1,477

TABLE III

EVALUATION OF DIFFERENT STACKED LSTM ARCHITECTURES (TWO LAYERS - NADAM OPTIMIZER - LOOKBACK VALUE OF 20

Layer (units) 1	2	%Precision	%Recall	%F-score	Number of epochs for convergence
16	8				did not converge
32	16				did not converge
64	32	91.59	99.41	95.34	1,134
128	64	74.36	98.23	84.64	1,339
256	128	81.67	97.79	89.01	20,598

E. Bidirectional LSTM Evaluation

Previously we have considered a unidirectional LSTM architecture, but it is possible to use a bidirectional architecture [29]. In that case, the network consists of two different hidden layers: one that processes the input sequence forward, like in the unidirectional architecture, whereas the other one processes it backward. Using a bidirectional LSTM made of two unidirectional LSTM layers (256 units in first layer and 128 units in second one) we obtained the following result with NADAM optimizer and lookback set to 20: 77.32 %Precision, 84.54 %Recall, and 79.94 %F-score after $1,531$ epochs. Clearly this kind of architecture is not interesting.

F. CNN Architecture

Convolutional neural networks have been used for many years to make very efficient image classifications. In our case study, considering a lookback value of 20 and the temporal data of an aircraft consisting of 5 different multivariate data series (altitude, latitude, etc.), we can see them as a 2D image of size 20×5. In practice, a convolutional layer followed by dense layers can be used. However, such a network converges very slowly and cannot reach the required precision in $3,000$ epochs. Nevertheless, it provides the following result: 84.46 %Precision, 89.69 %Recall, and 87 %F-score, which is not surprising because CNNs are not designed for this kind of classification

TABLE IV

EVALUATION OF A 1D CNN CONNECTED TO A STACKED LSTM FOR DIFFERENT LOOKBACK VALUES (THREE LAYERS - NADAM OPTIMIZER)

Lookback	%Precision	%Recall	%F-score	Number of epochs for convergence
10	77.32	96.62	85.90	565
15	84.38	98.38	90.85	601
20	83.19	97.64	89.84	737

G. Using CNN and LSTM Simultaneously

It is possible to make one step of 1D CNN and then to feed its output in a LSTM. This kind of architecture is sometimes used with time series. As a case study, we evaluated an architecture made of 16 convolution kernels of size 3 which are entered into a 3 layers LSTM (256, 128, and 64 units respectively in layer 1, 2, and 3). The results of the evaluation are summarized in Table IV. Choosing a first layer of convolutional neurons provides good results. Currently the optimal size has not been investigated. Nevertheless, this kind of architecture will be more deeply studied in the future.

V. CONCLUSION AND FUTURE WORKS

We have presented a first evaluation of the ability of some machine learning models, and more particularly deep learning ones, to detect a rough FDIA consisting in the alteration of the altitude value. The eXtreme Gradient Boosting algorithm, the well-known LSTM architecture, its bidirectional variant and a combination with a layer of convolutional neurons in order to change the representation of the input data, and finally, a CNN by viewing the input data as an image, were evaluated. The experiments show that, as expected due to the temporal evolution of the data, the LSTM architecture appears to be the most suited for the considered problem. Different optimizers, lookback values, and settings of the LSTM architecture have been compared. The best detection results have been obtained with a stacked LSTM of two layers composed of 64 units in the first one and 32 in the second, a lookback of 20 time steps, trained with NADAM optimizer. It should be noticed that these results are quite good considering that only 10 aircrafts have been used for the training. In the future, a larger number of aircrafts will be considered and a careful tuning of the hyperparameters studied. To sum up, this experimental study has yielded promising results which will be deepened in future works.

REFERENCES

[1] M. Avila, "Global Air Navigation Surveillance Consideration", NAM/CAR/SAM ADS-B Implementation Meeting/Workshop, Lima, Peru, 13-16 November 2017.

[2] A. Costin and A. Francillon, "Ghost in the air (traffic): On insecurity of ads-b protocol and practical attacks on ads-b devices," *Black Hat USA*, pp. 1–12, 2012.

[3] M. Amoozadeh, A. Raghuramu, C.-N. Chuah, D. Ghosal, H. M. Zhang, J. Rowe, and K. Levitt, "Security vulnerabilities of connected vehicle streams and their impact on cooperative driving," *IEEE Communications Magazine*, vol. 53, pp. 126–132, 6 2015.

[4] G. Liang, J. Zhao, F. Luo, S. R. Weller, and Z. Y. Dong, "A review of false data injection attacks against modern power systems," *IEEE Transactions on Smart Grid*, vol. 8, pp. 1630–1638, 7 2017.

[5] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[6] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[7] L. Basora, X. Olive, and T. Dubot, "Recent advances in anomaly detection methods applied to aviation," *Aerospace*, vol. 6, p. 117, Oct 2019.

[8] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," in *ACM sigmod record*, vol. 29, pp. 93–104, ACM, 2000.

[9] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *Kdd*, vol. 96, pp. 226–231, 1996.

[10] R. J. Campello, D. Moulavi, and J. Sander, "Density-based clustering based on hierarchical density estimates," in *Pacific-Asia conference on knowledge discovery and data mining*, pp. 160–172, Springer, 2013.

[11] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: ordering points to identify the clustering structure," in *ACM Sigmod record*, vol. 28, pp. 49–60, ACM, 1999.

[12] D. L. Iverson, R. Martin, M. Schwabacher, L. Spirkovska, W. Taylor, R. Mackey, J. P. Castle, and V. Baskaran, "General purpose data-driven monitoring for space operations," *Journal of Aerospace Computing, Information, and Communication*, vol. 9, no. 2, pp. 26–44, 2012.

[13] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*, pp. 413–422, IEEE, 2008.

[14] M. A. Pimentel, D. Clifton, L. Clifton, and L. Tarassenko, "A review of novelty detection," *Signal Processing*, vol. 99, pp. 215–249, 2014.

[15] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt, "Support vector method for novelty detection," in *Advances in neural information processing systems*, pp. 582–588, 2000.

[16] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures," in *Proceedings of ICML workshop on unsupervised and transfer learning*, pp. 37–49, 2012.

[17] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "Lstm-based encoder-decoder for multi-sensor anomaly detection," *arXiv preprint arXiv:1607.00148*, 2016.

[18] C. Zhang, D. Song, Y. Chen, X. Feng, C. Lumezanu, W. Cheng, J. Ni, B. Zong, H. Chen, and N. V. Chawla, "A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 1409–1416, 2019.

[19] K. K. Reddy, S. Sarkar, V. Venugopalan, and M. Giering, "Anomaly detection and fault disambiguation in large flight data: a multi-modal deep auto-encoder approach," in *Annual Conference of the Prognostics and Health Management Society*, 2016.

[20] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series," in *Proceedings*, p. 89, Presses universitaires de Louvain, 2015.

[21] R. Vinayakumar, K. Soman, and P. Poornachandran, "Applying convolutional neural network for network intrusion detection," in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 1222–1228, IEEE, 2017.

[22] D. Li, D. Chen, J. Goh, and S.-k. Ng, "Anomaly detection with generative adversarial networks for multivariate time series," *arXiv preprint arXiv:1809.04758*, 2018.

[23] E. Habler and A. Shabtai, "Using lstm encoder-decoder algorithm for detecting anomalous ads-b messages," *Computers & Security*, vol. 78, pp. 155–173, 2017.

[24] A. Cretin, B. Legeard, F. Peureux, and A. Vernotte, "Increasing the resilience of atc systems against false data injection attacks using dsl-based testing," in *Proc. Doctoral Symp.(ICRAT)*, pp. 1–3, 2018.

[25] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, ACM, 2016.

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

[27] Y. LeCun, Y. Bengio, *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[28] "The OpenSky network – Free ADS-B and Mode S data for research." https://opensky-network.org.

[29] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural networks*, vol. 18, no. 5-6, pp. 602–610, 2005.