Real-time Classification of Aircrafts Manoeuvers

Sami Jouaber, Silvère Bonnabel, Santiago Velasco-Forero, Marion Pilté, Jesús Angulo

Real-time Classification of Aircrafts Manoeuvers

Jouaber Sami[1,3*], Bonnabel Silvère[1,4†], Velasco-Forero Santiago[2†], Pilté Marion[3] and Angulo Jesus[2]

[1*]Center for Robotics (CAOR), Mines ParisTech, Paris, France.
[2]Centre de Morphologie Mathématique (CMM), Mines ParisTech, Fontainebleau, France.
[3]LAS, THALES, Massy, France.
[4]ISEA, UNC, Nouméa, France.

*Corresponding author(s). E-mail(s): sami.jouaber@mines-paristech.fr;
Contributing authors: silvere.bonnabel@mines-paristech.fr; santiago.velasco@mines-paristech.fr; marion.pilte@thalesgroup.com; jesus.angulo@mines-paristech.fr;
†These authors contributed equally to this work.

Abstract

Whether it be in air defense applications or for air traffic control it is highly desirable to be able to assess in real time the type of aircraft one is dealing with. This task may prove useful when the object refuses to cooperate or to confront the transmitted information with the observed trajectory. In the present paper we advocate an approach based on position (radar) measurements only, for versatility. Besides, we propose to focus on rotation-invariant kinematic trajectory features such as absolute velocity, curvature, torsion and parameters alike, as relevant distinctive features to automatically classify aircraft trajectories in real time. Those features are fed into convolutional neural networks that are state of the art for time series classification. Notably they seamlessly handle trajectories of variable length and hence may be used in real time. The constructed classifiers are trained with real data collected from publicly available information transmitted by the aircrafts. This allows for benchmarking of the proposed learning algorithms, as well as discussion on the best possible achievable accuracy.

Keywords: Radar processing, Air traffic control, Air traffic management, Target classification algorithms, Neural networks

1 Introduction

To increase the air traffic security and safety level it is useful to find ways to assess in real time the kind of object that is flying. Notably, the growth of Unmanned Aerial Vehicles (UAVs) integrating with the civilian airspace, whether for commercial purposes, or private leisure use, or possibly for malicious purposes, is a driver for novel algorithms in this direction. Indication on the type of aircraft comes with capabilities evaluation and thus may allow for threat assessment in turn.

The problem of aircraft type classification based on radar measurements was previously addressed using range profile in [1, 2], micro-Doppler in [3, 4] or Synthetic Aperture Radar (SAR) imaging [5–8]. However such measurements may not be available. In the present paper we focus on classification only based on radar position measurements.

Real-time classification of flying targets was to our best knowledge pioneered by [9] and their invention patented in [10]. The primary goal was to distinguish between aircrafts and biological targets such as birds, the challenge being that, e.g., a helicopter might fly as slow as a bird, and the secondary goal is to detect aircrafts lacking Secondary Surveillance Radar (SSR) data. They followed up on their approach for specifically detecting UAVs in the civilian airspace in [11]. The technique they advocate consists in extracting features from the measurements such as average velocity and total travelled distance, and then classify the targets using a training set and a classifier such as a standard Multilayer Perceptron Neural Network or Support Vector Machines. At a general level, the present paper differs from this prior line of research in at least two important ways. First, we address a different problem that consists in assessing a number of characteristics of flying aircrafts without addressing small-scale vehicles such as drones or biological objects. Then, we leverage the recent literature on time series classification that exploits advanced deep learning techniques such as Fully Convolutional Neural Networks (FCN) and Residual Networks (ResNet), see [12, 13]. At a more technical level, we advocate using different kinematic features that characterize the trajectory. We notably build on the insight of [14] that the torsion of the curve is a distinctive feature for high velocity targets such as fighters.

Our main contributions may be summarized as follows:

- To address the problem of real-time classification of aircrafts from position measurements, we collected Automatic Dependent Surveillance-Broadcast (ADS-B) data to form a dataset.
- For classification purposes, we advocate the use of Euclidean invariant features, namely the absolute velocity, curvature, torsion, altitude and its time

derivatives. Those may be directly extracted after a smoothing step based on local quadratic approximations of the trajectory, rendering the algorithm immune to time step variations and real-time.
- Three different classifiers based on recent methods from the literature of convolutional neural networks applied to time series are proposed to classify the aircrafts based on the extracted features: they notably output a probability the aircraft belongs to each class. They are shown to be executable in real-time and hence seamlessly accomodate trajectories whose length varies over time.
- The approach of collecting ADS-B data and retaining the associated categories for classification being novel to our best knowledge, we discuss the highest achievable accuracy for this kind of technique in function of the desired granularity of classification given that different types of aircrafts may exhibit identical trajectories.

The proposed methodology has been the object of a recent patent by the radar company Thales [15] as it may prove useful for air traffic management purposes, but has never been detailed and published in a scientific venue.

The remainder of the paper is organized as follows. Section 2 is devoted to the training data. In Section 3 the features that shall be used to classify trajectories are discussed and the extraction algorithm is presented. Section 4 then summarizes relevant literature on neural networks for time series classification. Building on it, we propose and detail our convolutional network-based classifiers in Section 5. Finally, Section 6 is devoted to experimental results and discussion.

2 Collected data

Using the Automatic dependent surveillance-broadcast (ADS-B) protocol, planes, helicopters, gyrocopters, tiltwing aircrafts as well as some ground vehicles broadcast their identity along with their location based on satellite navigation. This identity is then associated with an existing radar track. The information is publicly available and can be retrieved using appropriate receptors. It may thus serve as a labelled dataset in order to perform supervised learning, which is the route we have taken.

To this end, we used the ADS-B Exchange platform [16] to scrap aircrafts positions in a $1500km$ radius area in Europe for hours. Aside from GPS position and altitude, each broadcast can be associated with a specific aircraft by its International Civil Aviation Organization (ICAO) identifier, so its trajectory through time can be extracted. The ADS-B data fields also include some specifications about the aircraft, such as its model, species (airplane or helicopter), Wake Turbulence Category (WTC) which reflects the size of the object, type of engines (piston, turboprop or jet) and number of engines. These last four characteristics are used to label the aircraft with a four-digit tag as described in Table 1. For instance, by looking at the table, we see the label "1231" refers to an aircraft which is a medium airplane mounted with a single

Table 1 Description of the labels of the dataset

Position of the digit	Related characteristic	Description
#1	Species	1: airplane 4: helicopter
#2	WTC	1: light 2: medium 3: heavy
#3	Engine type	1: piston 2: turboprop 3: jet
#4	Number of engines	integer

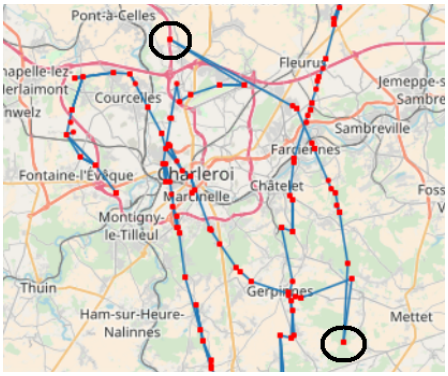

Fig. 1 Example of a raw trajectory, a Belgian fighter F-16 maneuvering around Charleroi. All the received locations are represented by red squares, consecutive positions are linked by a blue line. Some points seem to be way off the real trajectory, especially the two circled in black, revealing noise in the data.

jet engine, which mostly corresponds to fighters (surprisingly we were able to retrieve actual fighter trajectories from publicly available ADS-B data). The label is then expressed with a one-hot encoding to be the expected output. After the acquisition, 19 different labels were represented by more than 50 trajectories, the others being represented by less than 20 trajectories. Thus, we selected 50 trajectories for each of these 19 labels to build the dataset. Doing so, we chose to randomly discard some trajectories to obtain more balanced classes, to notably avoid overrepresentation of the tag "1232" which includes most of the commercial airliners.

An example of collected raw data is displayed on Figure 1. The location data are noisy because they come from many sources with different defects, mostly in the timestamps. For that reason, and also because we would like to enforce a regular time step to allow convolutional networks to work properly, the data had to be preprocessed. First the measured coordinates were converted to a Cartesian frame with all axes in meters. Then, to compute the position on a given axis x at a given time t, as well as various kinematic features based on its derivatives, we devised a smoothing algorithm that we now present.

3 Trajectory Smoothing and Feature Extraction

Our goal is to come up with an algorithm capable of inferring the aircraft category from position measurements only, as typically obtained through a radar. To this end, we consider the following features as desirable requirements:

1. the algorithm must be invariant to horizontal rotations and translations;
2. the algorithm must seamlessly handle trajectories with variable length;
3. the algorithm associates a probability with each category in which the observed aircraft may fall into.

The first requirement concerns the physics of the problem, as there is no reason why the North or the East direction shall be privileged to identify the class of the aircraft, as aircrafts' trajectories relative to departure and arrival are arbitrary. Its rationale is thus to match the physics of the motion. The second requirement has already been mentioned in the Introduction and is an obvious (albeit not trivial to accommodate) requirement for the algorithm to be real-time. Finally, the latest requirement is desirable for interpretability, especially in a context where some trajectories may be indistinguishable, although corresponding to different types of aircrafts.

To meet the three requirements above, as well as to pre-process the data in the form of a smoothing algorithm that is robust to noisy measurements and immune to varying step size, we locally approximate the trajectory as a quadratic function. To this end we regress the three polynomial coefficients that define the quadratic approximation using a Gaussian kernel smoother with radius τ as follows:

$$x(t) = \left[\underset{a,b,c}{\arg\min} \sum_i e^{\frac{-(t-t_i)^2}{2\tau^2}} \left(\begin{bmatrix} a \\ b \\ c \end{bmatrix} . \begin{bmatrix} t_i^2 \\ t_i \\ 1 \end{bmatrix} - x_i \right)^2 \right] . \begin{bmatrix} t^2 \\ t \\ 1 \end{bmatrix},$$

where the parameters a, b, c are easily computable through standard least squares formulas, while the parameter τ was set to $5s$ for our dataset. This value was chosen to be a few times longer than the usual $1s$ update rate of the ADS-B. This technique allows us to approximate the position of the aircraft at arbitrary time t. Moreover, by differentiation some kinematic features such as absolute velocity, curvature and torsion may then be extracted. Owing to Requirement 1 above, we only kept the features which are invariant to translations and rotations in the horizontal plane, namely speed, curvature, torsion, altitude and its three first derivatives. For each trajectory, we randomly selected ten minutes-long time segments sampled with a time-step of $0.1s$ to compute these features.

This process finally led to a dataset of shape $(9500 \times 6001 \times 7)$ for our classifiers input and (9500×19) for the expected output. For training purposes, this dataset was split in two, 70% were used for gradient computation and gradient-based optimization of the classifier and 30% were used to monitor the evolution of the accuracy of the classifiers. To keep these two pieces of

dataset independent, trajectories corresponding to the same aircraft were kept together.

4 A Recap on Neural Network-based Classifiers

In this section we briefly review the literature of neural network-based classifiers for time series classification, focusing on work that may be relevant to our problem, and then describe the basic blocks that will compose our classifiers.

4.1 Literature review

Regarding time series classification algorithms, state-of-the-art solutions involve Nearest Neighbors algorithms based on Dynamic Time Warping or its variations [17–19] as a relevant alternative to the Euclidean distance for time series classification. However the execution time for this kind of algorithm often proves too long for real-time classification. This is why in the present work we advocate instead the use of Deep Neural Network based algorithms. One challenging aspect our classification problem owed to the real-time requirement is that the classifier must accommodate trajectories whose length is varying with time. In terms of neural networks, different network architectures may apply. The simplest approaches involve regular fully connected neural layers as proposed by Wang *et al.* [12], but these solutions require a fixed length input, which is an impediment for our problem. Another solution is to use recurrent neural networks, such as the Time Warping Invariant Echo State Network (TWIESN) proposed by Tanisaro *et al.* [20], but most solutions rely on convolutional neural networks [21, 22]. These networks use small kernels that act as filters to extract local information directly from time series. It is also possible to manually augment the data by adding some preprocessing, like in the Multi-scale Convolutional Neural Network (MCNN) proposed by Cui *et al.* [23]. After the convolutions, a global pooling, usually by maximum or average, can aggregate the information through time to a fixed-size vector which can be used by a fully connected neural network for classification [12, 24]: this allows the classifier to work without any requirement on the length of the trajectories. A recent alternative to the global pooling layer is to use an attention layer [25].

4.2 Useful blocks

Having narrowed down our search for relevant classifiers, we now present the basic blocks that will be used to build them.

4.2.1 Convolutional Neural Networks

The Convolutional Neural Network (CNN) was originally designed for image classification and inspired by the organisation of biological neurons in the visual cortex. The idea is to apply a discrete 2D-convolution between an image and a trainable small kernel. Similarly, discrete 1D-convolutions can be applied

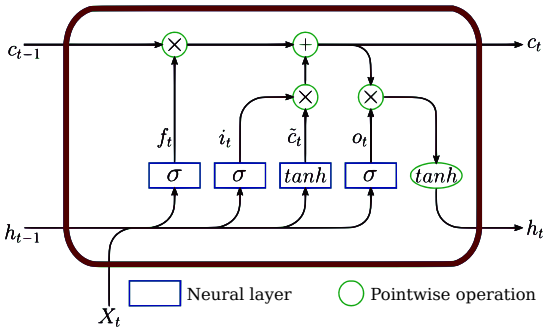

Fig. 2 Block diagram of the LSTM cell, blue rectangles represent trainable neural layers and green circles represent pointwise operation. X_t is the input sequence at time t and h_t the output sequence at time t.

to time series. A filter is then defined by its kernel, bias and activation function. Several filters can be stacked together to extract various features from the same multi-channel input. The transformation made by a convolutional filter can be written as

$$Y_i = f(\sum_{k=0}^{n_f-1} (W_{i,k} * X_k) + B_i),$$

where $*$ denotes the convolution operation, Y_i is the output of the i-th filter of the CNN, n_f is the number of stacked features in the input X, X_k is its k-th feature, W and B are the kernel and bias of the convolutional layer, and f is the non-linear activation function.

4.2.2 Recurrent Neural Networks

Recurrent Neural Networks (RNN) were designed to process sequences. The idea is to build a small neural network that will take the concatenation of the values of the input sequence at a given position t and its own previous output at position $t-1$. The most common RNN cell currently is the Long-Short Term Memory (LSTM [26]), the block diagram of this particular cell can be found in Figure 2. Its specificity is an additional memory called c in the block diagram. It consists of four different neural layers, the first three are made to erase and write the memory, while the last one expresses the output from this memory. The operations made by the cells can be written:

$$\begin{aligned}
f_t &= \sigma(W_f.[X_t, h_{t-1}] + B_f) \\
i_t &= \sigma(W_i.[X_t, h_{t-1}] + B_i) \\
\tilde{c}_t &= tanh(W_{\tilde{c}}.[X_t, h_{t-1}] + B_{\tilde{c}}) \\
o_t &= \sigma(W_o.[X_t, h_{t-1}] + B_o) \\
c_t &= f_t \times c_{t-1} + i_t \times \tilde{c}_t \\
h_t &= tanh(o_t \times c_t),
\end{aligned}$$

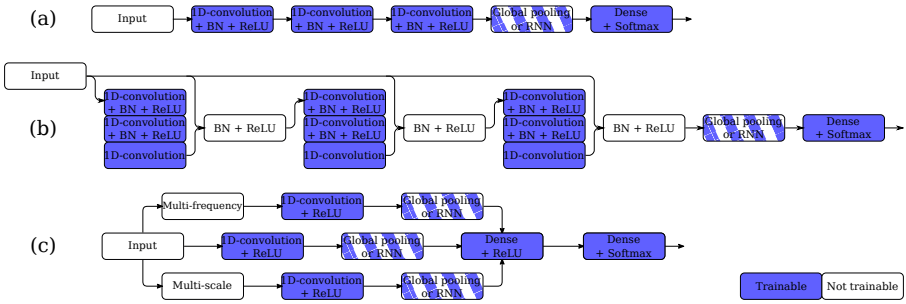

Fig. 3 Block diagrams of the different classifiers, FCN (a), ResNet (b) and MCNN (c).

where . denotes the matrix dot product, $\times$ the pointwise multiplication and [] the concatenation operation, c_t, h_t and X_t are respectively the memory, the output and the input of the cell at time-step t, Ws and Bs are the weights and biases of the cell. A RNN's output can be either a new sequence if we consider all of the outputs, or a fixed-size vector if we only keep the last one.

5 Proposed Classifiers

Here we describe the proposed three neural networks trained to solve our classification problem. These architectures are represented by block diagrams in Figure 3. Recently, Fawaz *et al.* [13] compared different classifiers on many time-series classification problems. We chose to retain Fully Convolutional Neural Networks (FCN) and Residual Networks (ResNet) proposed by Wang *et al.* [12] since they were shown to rank first for most of the datasets. We also tested the Multi-scale Convolutional Neural Network (MCNN) of Cui *et al.* [23].

5.1 Fully-Convolutional Neural Network

The FCN, represented in Figure 3 (a) is simple and straightforward. It involves three convolutional layers with Batch Normalization (BN) and Rectified Linear Unit (ReLU) activation function. BN [27] is commonly used to speed up the training process and sometimes to avoid overfitting. Then a pooling/RNN layer is used to aggregate the time series to a fixed-size vector, which is then fed to a fully-connected neural layer with a $softmax$ activation function. We can use either a Global Max Pooling (GlobalMaxPool) or a LSTM for the aggregation, the difference being that the LSTM is able to forget, unlike the GlobalMaxPool, for which a high value output by the last convolutional block will affect the future outputs. Using a $softmax$ activation function at its end, our classifier will output for each input a vector which can be interpreted as the probability for the aircraft to belong to each class.

5.2 Residual Network

The ResNet, represented in Figure 3 (b), uses three convolutional blocks made of three convolutional layers with BN and ReLU, similarly to FCN. At each block, right after the third convolution and before the BN, a skip connection is used to concatenate the raw input and the filter output. This architecture was originally designed for image recognition by He *et al.* [28], the skip connections allowing for mitigation of the Degradation problem of deep models. After the three blocks, aggregation and a dense layer with softmax activation are used, just like for the FCN.

5.3 Multi-Scale Convolutional Neural Network

MCNN, represented in Figure 3 (c), includes some preprocessing on the raw input sequence before feeding it to CNNs. The data are augmented with the result of different filters. In the Multi-frequency block, sliding averages of different length are used as low-pass filters with different cutoff frequencies. In the Multi-scale block, average poolings of different lengths are used for the following convolution to work with different time-scales. The output of these filters and the raw input are then fed to convolution layers, and finally to an aggregation layer and two dense layers, one with ReLU and one with softmax activation.

For all of these different algorithms, training means minimizing a given loss function by adjusting the trainable parameters using a given gradient-based algorithm computed by backpropagation. In our case, the loss function to minimize through training is the categorical cross-entropy, the opposite of the logarithm of the probability associated with the real class. The optimization algorithm is called Adadelta [29], which performs well in our experiment and does not require to set the initial learning rate parameter.

6 Experiments and Results

In the present section we present and discuss experimental results.

6.1 Training Parameters

The three different proposed classifiers have, besides the training parameters, a few hyper-parameters that must be tuned manually, namely the choice of the aggregation layer (Max or LSTM), the number of convolution filters per layer, and the size of their kernel, the size of the first dense layer and the scales and cutoff frequencies for the MCNN.

Regarding FCN, the numbers of convolutions in the three layers were set to 32, 64 and 32 and three different combinations of kernel length were tested, one with a long filter in the middle and short filters for the two others, one with increasing length and one with decreasing length.

Table 2 Summary of the different trained classifiers with their number of trainable parameters, their accuracy on the test dataset after training and their accuracy on two reduced problem: classifying the species of the aircrafts (plane or helicopter), and classifying the WTC for planes only.

Architecture	Aggregation	Hyper-parameters (k_1, k_2, k_3)	Parameters	Accuracy on 19 Classes (%)	Accuracy on Species (%)	Accuracy on WTC for planes(%)
FCN	Max Pooling	$5, 20, 5$	53 075	34.81	87.23	68.71
		$1, 9, 20$	60 371	33.93	89.05	69.24
		$20, 9, 1$	25 715	35.61	88.07	67.51
FCN	LSTM	$5, 20, 5$	78 515	32.84	87.16	70.49
		$1, 9, 20$	85 811	30.91	87.82	67.42
		$20, 9, 1$	51 155	32.95	88.91	68.04
ResNet	Max Pooling	$1, 13, 1$	114 995	33.72	**90.00**	67.73
		$1, 5, 9$	114 995	32.74	88.70	70.40
		$9, 5, 1$	98 739	35.23	88.28	67.64
ResNet	LSTM	$1, 13, 1$	85 891	35.65	88.18	70.44
		$1, 5, 9$	85 891	32.39	87.61	68.31
		$9, 5, 1$	64 259	34.00	87.61	69.64
MCNN	Max Pooling	$n_c, n_d, n_f = 15, 50, 6$	26 879	31.16	87.72	65.29
		$n_c, n_d, n_f = 30, 50, 4$	46 739	31.82	87.79	66.58
		$n_c, n_d, n_f = 15, 100, 6$	37 629	32.81	88.21	66.93
MCNN	LSTM	$n_c, n_d, n_f = 15, 50, 4$	25 679	29.79	87.26	66.84
		$n_c, n_d, n_f = 30, 50, 6$	59 939	28.14	86.00	64.80
		$n_c, n_d, n_f = 30, 50, 4$	53 939	29.68	87.37	66.53
MCNN-BN	Max Pooling	$n_c, n_d, n_f = 15, 50, 6$	26 879	**39.86**	88.39	73.82
		$n_c, n_d, n_f = 30, 50, 4$	46 739	35.30	88.11	71.96
		$n_c, n_d, n_f = 15, 100, 6$	37 629	39.02	88.67	**74.13**
MCNN-BN	LSTM	$n_c, n_d, n_f = 15, 50, 4$	25 679	32.53	88.95	72.04
		$n_c, n_d, n_f = 30, 50, 6$	59 939	34.98	89.75	72.89
		$n_c, n_d, n_f = 30, 50, 4$	53 939	35.33	88.84	70.27

Regarding ResNet, the same kind of combinations were tried inside each block. The numbers of convolutions were set to 16 for the layers of the first block, 32 for the second and 64 for the third.

Finally, regarding MCNN, many combinations were tested, only the three best for each of the two aggregation layers are presented in Table 2. The n_c parameter denotes the number of convolution filters per layer, n_d the size of the dense layer with the ReLU activation and n_f the number of scales and frequencies used. The length of all convolution kernels is set to 50.

6.2 Main Results

Table 2 summarizes the tested combinations of hyper-parameters for the different classifiers, along with their number of trainable parameters and the global accuracy on the test dataset after training. We see all classifiers achieve a classification performance into the right category of around one third. The best classifier according to the chosen metric is the ResNet with an LSTM aggregation and most of the kernel size in the middle layer of each block. The accuracy of this classifier is 35.65%. Note this is in line with Fawaz's conclusions [13].

6.3 Discussion

The score of 35.65% correct category assessment may seem poor. However, we believe it is not given the considered problem. Indeed, the granularity of labels displayed in Table 1 is high, and aircrafts belonging to two different classes may very well exhibit indistinguishable trajectories.

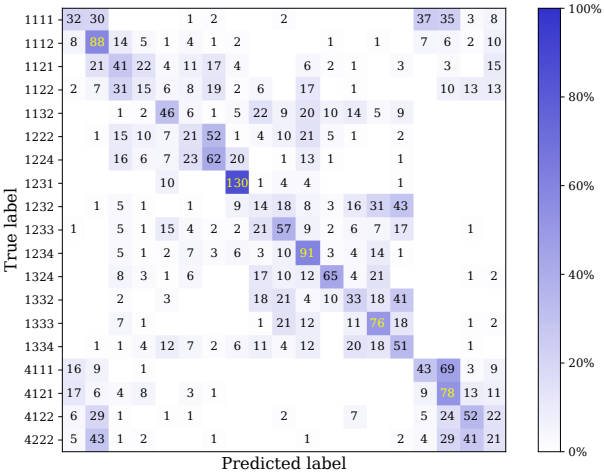

Fig. 4 Confusion matrix of the best classifier on the test dataset. Each box represents the number of aircrafts from the y-axis category classified as the x-axis category. A perfect classifier would have a fully diagonal confusion matrix.

The confusion matrix in Figure 4 gives insight into the type of confusion that is easily made and sometimes inevitable. For instance, the class 1231, usually associated with fighters, is easily recognizable, essentially owing to their distinctively high velocity. As a result a 87% classification accuracy could be achieved for this class. However, small planes with one piston engine (1111) can often be classified as two engines (1112) or small mono-engine helicopter (4111 and 4121) instead, owing to the great similarity (if not indistinguishability) of their kinematics. This is because we trained our classifier based on the ADS-B classes, that do not necessarily match a type of motion category, and hence are transmitted to the air traffic controller to provide it with an information that is in part complementary with the radar trajectory. By using state-of-the-art classifiers and a rich set of kinematic descriptors of the trajectory, we see classification results achieved by the various methods are close, and are likely to represent the best achievable results given the considered problem.

6.4 Coarser Classification Results

In practice, the granularity of ADS-B classification into 19 different classes may be too high for the purposes we pursue. It is then possible to regroup classes or delete some of them by removing the corresponding neurons in the last layer. For instance, one could focus only on the size of the airplane and its capability by trying to detect its WTC. A slightly modified version of the same classifier achieves a 70.44% accuracy in terms of WTC with the confusion matrix represented in Figure 5. This is a much more satisfactory result indeed. The accuracy of all filters on this reduced problem can be found on Table 2 along with the accuracy on the species classification problem.

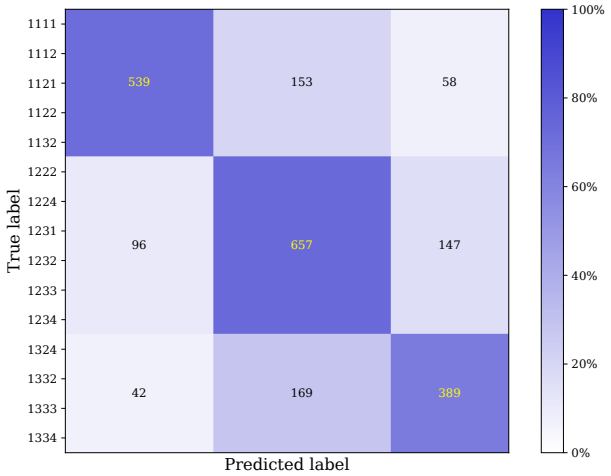

Fig. 5 Confusion matrix of the same classifier as before, adapted to the reduced problem of finding the WTC of a plane.

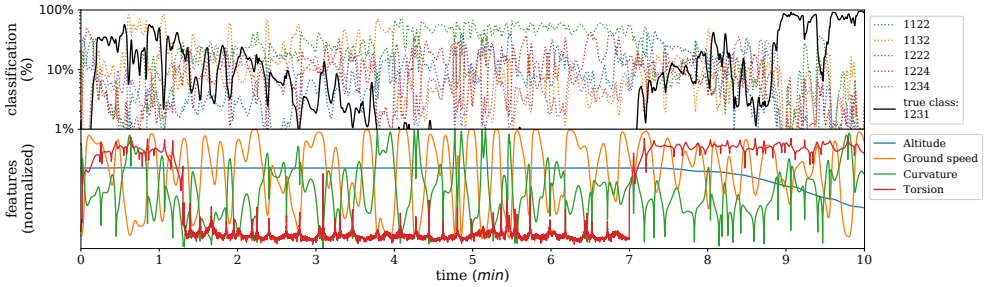

Fig. 6 Top plot: the real-time evolution of the probability assigned to the most likely classes over time on a log scale. Bottom plot: the main features used for this classification normalized and bounded using a tanh function.

6.5 What Does the Network Learn?

To complete our evaluation, let us see how the classifier behaves when confronted with a real-time stream of data as is the case in practice. In this case, it is sufficient to modify our aggregation layer to output the full sequence instead of the last point. This way, the classifier returns for each time the probability for the aircraft to belong to each class based only on past measurements.

We propose to focus on a case study. Figure 6 shows such a sequence for one trajectory. It gives insight into the most relevant features for the classification task, by showing what causes the probability to change most. The classifier used for these plots uses an LSTM as an aggregation layer, which usually leads to fluctuating probabilities. We thus used a Max to make the results more stable and presentable.

We can see the torsion plays a big role in identifying maneuvers (and hence classifying in turn). This is interesting as we see that curvature which is sufficient to discriminate between straight line and maneuvers has a more erratic behavior. This is consistent with torsion having already appeared as a key feature for fighter maneuvers when discussing dynamical models for this kind of military aircrafts in [14], based on twenty-six actual aircraft trajectories made available to the authors of the latter article by their sponsor. It is interesting to note, though, that here this fact is automatically "learned" by the algorithm.

We see the short maneuver during the first minute helps the classifier to correctly identify the class 1231 (medium jet with one engine, in the present case a fighter) up to some interference with the class 1132 (light jet with two engines). Then as it starts flying straight, the belief of the classifier in its previous output drops, as straight lines are not distinctive, if we think of airliners. This means the first maneuver was not long enough to allow for classification with certainty. After seven minutes, though, the aircraft begins its descent, and the probability of the correct class goes up and two minutes later the classifier becomes certain the aircraft is a fighter indeed.

7 Conclusion

Different classifiers were trained to address the rather difficult problem of classifying aircrafts into 19 categories solely based on kinematic features. The architectures of the classifiers and the chosen feature ensure the algorithms are immune to variations in the length of the trajectory and match a set of desirable physical requirements.

After training, the best classifiers obtain an accuracy of about 35% over the test dataset on the complete problem of classification into 19 classes. The classifiers can be adapted to solve reduced problems without the need for additional training by suppressing or merging classes. For instance a classifier can get an accuracy of about 70% on the problem of estimating the Wake Turbulence Category (WTC) of planes, which is encouraging, considering how noisy the dataset is. Indeed WTC is a useful feature to assess the size and the power of the aircraft. Moreover, the assigned probabilities can be monitored through time, this way we can reverse engineer the patterns in a trajectory that lead to a change in the output probabilities and thus find the most discriminating features given a dataset.

Beyond aircraft category assessment, we believe the present paper paves the way for other problems such as detecting aggressive behaviour or any other abnormal activity. In future work, we would also like to test the classifier on real position measurements obtained from a radar. This poses the issue of the extraction in real-time of kinematic parameters such as curvature and torsion. To this end, the features feeding the classifier could be obtained through the recently introduced Kalman filter of [30] that tracks velocity, curvature and torsion in real time.

Acknowledgment

The authors would like to thank THALES LAS and especially Frederic Barbaresco and Philippe Chopin, but also the Mathematics Foundation Jacques Hadamard (PGMO project) for their involvement in this project.

References

[1] Zyweck, A., Bogner, R.E.: Radar target classification of commercial aircraft. IEEE Transactions on Aerospace and Electronic systems **32**(2), 598–606 (1996)

[2] Yan, W., Zhu, Z., Hu, R.: A hybrid genetic/bp algorithm and its application for radar target classification. In: Proceedings of the IEEE 1997 National Aerospace and Electronics Conference. NAECON 1997, vol. 2, pp. 981–984 (1997). IEEE

[3] Lei, J., Lu, C.: Target classification based on micro-doppler signatures. In: IEEE International Radar Conference, 2005., pp. 179–183 (2005). IEEE

[4] Molchanov, P., Egiazarian, K., Astola, J., Totsky, A., Leshchenko, S., Jarabo-Amores, M.P.: Classification of aircraft using micro-doppler bicoherence-based features. IEEE Transactions on Aerospace and Electronic systems **50**(2), 1455–1467 (2014)

[5] Pierucci, L., Bocchi, L.: Improvements of radar clutter classification in air traffic control environment. In: IEEE International Symposium on Signal Processing and Information Technology, pp. 721–724 (2007). IEEE

[6] Thiagarajan, J.J., Ramamurthy, K.N., Knee, P., Spanias, A., Berisha, V.: Sparse representations for automatic target classification in sar images. In: 4th International Symposium on Communications, Control and Signal Processing (ISCCSP), pp. 1–4 (2010). IEEE

[7] Chen, S., Wang, H., Xu, F., Jin, Y.-Q.: Target classification using the deep convolutional networks for sar images. IEEE Transactions on Geoscience and Remote Sensing **54**(8), 4806–4817 (2016)

[8] Nasrabadi, N.M.: Deeptarget: An automatic target recognition using deep convolutional neural networks. IEEE Transactions on Aerospace and Electronic Systems **55**(6), 2687–2697 (2019)

[9] Ghadaki, H., Dizaji, R.: Target track classification for airport surveillance radar (asr). In: IEEE Conference on Radar, p. 4 (2006). IEEE

[10] Dizaji, R., Ghadaki, H.: Classification System for Radar and Sonar Applications. 7567203B2, 2006

[11] Mohajerin, N., Histon, J., Dizaji, R., Waslander, S.L.: Feature extraction and radar track classification for detecting uavs in civillian airspace. In: 2014 IEEE Radar Conference, pp. 0674–0679 (2014). IEEE

[12] Wang, Z., Yan, W., Oates, T.: Time series classification from scratch with deep neural networks: A strong baseline. In: 2017 International Joint Conference on Neural Networks (IJCNN), pp. 1578–1585 (2017). IEEE

[13] Fawaz, H.I., Forestier, G., Weber, J., Idoumghar, L., Muller, P.-A.: Deep learning for time series classification: a review. Data Mining and Knowledge Discovery **33**(4), 917–963 (2019)

[14] Nabaa, N., Bishop, R.H.: Validation and comparison of coordinated turn aircraft maneuver models. IEEE Transactions on aerospace and electronic systems **36**(1), 250–259 (2000)

[15] Chopin, P., Barbaresco, F., Jouaber, S.: Device for Identifying a Type of Aircraft, Associated Identification Method and Computer Program. EP 3 722 830 A1, 2020-10-14

[16] ADS-B Exchange platform. www.adsbexchange.com. Accessed: 2021

[17] Jeong, Y.-S., Jeong, M.K., Omitaomu, O.A.: Weighted dynamic time warping for time series classification. Pattern recognition **44**(9), 2231–2240 (2011)

[18] Ratanamahatana, C.A., Keogh, E.: Making time-series classification more accurate using learned constraints. In: Proceedings of the 2004 SIAM International Conference on Data Mining, pp. 11–22 (2004). SIAM

[19] Xi, X., Keogh, E., Shelton, C., Wei, L., Ratanamahatana, C.A.: Fast time series classification using numerosity reduction. In: Proceedings of the 23rd International Conference on Machine Learning, pp. 1033–1040 (2006)

[20] Tanisaro, P., Heidemann, G.: Time series classification using time warping invariant echo state networks. In: 15th IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 831–836 (2016). IEEE

[21] Le Guennec, A., Malinowski, S., Tavenard, R.: Data augmentation for time series classification using convolutional neural networks. In: ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data (2016)

[22] Zhao, B., Lu, H., Chen, S., Liu, J., Wu, D.: Convolutional neural networks for time series classification. Journal of Systems Engineering and Electronics **28**(1), 162–169 (2017)

[23] Cui, Z., Chen, W., Chen, Y.: Multi-scale convolutional neural networks for time series classification. arXiv preprint arXiv:1603.06995 (2016)

[24] Zheng, Y., Liu, Q., Chen, E., Ge, Y., Zhao, J.L.: Time series classification using multi-channels deep convolutional neural networks. In: International Conference on Web-Age Information Management, pp. 298–310 (2014). Springer

[25] Serrà, J., Pascual, S., Karatzoglou, A.: Towards a universal neural network encoder for time series. In: CCIA, pp. 120–129 (2018)

[26] Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)

[27] Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning, pp. 448–456 (2015). PMLR

[28] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)

[29] Zeiler, M.D.: Adadelta: an adaptive learning rate method. arXiv preprint arXiv:1212.5701 (2012)

[30] Pilté, M., Bonnabel, S., Barbaresco, F.: Tracking the Frenet-Serret frame associated to a highly maneuvering target in 3D. In: 2017 IEEE 56th Annual Conference on Decision and Control (CDC), pp. 1969–1974 (2017). IEEE

Ethics Declaration

Competing Interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding The research leading to these results received funding from Mathematics Foundation Jacques Hadamard (PGMO project) and the French Directorate General of Armaments (DGA).

Author Contributions Conceptualization, S.J., S.V-F, S.B. and M.P.;
Data curation, S.J.;
Funding acquisition, S.V-F, S.B. and J.A;
Methodology, S.J., S.B., S.V-F., and J.A.;
Writing - original draft preparation, S.J.;
Writing – review and editing, S.J., S.V-F. and S.B.;
Resources, S.V-F., M.P.;
Supervision, S.V-F., M.P., S.B.;
Visualization, S.J.;

All authors have read and agreed to the published version of the manuscript.
Data Availability The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.