Air traffic classification using convolutional neural networks

Adrien Marque, Daniel Delahaye, Pierre Maréchal, Isabelle Berry

Air traffic classification using convolutional neural networks.

Adrien Marque,[1)†] Daniel Delahaye,[2)], Pierre Maréchal[3)] and Isabelle Berry[1)]

[1)]University Toulouse 3 Paul Sabatier, CNRS, CerCo UMR5549, Toulouse, France
[2)]*OPTIM Team, French Civil Aviation University, France*
[3)] University Toulouse 3 Paul Sabatier, IMT Toulouse, France

[†]*email: adrien.romain.marque@orange.fr*

The difficulty of managing airspace is reflected in the complexity of forecasting its evolution. This paper presents a new neural network framework for managing images for which pixels are matrices with application to air traffic complexity map prediction. By modelling air traffic with a linear dynamical system, air traffic maps can be defined as images whose pixels are matrices. By computing intermediate steps, these air traffic maps are defined as images whose pixels are symmetric positive definite matrices. Then, we implement a convolution neural network with a specific data preprocessing step, new convolution, max-pooling, and flatten layers suitable to such images. The new convolution, max-pooling and flatten layers are capable of processing images coming from the data preprocessing step.

Key Words: air traffic management, linear dynamical system, complexity, convolution neural network..

1. Introduction

There are currently around 8,720 flights per day registered in France. This traffic creates a considerable workload for air traffic controllers. Thus, the airspace is divided into several elementary sectors which are managed by controllers. An increase in air traffic leads to congestion in these control sectors. Two strategies to alleviate this congestion are based on sector capacity and determined by the number of aircraft passing through in a given period.

On the one hand, controllers can accept aircraft above the capacity threshold. On the other hand, they can reject traffic before maximum capacity is exceeded. This simple fact shows that the current capacity definition is not enough to reflect the workload induced by a traffic situation. The workload is linked to the level of organisation of the trajectories in sectors. If such trajectories are organised like in a flow, the associated complexity is low. On the other hand, if trajectories have a mixing behaviour, then conflicts may appear, and the complexity increases.

The key idea of this approach is to model traffic in the airspace by a dynamical system. This dynamical system associates a velocity vector to each point of the airspace, creating a vector field that gathers the main feature of the traffic. At each point of the airspace, one can associate a level of complexity, which is summarised by a matrix. An airspace then can be represented as a 3D image defined by Eq.(1)

$$A: \{1,\dots,p\} \times \{1,\dots,n\} \times \{1,\dots,m\} \to \mathbb{R}^3. \quad (1)$$

where p, n, and m represent the number discretization steps for the altitude, the longitude, and the latitude, respectively.

Each $A(k,i,j)$ corresponds to the velocity vectors. We can build a complexity map L as defined by Eq.(2)

$$L: \{1,\dots,p\} \times \{1\dots,n\} \times \{1,\dots,m\} \to \mathbb{R}^{3\times 3}. \quad (2)$$

The Fig.1 is an example of a complexity map. Our model assigns a label to each air traffic map L, which measures the associated level of traffic complexity. A detailed explanation of this assignment label can be found in Section 2.

Since air traffic maps are similar to 3D images, the problem of air traffic organisation can be framed as a classification problem. Air traffic maps defined by Eq.(2) can be defined as 3D images whose pixels are real matrices. In the fields of machine learning and statistics, convolutional neural networks (CNN) have proven to be the most efficient algorithms for classifying real images. However, the classification of air traffic maps T, as defined by Eq.(2), poses a unique challenge as they do not conform to conventional images.

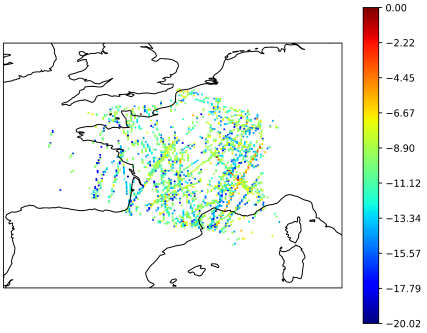

*Fig 1: Example of a complexity map of French airspace over an entire day. All flight levels are represented.*The represented values are computed by the f function (Eq.(8)).

Since there are no neural networks capable of classifying air traffic maps, we transform the air traffic maps into new air traffic maps defined by equation (3)

$$T: \{1,\dots,p\} \times \{1\dots,n\} \times \{1,\dots,m\} \to S_3^+. \quad (3)$$

where S_3^+ is the symmetric positive definite set.

Since S_3^+ is a Riemannian manifold, we have developed a neural network inspired by CNN that is designed to classify these air traffic maps. The neural network's architecture includes a data preprocessing step where the natural logarithm of each $T(k,i,j)$ is computed, and a new convolution, subsampling, and flattening layers suitable for air traffic maps are detailed in Section 3.

The paper is organised into three sections. Section 2 is dedicated to air traffic modelling and label assignment.

Specifically, air traffic maps are generated using a linear dynamical system. Section 3 introduces the state-of-the-art CNN and our neural network to classify air traffic maps. T. We implement convolution, max-pooling, and flatten layers tailored to air traffic maps. Section 4 presents our airspace database and the results obtained by our convolutional neural network model.

2. Linear dynamical system to modelize air traffic map

The airspaces are defined by Eq.(1). Each $A(k,i,j)$ is a velocity aircraft vector. Using a linear dynamical system, we transform Eq.(1) to Eq.(2). Each $M(k,i,j)$ of the air traffic map is called a *complexity matrix*, which informs on local aircraft trajectories complexity.

2.1 State-of-the-art

The "Free Flight" operational concept has generated significant research interest in ATC complexity. Dynamic density was an integral part of Free Flight.

Wyndemere Inc. [1)] proposed a measure of the perceived complexity of an air traffic situation. This measure is linked to the cognitive workload of the controller, with or without knowledge of the aircraft's intentions. The metric is human-oriented and, therefore, very subjective.

Laudeman et al. from NASA [2)] developed a metric called *dynamic density* which is more quantitative than the previous ones.

A study aimed at identifying complexity drivers using judgement analysis identified *airspace design* as the second most important factor behind traffic volume[3)]. The impact of the structure on the controller workload can be found in the paper [4,5)].

But, all previous metrics only capture a single characteristic. In this paper, we are interested in a model capable of capturing as many possible situations as possible. Linear and nonlinear dynamical systems were thus implemented, each presenting its classes of metrics identifying the disorder of a traffic model. The linear dynamical systems are based on geometric properties and propose new metrics capable of extracting characteristics on the complexity of traffic such as proximity, convergence and sensitivity. The linear dynamical systems are based on dynamic air traffic modelling and use topological entropy to measure air traffic disorder. Topological entropy (Kolmogorov entropy) is the only metric capable of capturing most characteristics of complexity. The nonlinear form can even identify possible organisational trajectories (planes following the same path at the same speed). Since the computation time of nonlinear dynamical systems is much longer than that of dynamic linear systems and the accuracy is therefore not sufficient, we are naturally turned to dynamic linear systems.

2.2. Principle

In this approach, we model a set of aircraft trajectories using a linear dynamical system: $\vec{Y} = A \times \vec{X} + \vec{B}$. This equation associates a velocity vector $\vec{Y}$ with a position in space represented by coordinate $\vec{X}$. The congestion matrix A describes the linear mapping between the position vector $\vec{X}$ and the velocity vector $\vec{Y}$. In contrast, the vector $\vec{B}$ to the mean value of the vector field. The eigenvalues of matrix A represents the system's evolution. Specifically, the real part of these eigenvalues determines whether the system is in contraction mode or in expansion mode: a positive real part indicates expansion, while a negative real part indicates contraction.

The imaginary part of the eigenvalues corresponds to the curl tendency of the vector field. level. Depending on these eigenvalues, a dynamical system can evolve through contraction, expansion, rotation, or a combination of these modes. The Fig.2 illustrates four typical examples for which the matrix A and the associated eigenvalues are computed. When the relative distances between aircraft remain unchanged over time, the real parts of the eigenvalues of matrix A are zero. It is illustrated in Fig.2 by situations 1 and 4. When the norms of the relative distances between aircraft decrease over time, the real parts of the eigenvalues are negative. It is illustrated in Fig.2 by situation 2. Conversely, when those relative distances increase over time, the real parts of the eigenvalues are positive. It is illustrated in Fig.2 by situation 3.

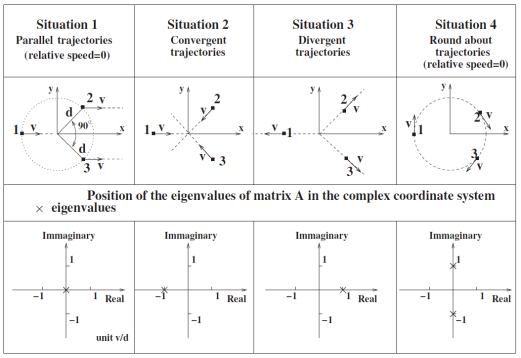

Fig 2: Eigenvalues loci for several typical situations.The small squares are the initial positions of aircraft at a given time [6)].

2.2. The airspace database

We consider a set of aircraft trajectories in a given airspace. For each point of such airspace, we consider a neighbourhood where trajectory samples are extracted. For such aircraft velocity vectors, we compute the associated linear dynamical system thanks to a Least Mean Square approach in order to compute the A matrix and the $\vec{B}$ vector.

$$Y_i = A.X_i + B. \qquad (4)$$

where Y_i is the observed aircraft velocity vectors at position X_i.

The dynamical system has to be adjusted with the minimum error based on the aircraft observations (positions and velocity vectors). This fitting has been done with a Least Mean Square minimization (LMS) method. For each considered aircraft i, it is supposed that position and velocity vectors are given by : $\vec{X}_i = [x_i, y_i, z_i]^T$ $\vec{V}_i = [v_{x_i}, v_{y_i}, v_{z_i}]^T$. An example of such observations is given on Fig.3 for which aircraft observations are represented by the blue arrows.

We then construct an error criterion E, between the

dynamical system model and the observation, based on a norm (Euclidean, in our case), which should be minimised in relation to matrix A and vector $\vec{B}$ and therefore in relation to matrix C, which represents the parameters of the model:

$$E = \sqrt{\sum_{i=1}^{N} ||\vec{V}_i - A.\vec{X}_i||^2}. \quad (5)$$

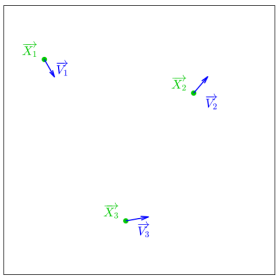

Fig 3. Radar captures associated with three aircraft. In this example only one time sample is considered.

In order to use matrix forms, we rewrite Eq.(4) as $V = C \times X$, introducing the following matrices:

$$X = \begin{pmatrix} x_1 & \cdots & x_N \\ y_1 & \cdots & y_N \\ z_1 & \cdots & z_N \\ 1 & \cdots & 1 \end{pmatrix}, V = \begin{pmatrix} v_{x1} & \cdots & v_{xN} \\ v_{y1} & \cdots & v_{yN} \\ v_{z1} & \cdots & v_{zN} \end{pmatrix},$$

$$\text{and } C = \begin{pmatrix} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ a_{31} & a_{32} & a_{33} & b_3 \end{pmatrix}$$

where $X \in \mathbb{R}^{4\times N}$, $V \in \mathbb{R}^{3\times N}$, $C \in \mathbb{R}^{3\times 4}$, $A \in \mathbb{R}^{3\times 3}$, $B \in \mathbb{R}^{3\times N}$ and N represents the number of observations at a given instant (number of aircraft present in a sector at a given instant). The Eq.(5) can be written as Eq.(6)

$$E = \sqrt{\sum_{i=1}^{N} ||\vec{V}_i - (C.\vec{X}_i)||^2}. \quad (6)$$

In a matrix form $E = ||V - C.X||$.
The derivative of such expression with respect to C is given by Eq.(7)

$$\nabla_C E^2 = -2.(V - C.X).X^T. \quad (7)$$

E is minimum when :

$$\nabla_C E = 0 \Rightarrow C.X.X^T = V.X^T.$$

Then $C_{opt} = V.X^T.(X.X^T)^{-1}$. On the right side, we recognize the pseudo-inverse of matrix X^T :

$$X^+ = X^T.(X.X^T)^{-1}.$$

In some situations, $X.X^T$ is not invertible, and the computation of the C_{opt} is not possible by using such equation. In this case, the classical SVD decomposition trick is applied:

$$X^T.(X.X^T)^{-1} = L^T.S^{-1}.R$$

where S is a diagonal matrix containing the singular values (only the significant singular values are inverted in this formula in order to control the conditioning of the algorithm).

$$\Rightarrow C_{opt} = V.L^T.S^{-1}.R$$

Based on C the matrix A is extracted for which an eigenvalue decomposition is computed :

$$A = B.D.B^T.$$

The diagonal matrix D contains the eigenvalues. When such eigenvalues have positive real parts, the system is in expansion mode, and when they are negative, the system is in contraction mode. We obtain the air traffic map defined by Eq.(2)

$$L : \{1,...,p\} \times \{1 ...,n\} \times \{1, ...,m\} \rightarrow \mathbb{R}^{3\times 3}. \quad (7)$$

where $L(k,i,j)$ is A matrix.
For more information on linear dynamical systems to model air traffic maps and eigenvalues impact, readers are invited to consult [6,7].

2.3. The assignment labels

Airspace maps need to be labelled in terms of complexity to feed our neural network. To achieve this goal, we associate each point in the airspace with the negative real part of the eigenvalue of the A matrix (positive values represent the expansion of the vector field and will not generate conflicts). Then, for each point, we compute the following function f Eq.(8):

$$f(M(k,i,j)) = \sum_{i=1}^{3} \log(-\lambda_i) \quad (8)$$

where $M \in \mathbb{R}^{3\times 3}$ and λ_i negative eigenvalues of the M matrix.
The function f, as shown in Fig.4 on the French airspace, is a representation of the complexity which has to be managed by air traffic controllers.

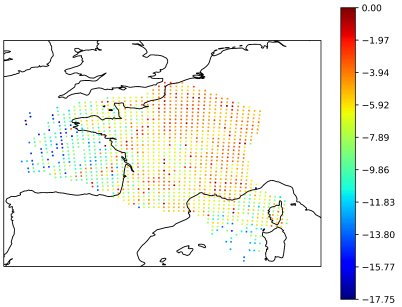

Fig 4. Complexity map of the French airspace. The represented values are computed by the f function (Eq.(8)).

The *global complexity* associated with an air traffic map L is determined using the function c:

$$c(L) = \sum_{k=1}^{p}\sum_{i=1}^{n}\sum_{j=1}^{m} C(k,i,j) = \sum_{k=1}^{p}\sum_{i=1}^{n}\sum_{j=1}^{m} f(L(k,i,j)).$$

Such global complexity is evaluated on each air traffic map composing the database and is then categorised into four

groups. These groups are established based on the quartile distribution of all c values computed on the air traffic map database. Ultimately, we assign a mark from 1 to 4 to label the map, reflecting the difficulty of managing airspace for controllers. A mark of 1 indicates that air traffic management presents no difficulty for controllers in managing the airspace. A mark of 4 indicates that air traffic management represents a critical difficulty for controllers.

The function c, a key component in our process, is then applied to the air traffic map at times t and $t + 1$. It allows us to understand the evolution of the global complexity between times t and $t + 1$ by computing the difference θ:

$$\theta = c(L_{t+1}) - c(L_t)$$

where L_{t+1} and L_t are air traffic maps at the time t+1 and t respectively.

When the difference is negative, the global complexity associated with an air traffic map at time t is more significant than at time $t + 1$. The evolution of air traffic maps between times t and $t + 1$ is more manageable for air traffic controllers. Air traffic maps with a negative difference are associated with a mark of zero.

When the difference is positive, the global complexity associated with an air traffic map at time $t + 1$ is more significant than at time t. Consequently, the evolution of air traffic maps will become more complex for air traffic controllers to manage. The difference values are divided into four groups defined by the distribution of quartiles across the entire air traffic maps database. Ultimately, we assign a mark from 1 to 4 to label these maps, reflecting the evolution difficulty of managing air traffic maps for controllers.

Therefore, an air traffic map is associated with these two marks. Combining these two marks, we obtain a new score with two pieces of information on airspace management: the difficulty of controllers managing airspace at time t and its evolution managing difficulty at time t+1. The new score evolves between a score between 1 and 20. For example, a score of 12 corresponds to the combination of the score of 2 obtained by the c function as well as the score of 4 obtained by the difference function θ. This score of 12 means air traffic mapping is relatively manageable at time t and will become complex at time $t + 1$.

3. Neural network for air traffic maps classification

Our neural network will have to solve classification problems. Air traffic maps are similar to 3D images whose pixels are the complexity matrices. We will first present a state-of-the-art neural network capable of classifying conventional images. Secondly, we will present our neural network, inspired by traditional CNN, by implementing a data preprocessing step, a new convolution, max-pooling, and flattening layers adapted to air traffic maps.

3.1. State-of-the-art

The visual cortex inspires convolutional neural networks. As illustrated in Fig.5, a convolutional neural network consists of an input layer, hidden layers, and an output layer. These hidden layers are divided into two networks: a feature extraction network and a classifier network.

The feature extraction network comprises numerous convolution and pooling layer pairs. The pooling layer reduces the dimensionality. Feature extraction aims to find the most compact and informative set of features and distinct patterns to enhance the classifier's efficiency.

The classifier network is a succession of fully connected layers. Each layer contains a certain number of cells, also known as neurons. An activation function is associated with each layer. This network aims to classify feature maps derived from the feature extraction network.

CNNs have proven their efficiency in image classification [9], object detection [10], facial recognition [11] and video analysis [12,13].

In 2017, a groundbreaking neural network was implemented based on symmetric positive-definite matrices (SPD), the Riemannian manifold. This neural network preserves the (SPD) structure across layers. It integrates the Riemannian structures into the deep network architectures. A novel backpropagation algorithm sets it apart: it trains the network by exploiting a stochastic gradient descent optimization algorithm on Stiefel manifolds. This specific neural network is applied to emotion recognition, action recognition, face verification[14], and recognition of skeleton-based hand gestures[15].

For conventional images, classification problems for classic images are most of the time defined by a parametric model:

$$N: \Omega \times W \to K$$

where W is a set of free parameters, Ω a set of outcomes into a finite number of subset and K a finite whose elements are the labels.

A loss function E is used to characterise the model's goodness-of-fit. The learning algorithm for the model N involves minimising the loss function based on the set W.

After computing congestion matrices A using a linear dynamical system, we have $\Omega = \mathbb{R}^{3\times 3}$.

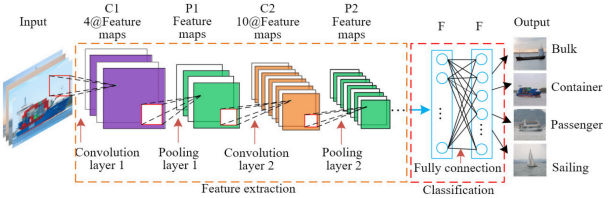

Fig 5: An example of convolution neural network structure [8].

After computing congestion matrices A using a linear dynamical system, we have $\Omega = \mathbb{R}^{3\times 3}$. Therefore, we implement our specific convolutional neural network model. Similar to a conventional neural network, our convolutional model is divided into a feature extraction network and a classifier network.

Moreover, our convolutional model also contains two passes: a forward and a backward pass. A pass is defined as a sequence of mathematical operations dependent on the layers of a neural

network. The forward pass propagates the input data from the first to the last layer, defining the neural network. For each input data, the result is compared to the associated label with a loss function. The loss function provides a result called error denoted by δ. This error is propagated from the last layer to the first layer. Conversely, the backward pass defines the mathematical operations allowing data propagation from the last layer to the first layer.

Since the pixels of air traffic maps are not real numbers, we cannot apply the traditional neural networks described in the state of the art. In this same state of the art, we have also noted the existence of neural networks capable of processing images such as air traffic maps.

3.2. The preprocessing data step

Through the interpolation of the velocity vectors by a linear dynamical system, the input data are transformed into air traffic maps T, precisely defined by Eq.(2). This transformation is defined by Eq.(9):

$$N(k,i,j) = L(k,i,j) \times L(k,i,j)^T, \quad (9)$$

resulting in another air traffic map N, as precisely defined by Eq.(10)

$$N: \{1,\dots,p\} \times \{1\dots,n\} \times \{1,\dots,m\} \to S_3, \quad (10)$$

where S_3 is the 3×3 symmetric matrix set.

The eigenvalues of each pixel are thus positive or null. By employing the singular value decomposition (SVD) method to decompose each $N(k,i,j)$:

$$N(k,i,j) = U_{k,i,j}.D_{k,i,j}.V_{k,i,j}$$

where $D_{k,i,j}$ is the diagonal matrix containing the eigenvalues of $N(k,i,j)$ symmetric matrix.

Then, we add an $\epsilon > 0$:

$$N(k,i,j) = U_{k,i,j}.D_{k,i,j}.V_{k,i,j} + \epsilon \times I_{3\times3},$$

where $I_{3\times3}$ is the 3×3 identity matrix.

This step is significant as it helps to prevent any potential issues with zero eigenvalues, ensuring the stability and accuracy of our data transformation process. This leads us to air traffic maps T defined by Eq.(3)

$$T: \{1,\dots,p\} \times \{1\dots,n\} \times \{1,\dots,m\} \to S_3^+.$$

We implement specific layers inspired by CNN. The convolution operation is the first mathematical operation applied to input data. Since S_3^+ is a Riemannian manifold, convolution operations cannot be applied to air traffic maps T. Since kernels K are allowed to take negative values, producing convolved object $T * K$ whose values are no longer in S_3^+. In order to circumvent this difficulty, we apply convolutions to another air traffic map X where $X(k,i,j) = \log(T(k,i,j))$. Since $T(k,i,j)$ is a symmetric positive definite matrix, it can be diagonalized on an orthonormal basis with strictly positive eigenvalues. The logarithm of the matrix is obtained by applying the natural logarithms to the eigenvalues. Since S_3^+ is a Lie group, $X(k,i,j)$ is a symmetric matrix. The air traffic maps X are then defined by Eq.(11)

$$X: \{1,\dots,p\} \times \{1\dots,n\} \times \{1,\dots,m\} \to S_3 \quad (11)$$

where $X(k,i,j) = \log(T(k,i,j))$.

3.3. The forward pass

The new air traffic map X are the input data of our convolutional neural network model. In the sequel, we denote X as a hidden layer's input, whereas Z is the hidden layer's output.

3.3.1. Convolution layer

The convolution layer is a key part of a convolutional neural network. It performs a mathematical operation called convolution. The data that goes into this layer is filtered by a kernel, which is like a feature detector. It picks out specific features that make up the input images. The outputs of a convolution layer are called feature maps.

Since S_3 is a Euclidean vector space, we adapt the 3D convolution to air traffic maps X. Assume an input X filtered by a real kernel K; the feature maps Z are defined by Eq.(12)

$$Z = \sum_{k=1}^{d}\sum_{i=1}^{r}\sum_{j=1}^{c} X(p-k,n-i,m-j) \times K(k,i,j). \quad (12)$$

The output of convolution layers is forwarded to a subsampling layer. There are mainly three subsampling layers: average-pooling, min-pooling, and max-pooling. In this paper, we focus only on the max-pooling layer.

3.3.2. Max-pooling layer

After the convolution layer, there is mainly a subsampling layer. Subsampling is a technique that reduces reliance on precise positioning within feature maps. A kernel of dimension $d \times r \times c$ is associated with subsampling layers, which divide images into cells. In each cell, the subsampling layer retains just one piece of information related to a mathematical operation.

In this paper, we focus on the max-pooling layer. The feature maps are divided into some cells as in conventional convolutional neural networks. Since each $X(k,i,j)$ is a symmetric matrix, we must redefine the maximum function. Since each $X(k,i,j)$ is a real symmetric matrix, we have $||X(k,i,j)||_2 = \rho(X(k,i,j))$ where ρ is the spectral radius.

During the max-pooling layer, a significant process takes place. The positions where maximum values are located on the input data X are registered on a mask M. This step is crucial in the backward pass of the max-pooling layer.

Once the input data is divided into cells, a crucial step is taken to retain the maximum values for each cell. This is achieved by applying the norm $||.||_2$, a key element in our process that ensures the most significant information is preserved.

3.3.3. Flatten layer

There is a flatten layer after the succession of convolution and max-pooling layers. The practical application of this flattening layer is to vectorize the output of the feature extraction network, making it more accessible for the classifier network. The purpose of our flatten layer is the same as the conventional flatten layer. More specifically, our flatten layer is defined by Eq.(13)

$$Z(t) = X(k,i,j,a,b), \quad (13)$$

where $t = knm + im + j + 3a + b$

where $X(k,i,j,a,b)$ is the real value located at the position (a,b) on the matrix $X(k,i,j)$.

Since the outputs of the flatten layer are $\mathbb{R}^n$ vectors, our classifier network is a conventional artificial neural network (ANN). After forwarding spatial grids into the classifier network, the results are compared to the target using a loss function. Then, the gradient is backpropagated in the CNN. As the classifier network is a conventional ANN, we focus on our flatten, max-pooling, convolution layers' backward pass.

3.4. The backward pass

Once the spatial grids are forwarded into the classifier network, a crucial step in the training process, the results are compared to the target using a loss function. This comparison is a key part of a network's learning process, as it helps us understand the areas where the neural network is performing well and where it needs improvement. The backward pass, a significant part of this learning process, is then performed in the CNN. As the classifier network is a conventional ANN, we focus on our flatten, max-pooling, convolution layers' backward pass. The error is computed using the cross-entropy loss function. This error is then propagated from the last layer to the first layer. Since the classifier network is a conventional ANN, we focus on the flattening, max-pooling, and convolution layers. In the following, we denote the gradient data and the gradient loss as G_{data} and G_{loss}, respectively.

3.4.1. Flatten layer

The backward pass consists of reshaping the gradient coming from the classifier network. The purpose of our backward pass for the flatten layer is the same: to reshape the gradient coming from the classifier network. More specifically, our flatten layer is defined by Eq.(14):

$$G_{data}(k,i,j,a,b) = G_{loss}(t), \quad (14)$$

where $t = knm + im + j + 3a + b$.

3.4.2. The max-pooling layer

Regarding the backward pass, it is also inspired by the conventional max-pooling layer. As we have seen in the forward pass for the max-pooling layer, this layer is associated with a mask M. Eq.(15) defines the gradient data:

$$G_{data}(i,j) = G_{loss}(i,j). \quad (15)$$

If the positions of gradient data are not represented in the mask M, the gradient data takes the 2×2 null matrix as a value. For detailed information on max-pooling layers the readers can refer to [16].

3.4.3. Convolution layer

As for convolution layers, the backward pass can be divided into two steps: data gradient and kernel gradient computation. The data gradient results from the discrete convolution operation between the padding loss gradient and the 180° rotated kernels associated with the convolution layer. The data gradient is based on Eq.(16)

$$G_{data} = p(G_{loss}) \times K_{180^\circ}, \quad (16)$$

where p corresponds to the zero-padding operation.

The kernel gradient is the result of the input convolution layer X filtered by the loss gradient. For detailed information on convolution layers the readers can refer to [17].

4. Results

We have access to French airspace data. These data represent air traffic over a day. In total, our database contains 2875 air traffic maps defined by:

$$F : \{1,\dots,52\} \times \{1,\dots,180\} \times \{1,\dots,186\} \to \mathbb{R}^3.$$

Next, we associate labels with each air traffic map as it is detailed in Section 2.

To classify France's air traffic evolution, we implement a specific neural network whose layers are described in Section 3. The neural network consists of one data preprocessing step, one feature extraction network and a classifier network. The input airspaces are transformed into air traffic maps (pink) with the data preprocessing step. The feature extraction network is divided into two blocks. Each block contains two convolution layers and a max-pooling layer. The feature extraction network is illustrated in Fig.6.

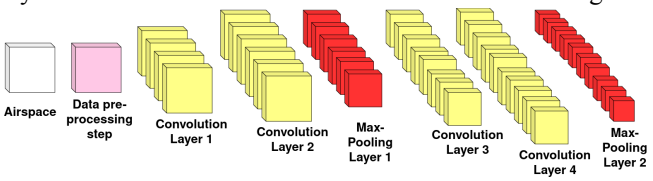

Fig 6. Architecture of our feature extraction network.

The classifier network consists of three fully-connected layers which contain 2835, 567, and 20 neurons. We apply the sigmoid function on the first two fully-connected layers (green). The sigmoïd function is also applied on the last fully-connected layer (blue). The error δ is computed with the cross-entropy loss function. The classifier network is illustrated in Fig.7.

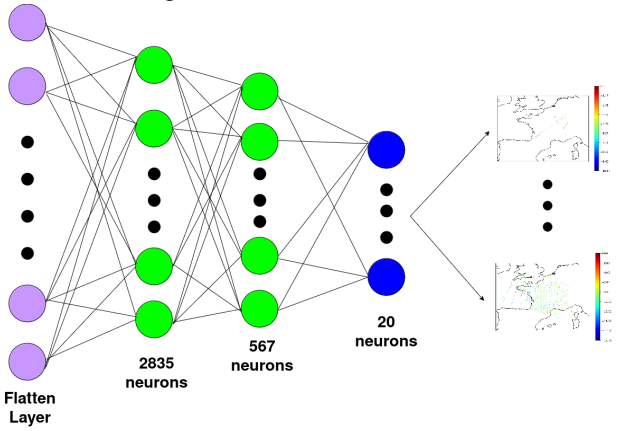

Fig 7. Architecture of our classifier network.

The hyperparameters, such as the training batch size, number of epochs, and the learning rate for the neural network during the training phase, are set to 50, 30, and 0.05.

Since we have only 2875 air traffic maps, we use a 5-cross

validation method. When evaluating different settings ("hyperparameters") there is a risk of overfitting on the test set because the parameters can be tweaked until the estimator performs optimally. Solutions to this problem are procedures called cross-validation and L1 regularisation. We divide the database into a training set and a validation set. In the approach, called k-fold cross-validation, the training set is split into k smaller sets. Then, we repeat k times the following procedure: a model is trained using $k-1$ of the folds as training data and the resulting model is validated on the remaining part of the data (i.e., it is used as a test set to compute a performance measure such as accuracy).

The resulting model is validated on the remaining part of the data: the validation set. The performance measure reported by k-fold cross-validation is then the average of the values computed in the loop. The 5-fold validation is illustrated in Fig.8. For more information on the k-fold cross-validation method is detailed in [18].

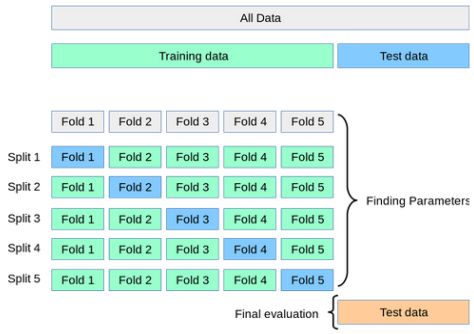

Fig 8: Illustration of 5-fold cross-validation method [19].

The database, including air traffic maps defined by Eq.(3), is then divided into the training set and the validation set. The training set contains 2300 air traffic maps (80% of air traffic maps including in the database) whereas the testing set contains 575 air traffic maps (20% of air traffic maps including in the database). The training set is divided into 5 folds including 460 air traffic maps. For each step, 1840 air traffic maps train the neural network whereas the remaining 460 air traffic maps are dedicated to the testing step. The neural network is validated on the 575 air traffic maps included in the validation set.

The neural network correctly classifies an average of 65.22% of France's air traffic map, improving to 91.31% after 20 epochs. The training step is illustrated in Fig.8.

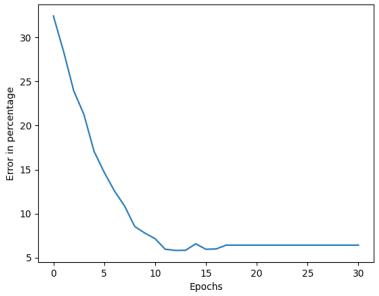

Fig 8: Evolution of the error percentage with epochs number.

During the testing step, our CNN correctly classified an average of 90% of France's air traffic. Although the input data differs, this neural network's results are similar to those obtained by random forest [20] and support vector machine (SVM) [21] algorithms. Figs.9 and.10 illustrate our neural network results. Fig 9 illustrates an airspace with a note equals to 2 for controllers to manage air traffic at time t. Fig.10 illustrates also an airspace that is complex for controllers to manage at time $t+1$. At time $t+1$, the evolution of air traffic is given a score of 4, indicating the difficulty of managing it at time $t+1$.

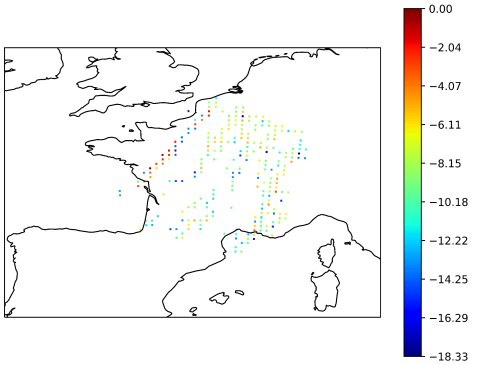

Fig 9: Our neural network predicts a note equals to 2 for air traffic management at time t. The represented values are computed by the f function (Eq.(8)).

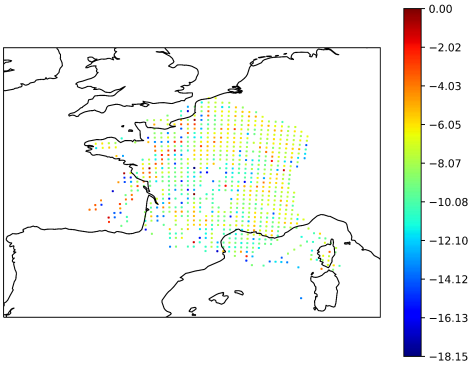

Fig 10: Our neural network predicts a note equals to 4 for evolution to air traffic management at time $t+1$. The represented values are computed by the f function (Eq.(8)).

The neural network can identify the global complexity level of a given airspace. On the other hand, the neural network can predict the evolution of this air complexity at a time $t+1$. Combining this information could significantly support controllers' decision-making. Indeed, providing the neural network with an air traffic situation could inform air traffic controllers about its potential critical development in air traffic. In other words, in the context where the neural network detects a conflicting evolution of air traffic, it could support air traffic controllers by providing them with a danger rating on the evolution of the airspace. This potential holds great promise for the future of air traffic management.

5. Conclusion

In this paper, we implement a neural network capable of predicting the evolution of air traffic complexity. The specificity of this neural network is based on the implementation of data preprocessing, new convolution, new max-pooling, and flatten layers able to manipulate

images composed of matrix pixels. We apply this neural network to an air traffic database.

Each air traffic map is assigned a label indicating its future evolution. Our neural network is trained using the robust 5-fold cross-validation method. Post-training, the neural network achieves an average classification rate of 91. 31% for air traffic maps. In the testing phase, the neural network has a classification rate of 88%.

The neural network's ability to detect patterns in air traffic maps is a significant breakthrough. The neural network can therefore identify the global complexity of an air traffic map and its future evolution. Ultimately, this neural network could be used as a valuable tool for air traffic controllers, providing them with insights into the likely evolution of air traffic and helping them in their decision-making process. However, the neural network is expensive in computation time and resources requiring improvement. In addition, the neural network needs to be trained on a more extensive database of French air traffic.

This opens the way for developing tools to help air traffic controllers make decisions in critical situations that can be complex to manage.

Acknowledgments

The authors would like to acknowledge Lavandier Julien for his assistance in realising this project.

This work was supported by the convention CIFRE 2019/1343, the company Siemens Healthineers and CerCo CNRS.

References

1) Windemere inc, An evaluation of air traffic control complexity, Tech. report, NASA 2-14284, 1996.

2) Laudeman, I. V., Shelden, S. G., Branstrom, R., & Brasil, C. (1998). Dynamic density: An air traffic management metric [Techreport].

3) Kirwan, B., Scaife, R., & Kennedy, R. (2017). Investigating complexity factors in UK air traffic management. In Engineering Psychology and Cognitive Ergonomics (pp. 189–195). Routledge.

4) Histon, J. M., Hansman, R. J., Aigoin, G., Delahaye, D., & Puechmorel, S. (2002). Introducing structural considerations into complexity metrics. Air Traffic Control Quarterly, 10(2), 115–130.

5) Histon, J. M., Hansman, R. J., Gottlieb, B., Kleinwaks, H., Yenson, S., Delahaye, D., & Puechmorel, S. (2002). Structural considerations and cognitive complexity in air traffic control. Proceedings. The 21st Digital Avionics Systems Conference, 1, 1C2-1C2.

6) Delahaye, D., & Puechmorel, S. (2010). Air traffic complexity based on dynamical systems. 49th IEEE Conference on Decision and Control (CDC), 2069–2074.

7) Delahaye, D., García, A., Lavandier, J., Chaimatanan, S., & Soler, M. (2022). Air traffic complexity map based on linear dynamical systems. Aerospace, 9(5), 230.

8) Ren, Y., Yang, J., Zhang, Q., & Guo, Z. (2019). Multi-feature fusion with convolutional neural network for ship classification in optical images. Applied Sciences, 9(20), 4209.

9) Chen, L., Li, S., Bai, Q., Yang, J., Jiang, S., & Miao, Y. (2021). Review of image classification algorithms based on convolutional neural networks. Remote Sensing, 13(22), 4712.

10) Sultana, F., Sufian, A., & Dutta, P. (2020). A review of object detection models based on convolutional neural network. Intelligent Computing: Image Processing Based Applications, 1–16.

11) Hassan, R. J., & Abdulazeez, A. M. (2021). Deep learning convolutional neural network for face recognition: A review. International Journal of Science and Business, 5(2), 114–127.

12) Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., & Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1725–1732.

13) Simonyan, K., & Zisserman, A. (2014). Two-stream convolutional networks for action recognition in videos. ArXiv Preprint ArXiv:1406.2199.

14) Teuwen, J., & Moriakov, N. (2020). Chapter 20 - Convolutional neural networks. In S. K. Zhou, D. Rueckert, & G. Fichtinger (Eds.), *Handbook of Medical Image Computing and Computer Assisted Intervention* (pp. 481–501). Academic Press. https://doi.org/https://doi.org/10.1016/B978-0-12-816176-0.00025-9

15) Huang, Z., & Van Gool, L. (2017). A riemannian network for spd matrix learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, *31*(1).

16) Nguyen, X. S., Brun, L., Lezoray, O., & Bougleux, S. (2019, June). A Neural Network Based on SPD Manifold Learning for Skeleton-Based Hand Gesture Recognition. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

17) Gholamalinezhad, H., & Khosravi, H. (2020). Pooling methods in deep neural networks, a review. ArXiv Preprint ArXiv:2009.07485.

18) Anguita, D., Ghelardoni, L., Ghio, A., Oneto, L., Ridella, S., & others. (2012). The'K'in K-fold Cross Validation. *ESANN*, *102*, 441–446.

19) Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research 12:2825–2830 https://scikit-learn.org/stable/modules/cross_validation.html

20) Moreno, F. P., Rodríguez, F. I., Comendador, V. F. G., Jurado, R. D.-A., Suárez, M. Z., & Valdés, R. M. A. (2024). Prediction of air traffic complexity through a dynamic complexity indicator and machine learning models. *Journal of Air Transport Management*, *119*, 102632.

21) Zhang, H., Jiang, C., & Yang, L. (2016). Forecasting traffic congestion status in terminal areas based on support vector machine. *Advances in Mechanical Engineering*, *8*(9), 1687814016667384.