Detection of precursors of combustion instability using convolutional recurrent neural networks

Antony Cellier, Corentin J. Lapeyre, Gorkem Oztarlik, Thierry Poinsot, Thierry Schuller, Laurent Selle

Detection of Precursors of Combustion Instability using Convolutional Recurrent Neural Networks

A. Cellier[a,*], C.J. Lapeyre[a], G. Öztarlik[b], T. Poinsot[b], T. Schuller[b], L. Selle[b]

[a] *CERFACS, 42 avenue Gaspard Coriolis, 31057 Toulouse, France*
[b] *Institut de Mécanique des Fluides de Toulouse, IMFT, Université de Toulouse, CNRS, Toulouse, France*

Abstract

Many combustors are prone to Thermoacoustic Instabilities (TAI). Being able to avoid TAI is mandatory to efficiently operate a system without sacrificing neither performance nor safety. Based on Deep Learning techniques, and more specifically Convolutional Recurrent Neural Networks (CRNN) [1], this study presents a tool able to detect and translate precursors of TAI in a swirled combustor for different fuel injection strategies. The tool is trained to use only time-series of unsteady sensors in stable conditions to predict the proximity of unstable operating points on a mass flow rate / equivalence ratio operating map, offering a real-time information on the margin of the system versus TAI. This allows to change operating conditions, and detect the directions to avoid in order to remain in the stable domain.

Keywords:
Thermoacoustic Instability, Instability Precursors, Deep Learning, Convolutional Recurrent Neural Networks

1. Introduction

The way towards leaner and fuel flexible combustion is paved with critical issues [1, 2]. Thermoacoustic Instabilities (TAI) threaten the integrity of the system and the safety of its user. Their prediction is crucial to further improve combustion devices. This work highlights a step toward the implementation of a Deep Learning tool for the prognosis of instability in a ducted lean multi-fuel swirled injector. Based on a real-time analysis of combustion noise, it predicts the proximity of unstable operating points in a bulk velocity and equivalence ratio diagram.

The framework of this task is a well known field of research in thermoacoustics known as "Precursor Detection". Several groups have highlighted that precursors of TAI are observed as the system gets closer to bifurcation [3–6]. However identifying a robust definition of precursors remains difficult. As pointed out by Juniper and Sujith [7], strategies to extract precursors for experimental systems are still challenging to set but robust stable signal analysis for instability prediction should be found in the near future. In fact, the existence of tangible information in combustion noise pressure has been discussed by Lieuwen [8] on a laboratory-scale combustor. An *et al.* [9] observed that pressure signals can warn about impending instability in a turbine engine. Rouwenhorst *et al.* [10] successfully produced an online strategy to obtain stability margins based on modal analysis of time-series for an annular combustor. Murayama *et al.* [11] and Gotoda *et al.* [12] have also shown that specific dynamic behaviors can be detected in noise, preceding a critical event such as lean blow-off, and proposed methods to identify precursors for such events. Yet, finding the right precursors to focus on for one specific device can be a thorny problem. Coupling Machine Learning techniques to pre-existing methods eases the identification and detection of precursors.

Sarkar *et al.* [13] have implemented real-time diagnostics based on Deep Learning applied to symbolic time series analysis of high-speed camera imaging. Furthermore, Kobayashi *et. al.* [14] highlighted then the possibility to extract meaningful information from pressure and heat release rate measurements for critical event prediction. Evidences of

[1] Source code and data for the deep learning in this study is available at https://gitlab.com/cerfacs/code-for-papers/2021/precursors-instability

[*] Corresponding author
Email address: cellier@cerfacs.fr (A. Cellier)

the benefits of Machine Learning techniques are also observed when affordable sensors are used. Sengupta *et al.* [15], through the analysis of a single pressure sensor, obtained insights on the stability margins of the system. Finally, McCartney *et al.* could couple Detrended Fluctuation Analysis to Machine Learning and outperformed former methods on the detection of precursors of instability from dynamic pressure measurements [16].

These results suggest that the time signals of pressure, velocity and heat release rate can contain sufficient information to predict that the instability is close, prior to bifurcation, and that a machine learning strategy can leverage data to automatically extract it. An additional step toward a user-centered precursor detection system is taken here by proposing a Convolutional Recurrent Neural Network (CRNN). The objective of this study is to use real-time pressure, velocity and heat release rate signals of a combustor to be able to change its operating conditions, here the equivalence ratio ϕ and total mass flow rate, controlled by the bulk velocity U_{sw}, without ever entering a domain of combustion instability. The tool must have been trained to recognize if a stable operating point is close in parameters (ϕ, U_{sw}) to unstable operating points, and predict in which directions the unstable points exist, offering the possibility to avoid them.

A concrete application of this tool could be explained as follows. Maps of operating conditions (U_{sw}, ϕ), tested *a priori*, could be used to know which regions are unstable and to be avoided. However, these maps can be subject to modifications due to changes in fuel, atmospheric conditions, hardware wears, ... It is thus interesting to use real time analysis of the combustor activity to predict whether combustion instability is close or not, without relying on initial maps anymore. The practical challenge of this study is that two operating parameters are studied simultaneously (ϕ, U_{sw}), implying that precursors of instability should have different signatures regarding a notion of direction on this bi-parameter plane. In addition, different injection strategies are tested in order to enhance the diversity of the dataset. In this context, Deep Learning, a sub-field of Machine Learning, can be a precious ally as it gives the possibility to analyze high-dimensional data such as high sampling rate time signals. Data-based methods applied to physics are currently in development in many fields, following progress of software and hardware. Accessible tools for Deep Learning [17] open the scope of possibilities. This study investigates the ability of neural networks to recognize instability precursors from time signals with good confidence and high accuracy for various fuel injection strategies, making a real-time evaluation of the stability of the burner possible.

First a description of the experimental setup and dataset is given. The learning task is then exposed with explanations about the learning model, its training and use. Finally, comments on the interpretability of the approach are presented, along with a discussion about choices made during the implementation of the method.

2. Experimental setup

The MIRADAS experimental setup used in this study is presented in Fig. 1 [18]. The instrumented system is composed of a swirled injector confined in a cylindrical quartz combustion chamber with an internal diameter of 46 mm and a length of 297 mm, opened at the top-end. The main mixture is injected in the lower part of the plenum of circular cross-section with a diameter of 65 mm, and homogenized by a honeycomb panel. The flow passes through a convergent and a swirler with feed channels oriented 15° off the swirler axis. Flame piloting is offered by the central injection tube dedicated to pure fuels. In this study, the configurations involve methane (CH_4) and hydrogen (H_2). Methane, hydrogen and air mass flow rates are controlled separately (Brooks SLA 585x series), making this system a modular burner for the study of multi-fuel/multi-injection control strategies.

Combustion stability is diagnosed with one Brüel & Kjær-TYPE 4954 microphone for the measurement of acoustic pressure, located at the exit section of the convergent (M1). It is coupled with a constant temperature hot-wire (Dantec 55P16 miniature hot wire probe and a Dantec 54T42 MiniCTA) for velocity measurement (HW). The unsteady heat release rate is deduced from the light emission intensity recorded by a THORLABS FB430-10 photo-multiplier and a narrow band-pass filter centered on the emission spectrum peak of CH^* radicals (PM).

3. The experimental dataset

The stability of the system under different operating conditions, controlled by the bulk velocity U_{sw} and the equivalence ratio ϕ, has been assessed in [18] where seven different fuel injection configurations have been tested, for premixed and piloted injections of methane (CH_4) and hydrogen (H_2). The fuel injection configurations are defined

in Tab. 1 and represented in Fig. 2. The influence of each injection strategy is measured in percentages of power and presented in Tab. 1. Most of the power is provided by premixed CH_4/air combustion.

For each fuel injection configuration, the bulk velocity U_{sw} is varied from 14 m.s^{-1} to 40 m.s^{-1} with a 2 m.s^{-1} step, and the equivalence ratio ϕ is varied from 0.70 to 0.85 with a 0.05 step. A maximum of 56 points per fuel injection configuration are acquired at a sampling frequency of 10 kHz for a 10.3 s duration. For several configurations, operating points at high bulk velocity and high equivalence ratio were not acquired as strong TAI threatened the integrity of the test bench. Furthermore, at low bulk velocity or low equivalence ratio, blow-off or flash-back phenomena were observed. This explains the variations in number of operating points acquired N_{op} in Tab. 1.

4. A simple definition of the proximity of combustion instability

4.1. Stability Analysis

The database is composed of 366 different operating points, indexed by the corresponding injection configuration, the bulk velocity U_{sw} and the equivalence ratio ϕ. For each operating point, the Root Mean Square (RMS) value of the acoustic pressure recorded by the microphone M1 is computed. Unstable operating points display high pressure oscillations, a main characteristic of the limit cycle during TAI [19]. The empirical limit of 250 Pa (141.9 dB) of RMS is observed to be the threshold of TAI for this combustor. Operating points with RMS pressure fluctuations levels above the threshold are labeled unstable. Among the 366 operating points obtained, 187 are evaluated stable, *i.e.* with a RMS pressure level lower than 250 Pa (141.9 dB).

4.2. Specificity of the database

In the MIRADAS burner, TAI does not monotonically increase with the bulk velocity and the equivalence ratio. For example in the Ref configuration in Tab. 1, the Root Mean Square (RMS) values of the acoustic pressure recorded by the microphone M1 are represented in Fig. 3, as a (U_{sw}, ϕ) map. Darker regions are subject to strong instability. It is thus possible to see stable regions framed by unstable regions in the way that it is difficult to properly define a simple rule associating a bulk velocity and an equivalence ratio to the stability of the combustor. Furthermore, when considering a different injection configuration where 1 % of the power is produced by the injection of H_2 in the pilot lance (Tab. 1, PH1), regions of instability are modified in Fig. 4 and cannot be associated precisely to a given bulk velocity and equivalence ratio. It is impossible to predict the position of unstable regions with respect to the operating parameters (U_{sw}, ϕ) without running unstable.

The following paragraphs present an approach based on the sole exploration of stable operating points in order to get information on the position of these unstable operating regions.

4.3. Label attribution

In a first step, a label is attributed to each stable operating point of the operating map. This index contains information on surrounding unstable points on a (U_{sw}, ϕ) plane. The label contains 4 binary digits, 1 digit per direction, indicating whether this direction points toward instability or not. An example of label attribution is given in Fig. 5. The boundary conditions for this attribution are treated such that a stable operating point located at a border is given a label 0 in the direction of outside the map. Examples are shown in Fig. 5 for $U_{sw} \in [14, 18]$ m.s^{-1}, $\phi = 0.7$ or $U_{sw} = 14$ m.s^{-1}, $\phi \in [0.7, 0.75]$. This procedure is done manually and produces a map of labels for the entire operating region. This label is the simplest indicator of the proximity of unstable points on a bi-parameter plane.

4.4. Definition of the objective

In order to build the labelling presented here, it is necessary to operate the experimental bench at every operating point, including unstable ones. The practical objective of this study is to see if it is possible to predict this labelling without ever entering unstable zones. Every stable operating point is associated with two sorts of information which are its position on a bi-parameter plane (U_{sw}, ϕ), and the signal acquired by the sensors (M1, HW, PM).

The following study offers a comparison between two different approaches to predict the label. First of all, a baseline is built to predict the label from the treatment of the position (U_{sw}, ϕ) of a stable operating point. Then, an analysis of the three-channels signal using Deep Learning techniques is performed on stable operating points in order to predict the label.

5. Presentation of the learning environment

5.1. State prediction in the machine learning context

Predicting the proximity of a system to instability can be translated to a Machine Learning task by defining the process to learn the association of an input X to a corresponding output y such that $X \in \mathbb{R}^{N_T \times N_C}$ is a sample of time signal provided by the sensors, for a stable operating condition, where N_T is the number of points in the sampling window and N_C is the number of channels considered, and y is a label translating the proximity of unstable operating points.

A supervised learning of this task is a two step procedure. First, the training dataset composed of samples of signal X^i and their corresponding output label y^i known exactly is built. It contains N_s samples $(X^i, y^i)_{i\in[\![1,N_s]\!]}$. Let f such that $\forall i \in [\![1, N_s]\!], y^i = f(X^i)$. A function $\hat{f}_\theta$ is searched under the form of optimizable models in order to approximate f, where θ is the set of parameters of the model. A metric called the *loss function* is used to assess how close $\hat{f}_\theta$ approximates f, and the process of optimizing $\hat{f}_\theta$ to minimize the resulting *loss function* on the dataset $(X^i, y^i)_{i\in[\![1,N_s]\!]}$ is called *training the model*. At the end of the training process, the best trained model defined by $\hat{f}^*_\theta$, and its parameters θ^* is frozen and saved [20]. It can then be used to associate new input samples X' to predicted labels $\hat{y}'$, where the exact label y' is not known *a priori*, because the operating point has not been tested before and its proximity to instability is thus unknown. This second phase is separated from the training, and called the *inference*. The predicted label is computed as $\hat{y}' = \hat{f}^*_\theta(X')$ which approximates $y' = f(X')$.

The learning environment can be summarized as follows. The signal $(X_i)_{i\in[\![1,N_s]\!]}$ composes the Input of the model, and $(y_i)_{i\in[\![1,N_s]\!]}$ represents the corresponding Output that the model is expected to produce. The optimizable Model $\hat{f}$ is searched in order to link efficiently the Input to the Output for further use. Eventually, a Baseline is defined in order to assert the performance of the Model. It is thus necessary to define the Input, the Output, the Model and its Baseline.

5.2. Choice of Input and corresponding Output

The 10.3 s long signal acquired at a given operating point is sampled by randomly cropping 300 windows of 0.3 s. This strategy is preferred to a classical regularly spaced sampling procedure to avoid injecting spurious patterns based on the choice of number of windows extracted. The choice of the duration 0.3 s of the sample has been influenced by two factors, the necessity to have a sufficient number of samples for the training in order to reach convergence, and the will to capture enough periods of oscillation given that the TAI frequency of the system always lies close to 590 Hz. The samples are built with three channels provided by the three sensors mounted on the experimental setup:

1. The *acoustic pressure* M1 upstream the swirler
2. The *acoustic flux* M1.HW, which corresponds to the product of the unsteady pressure and the hot wire velocity at the same location
3. The *unsteady heat release rate* PM

Each channel of the sample is normalized by its own maximum absolute value such that the signal lies in $[-1, 1]$. It insures that the most obvious traces of the bulk velocity and the equivalence ratio are impossible to retrieve in the signal. Besides, it is usual practice to feed a learning model with normalized signals as amplitude can vary from a configuration to another making generalization harder. One sample of an operating point is attributed the label corresponding to this operating point. The Input is thus a three-channels 0.3 s time signal and the Output is a four binary digit label. After sampling, the dataset contains 56100 3-channels signal samples with their corresponding labels, extracted from the 187 stable operating points.

5.3. An adequate Model: the CRNN

This work aims to classify multi-channel time-series. The classification of sound-like time-series is a recurrent topic in Deep Learning. A naive approach is to use a Multi-Layer Perceptron (MLP) [21]. Built with Fully Connected layers, its architecture is simple and easy to set-up, but limitations emerge when considering high-dimensional series where coherent patterns should be aimed at. A better architecture is thus obtained by enhancing the MLP with convolution layers in order to build a Convolutional Neural Network (CNN) [22]. Based on filtering techniques, a CNN learns an optimal set of filters in the form of convolutional kernels. Finally, a potential improvement of the efficiency of common CNN for time-series is obtained by the addition of a recurrent block [23]. The Long Short Term Memory (LSTM)

layer is chosen for this study for its good performance in collaboration with CNN [24, 25]. The CNN generally focuses on learning patterns at different scales. The LSTM layer, when put after the convolutional blocks, learns the time dependence between the patterns [26].

The Convolutional Recurrent Neural Network (CRNN) is composed of four stacked convolutional blocks. Each block is made of a 1-Dimensional convolution layer followed by a Batch Normalization [27], a Rectified Linear Unit (ReLU [28]) and a Max-Pooling layer. The four stacked blocks are connected to the LSTM layer. The output of the recurrent layer is finally associated to fully connected layers used to interpret the patterns and their dependence, learnt by the precedent layers and to output the predicted label. The task to perform is more specifically a *multi-class-multi-label* classification, *multi-class* as there are four different directions to assert and *multi-label* as one sample can be part of more than one class when several directions are simultaneously pointing toward instability. Therefore the output activation function is chosen to be four sigmoid functions, one for each direction, thus yielding four scores laying in $]0, 1[$. Each score is interpreted as a predicted probability that the concerned direction points toward TAI. The network used in this study is represented in Fig. 6 and detailed specifications are given in Tab. 2.

5.4. Definition of the Baseline

Before deploying the Deep Learning signal analysis, a simpler approach has to be implemented. Comparing the performance of this simple baseline strategy to the complete analysis gives the opportunity to emphasize both weaknesses and forces of the learning approach. The Baseline is built as follows. The dataset is composed of 187 stable operating points. Each operating point is associated with a couple (U_{sw}, ϕ). The objective of the baseline is to associate this couple to the label attributed to the operating point. This association relies on four Logistic Regressions (LR) taken from the Scikit-Learn Python library [29]. Each regression is expected to predict one direction out of the four. The LR consists in optimizing $\mathbf{w} = (w_1, w_2)$ and w_0 such that:

$$\begin{aligned} \hat{y}^i &= \hat{f}(w_0, \mathbf{w}, \mathbf{X}^i) \\ &= \text{sigmoid}(w_0 + (\mathbf{X}^i)^T \mathbf{w}) \\ &= \text{sigmoid}(w_0 + w_1 \overline{U}^i_{sw} + w_2 \overline{\phi}^i) \end{aligned} \tag{1}$$

where $\hat{y}^i$ is the predicted probability that the considered direction of the i^{th} sample is unstable, $\overline{U}^i_{sw} = U^i_{sw}/U^{max}_{sw}$ is the normalized bulk velocity of the i^{th} sample, $\overline{\phi}^i = \phi^i/\phi^{max}$ is the normalized equivalence ratio of the i^{th} sample, and $\mathbf{X}^i$ is the vector $(\overline{U}^i_{sw}, \overline{\phi}^i)$. The sigmoid function completes the model by ensuring that the score $\hat{y}^i$ lies in $]0, 1[$ with a strong dichotomy between scores close to 0 and scores close to 1 in order to obtain a clearer idea of the stability of the considered direction of the i^{th} sample. The weights $\mathbf{w}$ and w_0 are obtained following the minimization of the L2-regularized logistic cost function:

$$J = C \sum_{i=1}^{N_s} \log\left(\exp\left(-\left(2y^i - 1\right)\left((\mathbf{X}^i)^T \mathbf{w} + w_0\right)\right) + 1\right) + \frac{1}{2}\mathbf{w}^T \mathbf{w} \tag{2}$$

where C is a user defined constant and y^i is the true label. In this expression, $y^i = 1$ if the considered direction of the i^{th} sample is unstable, $y^i = 0$ if it is not. The minimization is performed by the L-BFGS-B algorithm [30].

This strategy only relies on the analysis of the correlation between the two operating parameters and the output labels. For example, if one operating point defined by its bulk velocity and equivalence ratio has the same label for every injection configurations, the Baseline will already perform perfectly on this operating point. The signal analysis is then useless. However if there are operating points on which the signal analysis outperforms the baseline, this would ensure that information is extracted from the signal, and could not be leveraged by the simpler Baseline. Figure 7 summarizes the learning task and the Baseline.

6. Training the CRNN

6.1. Splitting the dataset

In order to verify that the training is successful, the dataset is split into three distinct groups:

1. *training set:* The label attribution is learnt on the samples from the training set.
2. *validation set:* It is built to evaluate the error during the training on data that are not used for the training and therefore not seen by the model. It allows to monitor how well the model generalizes on unseen data through the epochs of training. Hyperparameters of the training process - how many epochs to train, parameters of the optimization algorithm, details of the network architecture - can be chosen optimally with respect to this generalization task, instead of optimizing only for the training loss and risking overfitting.
3. *testing set:* The samples of this set are kept completely unseen during the training. They are used to verify that the label attribution learnt on the training set is rightfully performed on new data.

The 187 operating points are allocated into the three sets so that the distribution of labels fits the natural distribution observed in the entire dataset, thus reducing unbalanced behaviors where only a few different types of labels can be precisely predicted. Figure 8 displays how the distribution into the sets is built. The complete set is composed of 187 stable operating points (accounting for 51600 samples), the training set has 132 stable operating points (39600 samples), the validation set has 27 stable operating points (8100 samples) and the testing set has 28 stable operating points (8400 samples). The evaluation of the Baseline is eventually made following the same dataset splitting such that performances of both methods can be compared after training.

6.2. Loss Function

The measure of the performance of the model through the epochs of training is done by the loss function. For this task, it is chosen to work with the L2-Regularized Binary Cross-Entropy (BCE_{L_2}) [31–33] . It is computed as follows for the four directions:

$$BCE_{L_2} = -\frac{1}{N_s}\sum_{i=1}^{N_s}\left(y^i \log(\hat{y}^i) + (1-y^i)\log(1-\hat{y}^i)\right) + L_2(\theta) \tag{3}$$

where N_s is the total number of samples composing the evaluated set (testing or validation set), $\hat{y}^i$ is the predicted probability that the considered direction of the i^{th} sample is unstable, y^i is the true label. In this expression, $y^i = 1$ if the considered direction of the i^{th} sample is unstable, $y^i = 0$ if it is stable. The operator $L_2(\theta)$ accounts for the L_2 norm of certain optimizable parameters θ of the model. It avoids that the parameters of the model reach too high values and smooths the training. The four BCE_{L_2} obtained for each direction are then averaged to produce a single value after each epoch, and this defines the loss.

This metric evaluates the confidence of the predictions by strongly penalizing overconfident behaviors. For example, a high testing or validation Binary Cross-Entropy [31, 32] in parallel with a high rate of true predictions can be interpreted as the fact that most of the predictions are right but when a prediction is wrong, it is far from the truth.

6.3. Training results

Over the epochs of training, the loss is monitored for both training and validation sets. Figure 9 highlights the fact that an optimum is obtained when the validation loss is minimal after 17 epochs representing a 20 minutes training on a Nvidia Tesla V100 GPU using the Keras framework with TensorFlow as backend [17, 34]. After training, this optimal network, frozen at 17 epochs, is saved for further use, namely predicting the proximity of unstable operating points based on the analysis of samples never seen during the training. Results are then compared to the Baseline.

6.3.1. First performance evaluation: Accuracy measurement

The output of both the model and the baseline is a vector of four values in $[0, 1]$. Each value is interpreted as the probability that the corresponding direction is unstable. The discriminant threshold of 0.5 is applied to these values such that a probability under (resp. over) the threshold informs that the direction concerned is predicted stable (resp. unstable). The accuracy is then computed as follows:

$$a = \frac{1}{N_s} \sum_{i=1}^{N_s} \frac{n^i}{4} \tag{4}$$

where N_s is the number of samples in the set considered and n^i is the number of directions correctly evaluated for the i^{th} sample. For example, if the i^{th} sample should have been attributed {*up*:0—*left*:0—*right*:1—*down*:1} but got attributed {*up*:0—*left*:0—*right*:1—*down*:0}, only three directions are correctly asserted, $n^i/4 = 0.75$. Using the formalism introduced in order to describe the loss function, n^i is written as follows:

$$n^i = \sum_{j=1}^{4} \left(1 - \left(\mathbb{1}_{thr}\left(\hat{y}^i_j\right) - y^i_j\right)\right) \tag{5}$$

where $\hat{y}^i_j$ is the predicted probability that the j^{th} direction of the i^{th} sample is unstable, y^i_j is the true label, and $y^i_j = 1$ if the j^{th} direction of the i^{th} sample is unstable, $y^i_j = 0$ if it is stable. The function $\mathbb{1}_{thr}$ is the indicator function set with the default discriminant threshold of 0.5. Therefore, $\mathbb{1}_{thr}(x) = 1$ if $x \geqslant 0.5$ and $\mathbb{1}_{thr}(x) = 0$ if $x < 0.5$.

The accuracy of the model and its Baseline on both validation and testing sets is given in Tab 3. First of all it is important to observe that the Baseline is already performing better than randomly guessing the directions pointing toward instability, showing that some basic information is already contained into the operating parameters. Then, the very high accuracy reached by the model is a first step toward the conclusion that the model can efficiently extract information from the signal in order to distinguish the directions of instability. However, as the task is multi-class-multi-label, the sole evaluation of the accuracy is not sufficient to conclude on the performance of the model. Actually it could be possible that three out of four directions are perfectly predicted for every samples and the last one would only be poorly detected. Hence, a Receiver Operating Characteristic (ROC) study is conducted [35].

6.3.2. Second performance evaluation: ROC curves

The Receiver Operating Characteristic study is adapted to binary classification. It consists in evaluating both the false positive rate (FPR) and the true positive rate (TPR) given by the model as the discriminant threshold introduced and fixed to 0.5 in the last paragraph is moved. For each value of the threshold, the TPR is plotted versus the FPR, producing a curve bounded by a square of side 1 and linking the point $\{0, 0\}$ to the point $\{1, 1\}$. The rates are obtained on the combined validation and testing datasets, composed of samples not used for training. For example, as the threshold changes, a perfect classifier would not change its results, even if the threshold is moved far from 0.5, *i.e.* close to 0 or close to 1. The corresponding ROC curve would limit a perfect square. The area under the curve (AUC) is then reaching 1. A random classifier produces a straight line between $\{0, 0\}$ and $\{1, 1\}$ with an AUC of 0.5. A good classifier is therefore characterized by a high AUC ranging from 0.8 to 1.0. It is then possible to compare two binary classifiers by observing their respective AUC.

The task performed in this document is not strictly a binary classification. Nevertheless, when each direction is considered separately, the problem becomes a set of four binary classifications, each task being to define the class of the direction: stable (0) or unstable (1). Thus, four ROC curves are obtained, one for each direction, allowing a separated evaluation of the performance of the two methods regarding each direction. Figure 10 plots these curves along with the corresponding AUC. It appears that the directions "UP" (higher equivalence ratio) and "DOWN" (lower equivalence ratio) are well classified by both methods. However, the trained model manifests higher AUC, exceeding 0.9, showing that these directions are more efficiently classified by the model than by the baseline. Results are even clearer regarding the directions "LEFT" (lower bulk velocity) and "RIGHT" (higher bulk velocity). The Baseline laboriously reaches AUCs near 0.6 when the trained model continues to perform around 0.9.

To demonstrate the use of ROC curves to set safety margins, two constraints are imposed. First, the False Negative Rate (FNR) should not exceed 5 %, which corresponds to a True Positive Rate higher than 95 %, then the False Positive Rate (FPR) should be below 20 %. These mimic system constraints such as damaging instabilities that must be avoided,

and the wish to thoroughly explore the stability map, respectively. These two different constraints lead to performances represented in Fig. 11. In the first case, the FPR is contained for every direction, regarding the high expectations in terms of safety. In the second case, TPR are elevated enough for a precise exploration. The ROC study thus offers the opportunity for the user to define its tolerances regarding the specifications of the experimental setup, leading to different accuracy trade-offs.

The ROC study thus confirms that the training of the model went successfully, rendering balanced results for the prediction of instability through the analysis of stable state samples of signal. The model adds the possibility to determine the directions concerned by the instability on a bi-parameter plane. User imposed precision and safety tolerances can then be used to enhance the definition of thresholds for a better detection of the instability.

7. Using the trained CRNN to predict instability

The best trained model obtained in the previous sections is now frozen to predict labels on operating points never seen before. The following paragraphs introduce a possible application for this selected network.

7.1. Real-time prediction

The network is trained to attribute a label on a 0.3 s time sample at an acquisition frequency of 10 kHz. In real use, the experiment can be lit for minutes and sensors deliver information continuously. For real-time prediction, the signal from the sensors must be seen by the network regularly under the exact form of a 0.3 s sample acquired at 10 kHz. In order to approach this configuration, the trained network is successively fed by the content of a rolling window passing on a 10.3 s long signal as illustrated in Fig. 12. At each step of the rolling window, the model produces a label predicting the proximity of unstable operating points, based on its training. This operation is finally averaged over the total acquisition time of 10.3 s, yielding an overall prediction compared to the original label determined by the experiment.

An important parameter of the rolling window process is the delay imposed between each translation step. The minimal delay is theoretically imposed by the acquisition frequency and thus reaches 0.1 ms. However it can be changed in order to fit the practical limitations of the acquisition system. In fact, its first order limitation is the inference delay, which is the time needed for the network and the architecture translating its results to produce one label from the analysis of one sample. The longer the inference delay is, the more spaced the predictions are, impeding a real-time use. On an Intel i5-8210Y CPU dedicated to personal computers, the maximum inference time is 19 ms. Thus a minimum of 19 ms is to be imposed as the refreshing time in-between each prediction. This delay is restrained enough to guarantee a fast response of the analyzing system to sudden changes of the operating parameters of the experimental setup, offering the possibility to use the model as a real-time diagnostic when directly wired to the stream of information obtained by the sensors. It should be noted that modern dedicated architectures for network inference, *i.e.* in embedded systems, can be orders of magnitude faster, thus significantly lowering the limitations in real systems.

7.2. Real use scenario

In order to simulate a real-use case, the rolling-window is set with a 20 ms delay between two successive evaluations. Figure 13 shows the rolling window evaluation of the model on the operating point PC1, $U_{sw} = 30\,\mathrm{m.s^{-1}}$, $\phi = 0.70$, taken from the testing dataset. The four outputs of the network, with respect to the four directions assessed, are plotted in four different graphs. The point considered in Fig. 13 is surrounded by two unstable operating points, the first one in the upper direction ($U_{sw} = 30\,\mathrm{m.s^{-1}}$, $\phi = 0.75$), and the second one on the right ($U_{sw} = 32\,\mathrm{m.s^{-1}}$, $\phi = 0.70$). Therefore, the network should predict constant values of 1 for the "UP" and "RIGHT" directions and constant values of 0 for the "DOWN" and "LEFT" directions. In practice, for this operating point, the network is able to give the right prediction nearly constantly. However, due to the intermittent nature of the signal recorded by the sensors for operating points close to instability, it appears that the network is mislead during short periods concerning the higher bulk velocity direction, *i.e.* "RIGHT" in Fig. 13, leading to the value 0.9. Yet, the tool is still steady enough for the interpretation of the directions pointing toward unstable operating points to remain clear, confirming that fine specific features of the signal have been extracted from the noise and are translated by the CRNN in order to inform the user. The possibility to have an updated prediction every 20 ms gives the opportunity to have a real-time directional information on the proximity of unstable operating points.

8. Influence of essential choices made during implementation

During the implementation of the final approach, steps have been taken regarding two essential elements of the model, the structure of the input and the architecture of the network. In order to observe the effects of these developments, the accuracy is given along with the AUC from the ROC analysis. A third metric is added in order to measure the confidence of the predictions, the Binary Cross-Entropy (BCE) [31, 32], formerly used with regularization as a loss function during the training phase. Here, only the Cross-Entropy term is kept, such that for each of the four directions:

$$BCE = -\frac{1}{N_s}\sum_{i=1}^{N_s} y^i \log(\hat{y}^i) + (1-y^i)\log(1-\hat{y}^i) \tag{6}$$

where N_s is the total number of sample composing the evaluated set (testing or validation set), $\hat{y}^i$ is the predicted probability that the considered direction of the i^{th} sample is unstable, y^i is the true label, and $y^i = 1$ if the considered direction of the i^{th} sample is unstable, $y^i = 0$ if it is not. The four BCE obtained for each direction are then averaged to render the overall Binary Cross-Entropy of the model.

8.1. Choice of the set of sensors

A key step when building the tool is to choose the content of the samples placed at the input of the network. The number of channels is essential to define, provided that the more sensors are available the better it usually is in order to capture the dynamic of a system, and that the invasiveness of diagnostics is currently a crucial limitation for real applications. To briefly study the influence of the choice of sensors on the performance of the predictor, three configurations have been tested. In the first configuration, the only sensor studied is the microphone M1. The prediction is made on single-channel samples. In the second configuration both the photo-multiplier PM and the microphone M1 are considered. The prediction is made on two-channel samples. Finally, the third configuration is the three-channels sample configuration, proposed in previous sections. Table 4 shows the difference in precision appearing when the number of channels is changed.

Both the accuracy and the AUC worsen with nearly constant BCE when the number of channels is reduced. However it is interesting to point out that using only the pressure sensor is already enough to reach 87.6 % of validation accuracy and 84.8 % of testing accuracy with acceptable AUC. Furthermore, when coupled with a photo-multiplier, the prediction is improved to reach a precision close to the three-channels configuration. The three-channels configuration however remains the most efficient one. Nonetheless, it appears that the evaluation of the acoustic pressure alone can be sufficient to obtain a fair prognosis of the stability of the combustor.

This succinct comparison is a first step of an ongoing study of the influence of the choice of sensors on the efficiency of the prediction. The next step is to test every combination of sensors and evaluate its influence on the quality of the result, along with the minimization of the invasiveness of the diagnostics involved.

8.2. Evolution of the architecture of the model

The CRNN is built by stacking layers of different types. In order to observe the benefits of the addition of each type of layer, three networks are evaluated. The first network is a three layer MLP built with the same layers as the three last Fully Connected layers of the final CRNN (See Tab. 2). The second network is a CNN built with four convolution layers stacked over the MLP. Finally, the third network is the final CRNN architecture. Each network is representative of a step of the evolution of the model through the development process. Evaluated on the same dataset, the Binary Cross-Entropy (BCE) loss and the accuracy for both validation and testing sets along with the AUCs are computed and given in Tab. 5.

The addition of convolution layers on top of the MLP enhances its accuracy but has a limited effect on the confidence of the network, as shown by BCE scores. However, the combination of convolution layers with a recurrent LSTM layer both increases the accuracy and the confidence of the model. Therefore, for this specific task, it appears that an efficient architecture is obtained with a CRNN approach.

The path to the final architecture is therefore marked by two main observations. First, the accuracy of a Multi-Layer Perceptron (MLP) when performing this multi-class-multi-label classification task is low. Enhancing the MLP with four convolution layers and thus building a Convolutional Neural Network (CNN) normally helps to aim for patterns in

the signal and, in this case, increases the accuracy and AUC of the model. Then, the CNN displays high validation and testing accuracy when performing the task but tends to do over-confident wrong predictions. Adding the LSTM recurrent layer helps to mitigate this flaw and thus both increases the accuracy and decreases the BCE, leading to better results.

This study does not offer a comprehensive search for the optimal network for the task at hand, and numerous parameters could be explored for each network. The focus here is on the proof of concept. However, further work, notably including more data, could focus on optimizing the neural network, e.g. using Neural Architecture Search [36].

9. Critical analysis

9.1. Interpretability of a neural network

The promising results obtained in this study suggest that meaningful features are present in the signal at the onset of the instability and that these features seem to translate a directional information on a bi-parameter operating plane. However, the use of Deep Learning raises one crucial criticism. Even if the model efficiently associates features to empirical scores, it does not provide the exact characteristics of these features. This is due to the fact that the model is built by an optimization algorithm which is practically impossible to interpret. This can be a problem when the reliability of the tool has to be evaluated, and could limit the applicability of such tools for real-world combustors, however well they perform during training and testing. Progress in these techniques must be accompanied by the development of tools to interpret them in the future.

Supervised Machine Learning has been described as the ability to produce internal representations from the analysis of inputs and their corresponding outputs [20]. Interpreting the model is equivalent to finding a way to explain these representations. Several groups have worked on the interpretability of neural networks through the study of representations [37, 38]. Yet, these remain case-by-case approaches adapted to very specific tasks. In this study, there is no simple intuition to explain what allows convergence based on combustion theory. In fact, it is also precisely one of the supporting reasons for the use of Deep Learning. Its power is revealed when input sizes are large and when the features searched in it are unknown *a priori* by the user.

9.2. The context of data-starvation

Using Deep Learning techniques in an experimental context where the dimensions of the dataset can be strongly limited underlines a recurrent problem of such an approach. The learning algorithm demands a certain quantity of training items which is difficult to evaluate beforehand as it strongly depends on the task. In this study, the apparently low number of stable operating points available is mitigated by the possibility to sample each operating point in overlapping windows. Still, it does not protect from having created hidden trivial structures not seen at first glance but found by the model during training.

However, the construction of a simpler baseline helps to deal with dataset bias. On the one hand, it gives an easy-to-setup approach to solve the problem with performances comparable to more complete approach. On the other hand, it helps to see the hidden structures of the dataset. In fact, if the baseline displays results close to or better than the model, it is an obvious sign that the hidden parameter is revealed. For example in this case, if the baseline (focused on the bulk velocity and the equivalence ratio) performed as well as the model, it would emphasize the fact that these operating parameters are detectable in the signal and sufficient to conclude on the proximity of instability. As it is not clearly the case in this study, the intuition that discriminant patterns compose the signal and could be picked up by the CRNN to give a directional information on the stability of the burner still holds.

Nonetheless, increasing the size of the dataset and finding more robust definitions of the proximity of instability are further objectives to reach, this work being a very first step.

10. Conclusion

The use of a CRNN for the analysis of combustion dynamics shows promising results to change operating conditions without entering an unstable domain. The trained network predicts the proximity of unstable operating points on the bulk velocity-equivalence ratio plane for different configurations by treating short windows of 0.3 s of acoustic pressure, acoustic flux and heat release rate fluctuations recorded during stable states of the system.

The steps to build the learning tool are:

1. *Building a dataset*: The signal from the three sensors is acquired systematically by running different fuel injection configurations under varying operating conditions. The operating points follow a 2-Dimensional grid $[U_{sw}, \phi]$.
2. *Label attribution*: The stable operating points are labelled with respect to their proximity to unstable operating points considering four directions on a (U_{sw}, ϕ) plane.
3. *Sampling*: The labelled stable operating points are sampled in three channels, 0.3 s elements. A sample is therefore the input of the network and its corresponding label is to be predicted by the network.
4. *Splitting the dataset*: The dataset is split into three sets : the training set, the validation set and the testing set.
5. *Building a Baseline*: A baseline is set up in order to further compare after-training results.
6. *Training*: The network is trained to associate stable, three-channel, 0.3 s samples to their corresponding labels. Overfitting is reached as the loss on the validation set starts to increase again through epochs of training. The network is frozen and saved before overfitting.
7. *Testing*: Scores on the testing set, samples that have never been seen by the network during training, reveal the efficiency of the network to generalize what has been learnt during training.
8. *Using the trained network* : Based on a rolling-window evaluation of stable signal, the trained network can be used to predict the proximity of unstable operating points in real-time.

Results show that it is possible to locate unstable regions in the operating map based on records during stable operation and signal treatment from a limited number of sensors, thus avoiding violent instability episodes when operating in unknown conditions. This analysis can be done in real-time, giving a constant help as signal streams from the sensors. Additionally, the CRNN approach appears as an efficient first step towards a data centered tool able to systematically warn the user about impending critical issues.

Several directions must be investigated to improve this initial study. First, it would be necessary to improve the generalization ability of the tool to a wider range of configurations with finer operating grids and output labels containing more information. Evaluating the robustness to transformations of the burner and operating atmospheric conditions is necessary to ensure that the method can learn a general feature about proximity to instability, regardless of the configuration. Therefore, a complete analysis of the resilience of the model to changes in atmospheric conditions and injection configurations is needed. Furthermore, the modularity of the MIRADAS burner allows the transformation of its geometry (swirler, chamber and plenum), a key factor toward an efficient and robust learning system for the prediction of thermo-acoustic instability in different combustion devices. The construction of new metrics to predict, applicable to different combustors and thus less linked to empirical considerations is also required for generalization. In parallel, the influence of changes in the structure of the network and the choice of sensors to analyze should be studied, together with a complete hyperparameter tuning procedure, so that the optimum performances of such an approach can be achieved. Finally, a better interpretation of what the network can unveil from combustion dynamics is needed. It is the key step to both improve the theoretical understanding of precursors of combustion instability and design reliable, and efficient tools for their diagnostic in real-size combustion devices.

Acknowledgements

This project has received funding from the European Research Council under the European Union's Horizon 2020 research and innovation program Grant Agreement 832248, SCIROCCO.

References

[1] S. Taamallah, Z. A. Labry, S. J. Shanbhogue, A. F. Ghoniem, Thermo-acoustic instabilities in lean premixed swirl-stabilized combustion and their link to acoustically coupled and decoupled flame macrostructures, Proceedings of the Combustion Institute 35 (3) (2015) 3273–3282.

[2] T. Lieuwen, H. Torres, C. Johnson, B. Zinn, A mechanism of combustion instability in lean premixed gas turbine combustors, Journal of Engineering for Gas Turbines and Power 123 (1) (2001) 182–189.

[3] V. Nair, G. Thampi, S. Karuppusamy, S. Gopalan, R. I. Sujith, Loss of chaos in combustion noise as a precursor of impending combustion instability, International Journal of Spray and Combustion Dynamics 5 (4) (2013) 273–290.

[4] V. Ramanan, S. R. Chakravarthy, S. Sarkar, A. Ray, Investigation of combustion instability in a swirl-stabilized combustor using symbolic time series analysis, Proceedings of the ASME 2014 Gas Turbine India Conference, GTINDIA2014-8280.

[5] A. Krishnan, R. I. Sujith, N. Marwan, J. Kurths, On the emergence of large clusters of acoustic power sources at the onset of thermoacoustic instability in a turbulent combustor, Journal of Fluid Mechanics 874 (2019) 455–482.

[6] N. B. George, V. R. Unni, M. Raghunathan, R. I. Sujith, Pattern formation during transition from combustion noise to thermoacoustic instability via intermittency, Journal of Fluid Mechanics 849 (2018) 615–644.
[7] M. P. Juniper, R. Sujith, Sensitivity and Nonlinearity of Thermoacoustic Oscillations, Annual Review of Fluid Mechanics 50 (1) (2018) 661–689.
[8] T. Lieuwen, Online combustor stability margin assessment using dynamic pressure data, Journal of Engineering for Gas Turbines and Power 127 (3) (2005) 478–482.
[9] Q. An, A. M. Steinberg, S. Jella, G. Bourque, M. Füri, Early Warning Signs of Imminent Thermoacoustic Oscillations Through Critical Slowing Down, Journal of Engineering for Gas Turbines and Power 141.
[10] D. Rouwenhorst, J. Hermann, W. Polifke, Online Monitoring of Thermoacoustic Eigenmodes in Annular Combustion Systems Based on a State-Space Model, Journal of Engineering for Gas Turbines and Power 139 (2) (2017) 1–8.
[11] S. Murayama, K. Kaku, M. Funatsu, H. Gotoda, Characterization of dynamic behavior of combustion noise and detection of blowout in a laboratory-scale gas-turbine model combustor, Proceedings of the Combustion Institute 37 (4) (2019) 5271–5278.
[12] H. Gotoda, Y. Shinoda, M. Kobayashi, Y. Okuno, S. Tachibana, Detection and control of combustion instability based on the concept of dynamical system theory, Physical Review E - Statistical, Nonlinear, and Soft Matter Physics 89 (2) (2014) 1–8.
[13] S. Sarkar, K. G. Lore, S. Sarkar, V. Ramanan, S. R. Chakravarthy, S. Phoha, A. Ray, Early Detection of Combustion Instability from Hi-speed Flame Images via Deep Learning and Symbolic Time Series Analysis, Annual conference of the prognostics and health management society.
[14] T. Kobayashi, S. Murayama, T. Hachijo, H. Gotoda, Early Detection of Thermoacoustic Combustion Instability Using a Methodology Combining Complex Networks and Machine Learning, Physical Review Applied 11 (6) (2019) 1.
[15] U. Sengupta, C. E. Rasmussen, M. P. Juniper, Bayesian Machine Learning for the Prognosis of Combustion Instabilities from Noise, Proceedings of the ASME 2020 Turbomachinery Technical Conference Exposition, GT2020-14904.
[16] M. McCartney, T. Indlekofer, W. Polifke, Online detection of combustion instabilities using supervised machine learning, Proceedings of the ASME 2020 Turbomachinery Technical Conference Exposition, GT2020-14834
[17] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, J. Yangqing, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems (2015).
[18] G. Oztarlik, L. Selle, T. Poinsot, T. Schuller, Suppression of Instabilities of Swirled Premixed Flames with Minimal Secondary Hydrogen Injection, Comb. and Flames (214) (2020) 266–276.
[19] T. Poinsot, D. Veynante, Theoretical and Numerical Combustion, Third Edition (www.cerfacs.fr/elearning), 2011.
[20] I. Goodfellow, Y. Bengio, A. Courville, Machine Learning Basics, in: Deep Learning, MIT Press.
[21] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning internal representations by error propagation, MIT Press, Cambridge, MA 1 (V) (1986) 318–362.
[22] I. Goodfellow, Y. Bengio, A. Courville, Convolutional Networks, in: Deep Learning, MIT Press.
[23] A. B. Nassif, I. Shahin, I. Attili, M. Azzeh, K. Shaalan, Speech Recognition Using Deep Neural Networks: A Systematic Review, IEEE Access 7 (2019) 19143–19165.
[24] J. Zhao, X. Mao, L. Chen, Speech emotion recognition using deep 1D & 2D CNN LSTM networks, Biomedical Signal Processing and Control 47 (2019) 312–323.
[25] N. Yalta, S. Watanabe, T. Hori, K. Nakadai, T. Ogata, CNN-based Multichannel End-to-End Speech Recognition for Everyday Home Environments, EUSIPCO.
[26] S. Hochreiter, J. Schmidhuber, Long Short-Term Memory, Neural Computation 9 (8) (1997) 1735–1780.
[27] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, 32nd International Conference on Machine Learning, ICML 2015 1 (2015) 448–456. arXiv:1502.03167.
[28] K. Jarrett, K. Kavukcuoglu, M. Ranzato, Y. LeCun, What is the best multi-stage architecture for object recognition?, Proceedings of the IEEE International Conference on Computer Vision (2009) 2146–2153
[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine Learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.
[30] C. Zhu, R. H. Byrd, J. Nocedal, L-BFGS-B: Algorithm 778: L-BFGS-B, FORTRAN routines for large scale bound constrained optimization, Transactions on Mathematical Software 23 (1997) 550 – 560.
[31] T. M. Mitchell, Artificial Neural Networks, in: Machine Learning, McGraw-Hill Science/Engineering/Math, 1997.
[32] I. Goodfellow, Y. Bengio, A. Courville, Deep Feedforward Networks, in: Deep Learning, MIT Press.
[33] I. Goodfellow, Y. Bengio, A. Courville, Regularization for Deep Learning, in: Deep Learning, MIT Press.
[34] F. Chollet, Deep Learning with Python, Manning Publications, 2018.
[35] T. Fawcett, An introduction to ROC analysis, Pattern Recognition Letters 27 (8) (2006) 861–874.
[36] T. Elsken, J. H. Metzen, F. Hutter, Neural architecture search: A survey, arXiv 20 (2019) 1–21. arXiv:1808.05377.
[37] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, L. Kagal, Explaining explanations: An overview of interpretability of machine learning, Proceedings - 2018 IEEE 5th International Conference on Data Science and Advanced Analytics, DSAA 2018 (2019) 80–89arXiv:1806.00069,
[38] V. Buhrmester, D. Münch, M. Arens, Analysis of Explainers of Black Box Deep Neural Networks for Computer Vision: A Survey (Ml) (2019) 1–22. arXiv:1911.12116.

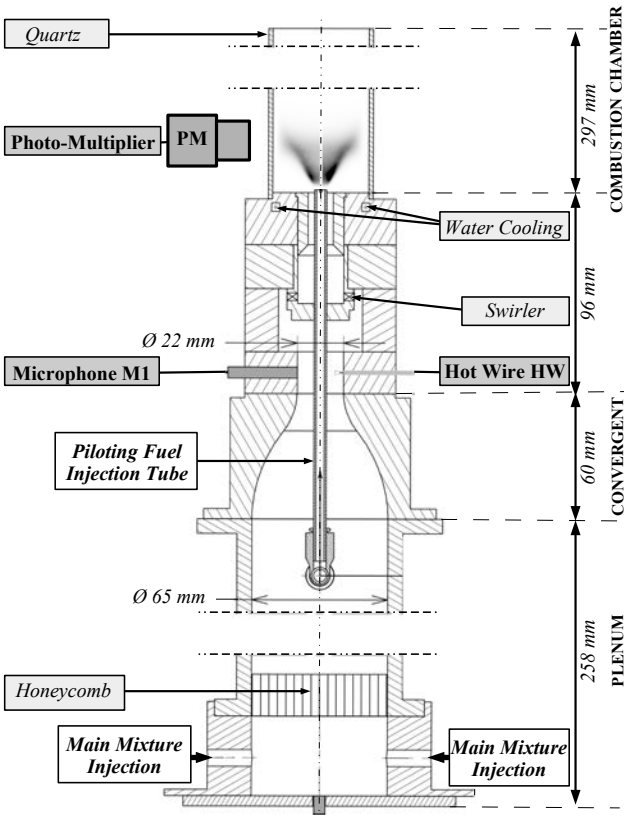

Figure 1: MIRADAS experimental setup on which the analysis is performed [18].

Table 1: Description of the injection configurations.

	Main injection	**Secondary injection**	N_{op}
Ref		X	56
PC1		pilot CH_4 (1 % in Pow.)	55
PC2	premixed	pilot CH_4 (2 % in Pow.)	52
PH1	CH_4/air	pilot H_2 (1 % in Pow.)	50
PH2		pilot H_2 (2 % in Pow.)	51
MH1		prem. H_2 (1 % in Pow.)	50
MH2		prem. H_2 (2 % in Pow.)	52
Total			366

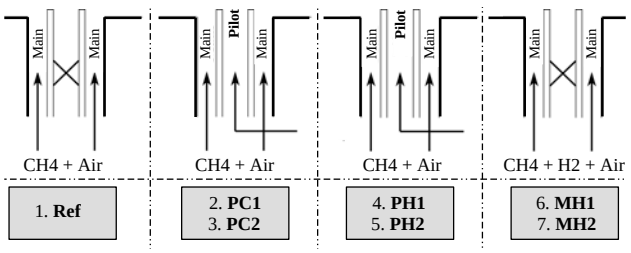

Figure 2: Schematic representation of the injection configurations [18].

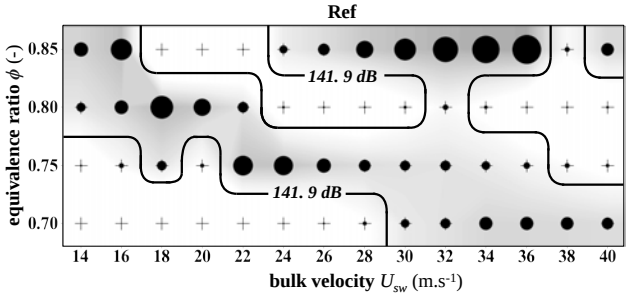

Figure 3: Mapping of the RMS value of the acoustic pressure from the microphone M1 in case Ref. The diameters of the circles are proportional to the RMS of the acoustic pressure level in dB.

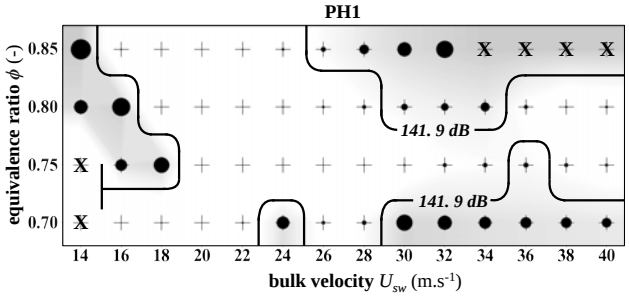

Figure 4: Mapping of the RMS value of the acoustic pressure from the microphone M1 in case PH1. The diameters of the circles are proportional to the RMS of the pressure level in dB. Operating points marked with "**X**" could not be acquired.

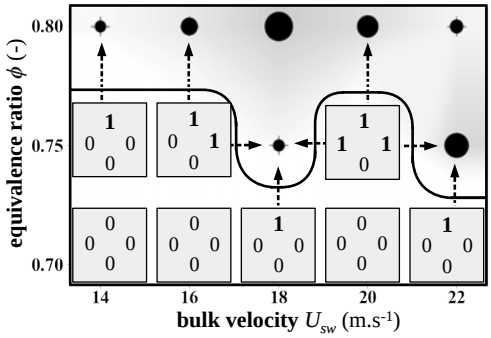

Figure 5: Representation of the label attribution for stable operating points.

Table 2: Detailed specifications of the network.

Layer	**Specifications**
Convolution 1	Filters : 32 Kernel Size : 13
Maxpooling	Pool Size : 2
Convolution 2	Filters : 32 Kernel Size : 11
Maxpooling	Pool Size : 2
Convolution 3	Filters : 64 Kernel Size : 9
Maxpooling	Pool Size : 2
Convolution 4	Filters : 128 Kernel Size : 7
Maxpooling	Pool Size : 2
LSTM	Units : 128
Fully Connected	Units : 256
Fully Connected	Units : 128
Fully Connected	Units : 4

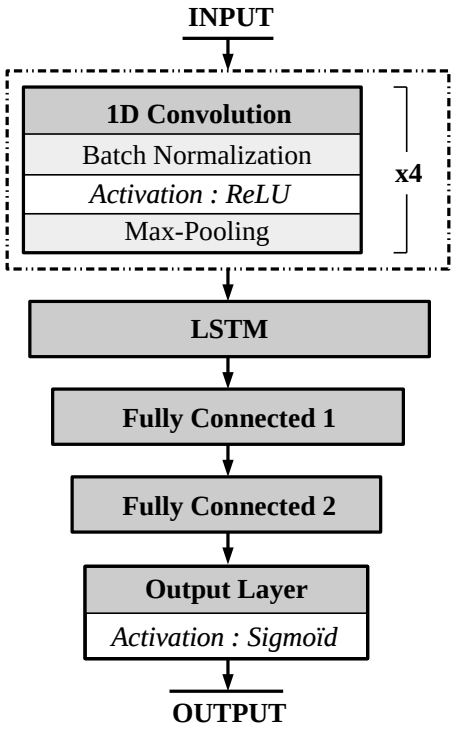

Figure 6: Schematic representation of the CRNN used in this study [24].

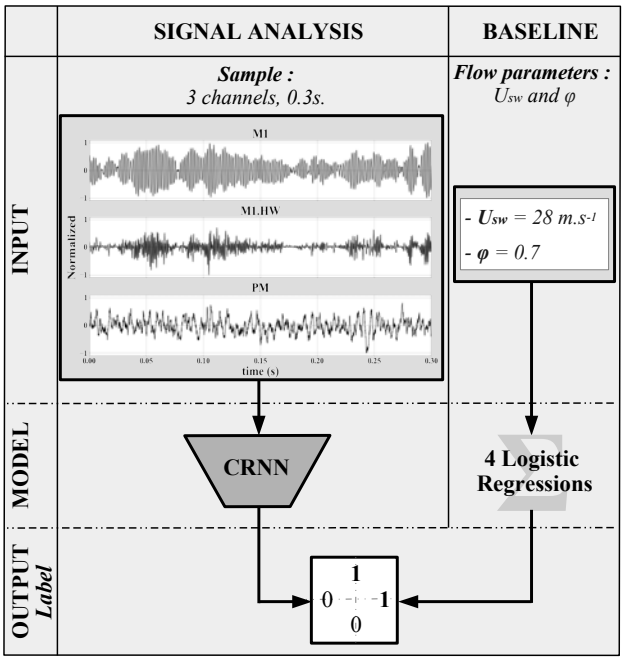

Figure 7: Summary of the learning task with an example of input and the corresponding label attribution. The sample is taken from Ref, $U_{sw} = 28\,\mathrm{m.s^{-1}}$, $\phi = 0.7$.

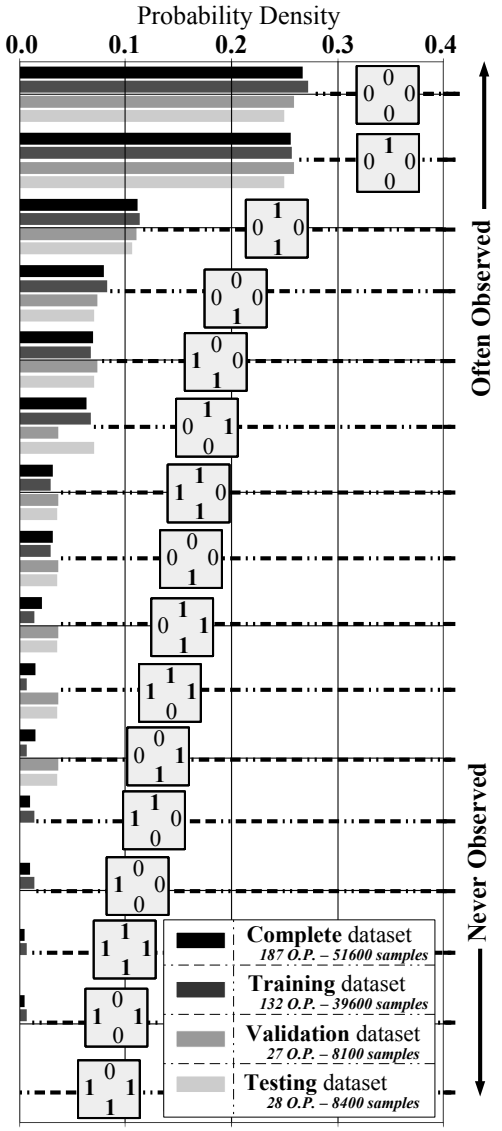

Figure 8: Distribution of the datasets after splitting regarding the occurrence of each type of observable label.

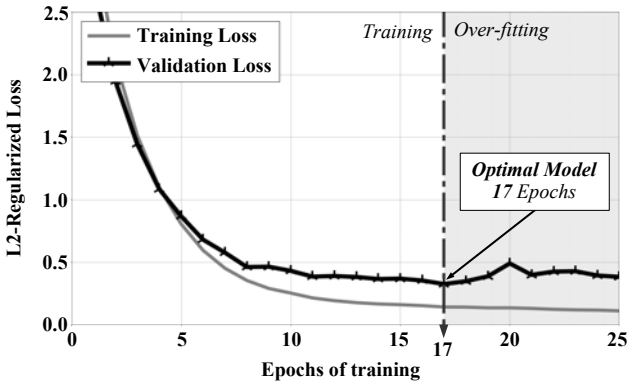

Figure 9: Plot of the L2-Regularized Loss (Binary Cross-Entropy) versus the epochs of training. The optimal model is obtained after 17 epochs. Further training decreases the training loss without improving the validation loss.

Table 3: Accuracy a in Equation (4) of the best trained model in comparison to the Baseline

	Trained Model	**Baseline**
Validation	90.6 %	77.8 %
Testing	87.7 %	76.8 %

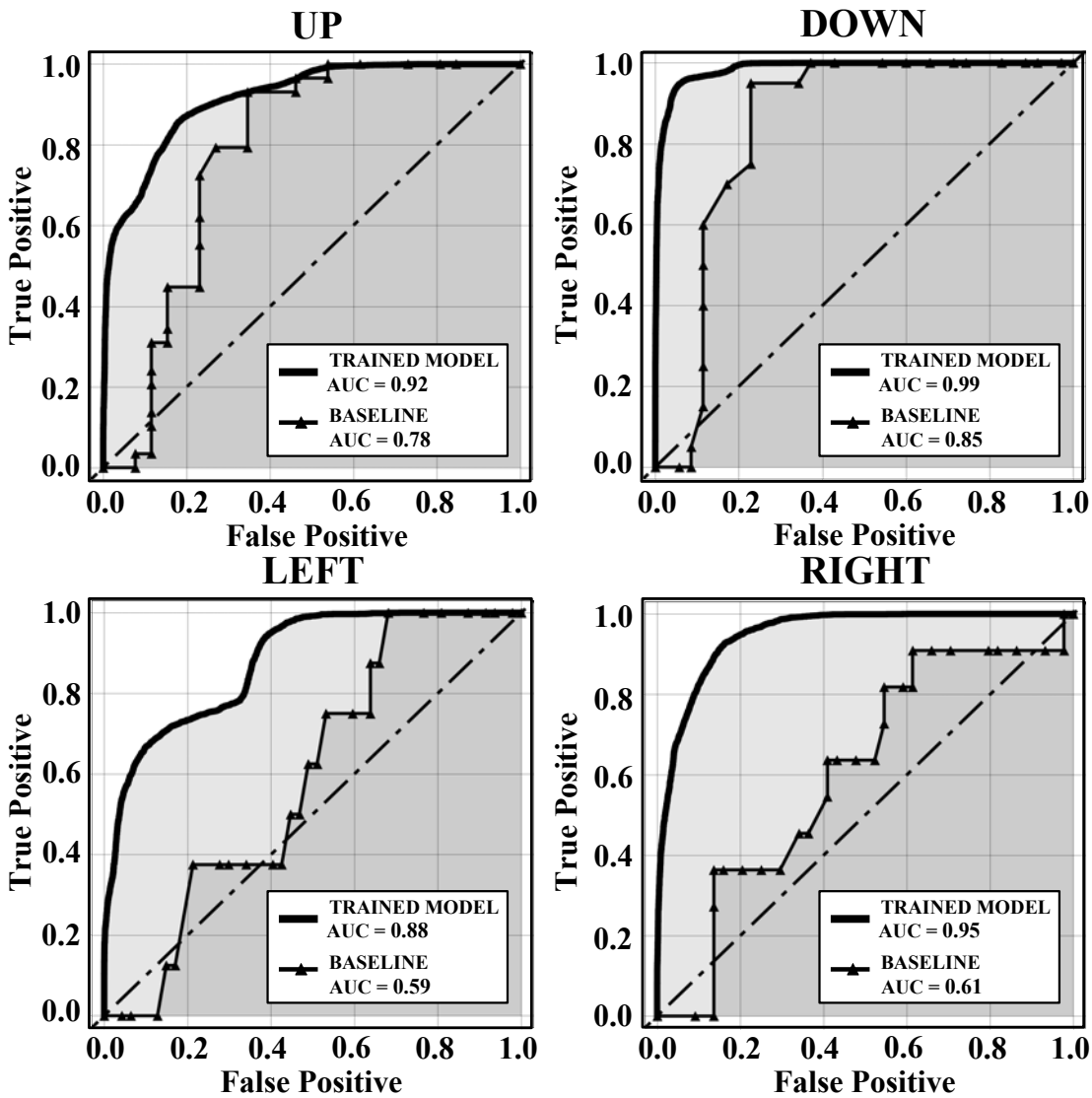

Figure 10: Receiver Operating Characteristic (ROC) curves computed for each direction evaluated by the model and its baseline. The Area Under the Curve (AUC) is given for each method and corresponds to the surface filled in different grey tones.

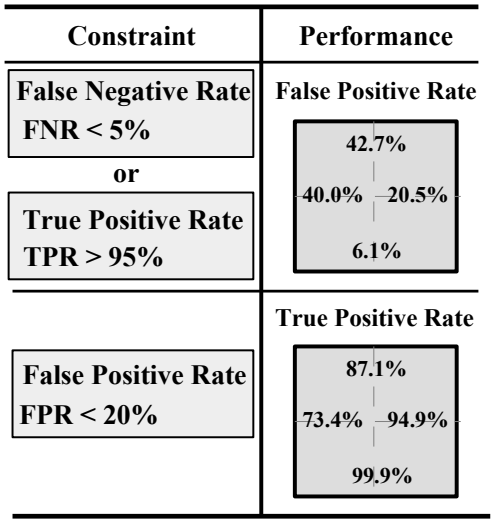

Figure 11: Choice of constraints and corresponding performances in terms of False Positive Rate (FPR) and True Positive Rate (TPR) for the four evaluated directions.

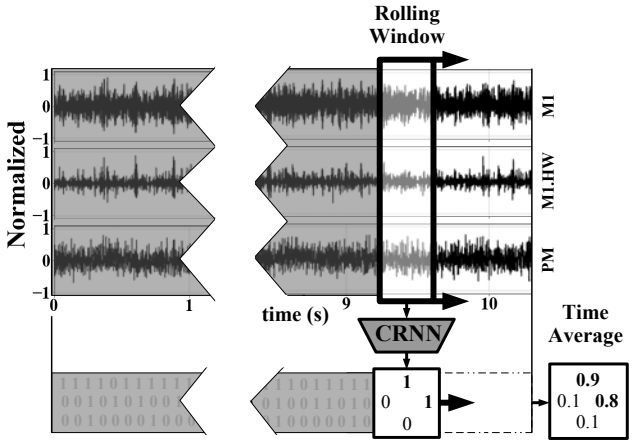

Figure 12: Representation of the rolling window evaluation made to simulate the prediction on a streaming signal.

Table 4: Influence of the number of channels on the performance of the method.

		1 Ch.	2 Ch.	3 Ch.
Validation	Accuracy	87.6 %	88.4 %	90.6 %
	BCE	0.33	0.28	0.27
Testing	Accuracy	84.8 %	87.5 %	87.7 %
	BCE	0.35	0.30	0.34
ROC-AUC	UP	0.89	0.91	0.92
	DOWN	0.98	0.98	0.99
	LEFT	0.78	0.84	0.88
	RIGHT	0.92	0.96	0.95

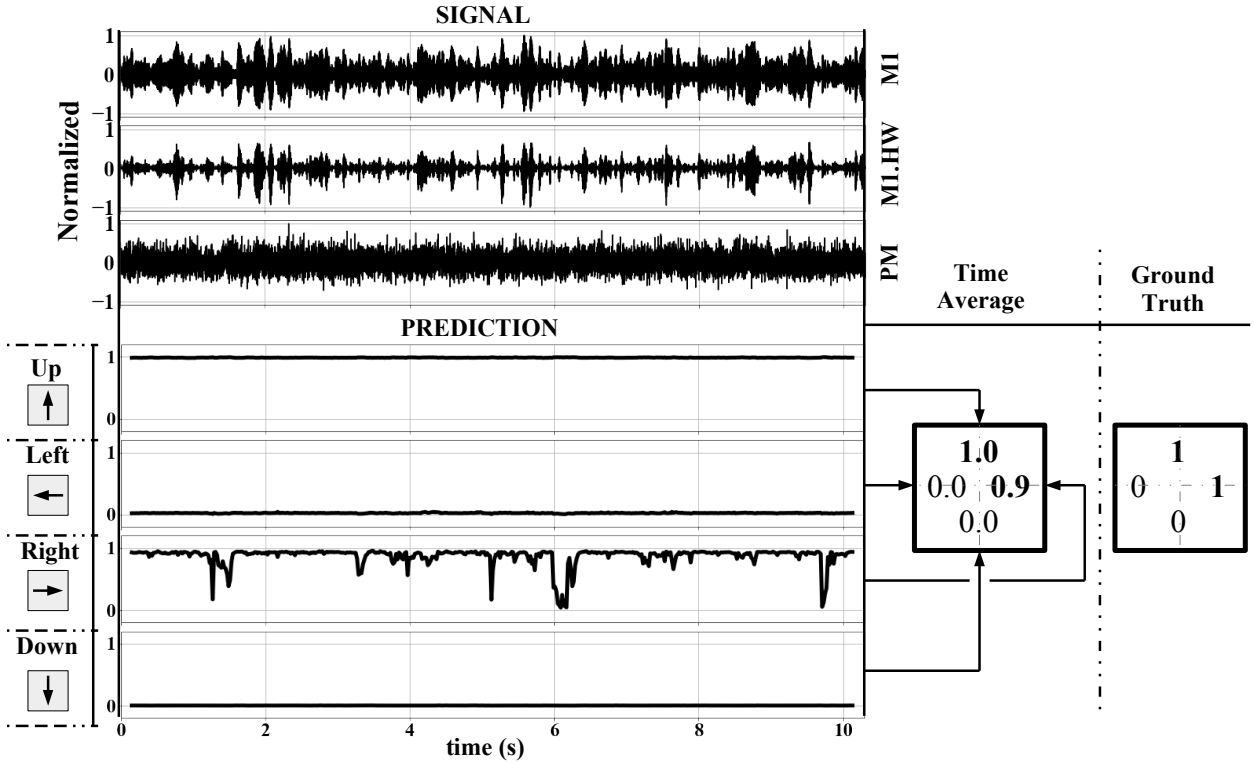

Figure 13: Rolling-window evaluation of the best trained network on **PC1**, $U_{sw} = 30\,\mathrm{m.s^{-1}}$, $\phi = 0.7$, an operating point of the testing set. Predictions are plotted in four graphs. Each graph is dedicated to one of the four directions asserted (up, left, right and down). The time average of the prediction for each direction is given, to be compared to the ground-truth values.

Table 5: Influence of the structure of the network on the performance of the method.

		MLP	**CNN**	**CRNN**
Validation	Accuracy	79.8 %	88.9 %	90.6 %
	BCE	0.45	0.30	0.27
Testing	Accuracy	79.7 %	84.9 %	87.7 %
	BCE	0.44	0.42	0.34
ROC-AUC	UP	0.71	0.91	0.92
	DOWN	0.97	0.97	0.99
	LEFT	0.77	0.86	0.88
	RIGHT	0.66	0.92	0.95