SESAR Engage KTN – catalyst fund project final technical report

Project title:	CARGO
Coordinator:	[redacted]
Consortium partners:	Università degli Studi di Padova – Dipartimento di Geoscienze
Thematic challenge:	TC3 Efficient provision and use of meteorological information in ATM
Edition date:	31 December 2020
Edition:	1.0
Dissemination level:	Public
Authors:	[redacted] / Università degli Studi di Padova
	[redacted] / GReD srl
	[redacted] / Ludwig-Maximilians-Uiversität München
	[redacted] / Leonardo

This project has received funding from the SESAR Joint Undertaking under the European Union's Horizon 2020 research and innovation programme under grant agreement No 783287.

1. Abstract and executive summary

1.1 Abstract

This project has combined measurements from different instruments to develop a nowcasting algorithm of extreme weather events in a localised area around the Malpensa airport with the aim of improving aviation safety. The radar reflectivity has been used as reference to define and select the extremes; Global Navigation Satellite System (GNSS) zenith total delay, atmospheric parameters from weather stations, and lightning have been used as inputs of a neural network to predict the development of the weather events in the near future (from 30 to 90 minutes before). The results show an accuracy of 0,75 in nowcasting the extreme events when using all the datasets as inputs and decreasing accuracy when excluding one of the inputs. However, there are still several tests that should be performed to understand the optimal setting of the algorithm. This project was the first experiment to collect so many atmospheric sensors in a localised area to nowcast extreme events with ATM purposes and posed the basis to develop a deeper study on this field.

1.2 Executive summary

Monitoring and predicting extreme atmospheric events, is very challenging especially when they develop locally in a short time range. Several studies have shown in the past the capabilities of ground based Global Navigation Satellite System (GNSS) for studying and predicting severe weather events. The GNSS allows the measurement of the zenith total delay (ZTD) strictly related to the atmospheric Integrated water vapour, which is the engine of the convection. Moreover, there is evidence that an abrupt increase in the total lightning discharge rate, often precedes severe weather occurrences on the ground.

The Lombardy region, in North Italy, is often affected by severe weather events with an increase of frequency and intensity in the last decades. Milano Malpensa airport is located in Lombardy and it is the second largest airport of Italy in terms of passengers (after Roma Fiumicino) and the first in terms of freight, so it is a good hotspot to develop a nowcasting algorithm for severe weather and to understand the impact of the severe weather on air traffic.

The objective of the project is to develop a nowcasting algorithm based on Neural Networks using input data from weather stations, lightning detectors and ground based GNSS receivers located in the area of interest and to create as output a polygon highlighting the risk for air safety.

We have selected more than 600 severe/extreme weather cases affecting the area of Malpensa in the period 2011-2019. The selection of severe weather events is based on echo radar intensity and rain rate from weather stations. For the predictive task required by this project, a feed-forward fully connected neural network has been chosen since it is the most versatile and is easily adaptable to our needs. Neural networks can in fact learn a wide variety of non-linear behaviours, as long as there are enough data for the training. The number of extreme events is small by definition and thus difficult to identify. The strong imbalance between meteorological circumstances that produce no occurrence of events, occurrence of average storms and occurrence of extreme storms makes the task extremely difficult. The neural network that we have chosen can capture non-linear behaviours between variables and is capable of accomplishing both regression and classification tasks with minor modifications.

We have run several different neural network configurations to understand which one could be the best for our case and we obtained the best algorithm accuracy by using all the variables as inputs (GNSS ZTD from 6 stations, meteorological parameters, lightning). However, the use of a smaller

number of GNSS receivers, slightly lowers the accuracy, so an optimisation of the network setting must be evaluated considering both the computational and the economic cost. The ZTD alone has a large impact on the prediction of the events, but the additional parameters help to improve the nowcasting by at least 20%. From the algorithm runs, it looks like the wind information decreases the capabilities of the nowcasting, however we wanted to deepen the analysis to understand the reason. A statistical analysis based on 40 case studies randomly chosen in the same period, shows that 78% of the extreme events are characterised by wind field convergence, just 25% of the cases have a decreasing trend of ZTD 30 minutes before the rainfall, and there is a never decreasing ZTD trend 30 minutes before the event corresponding to wind field divergence. This means that the wind direction and intensity contain a great value to forecast the extreme events, but neural network architecture that we implemented is not able to capture the relevant information. Thus, the algorithm can be improved to understand how to correctly provide the wind direction and intensity as inputs.

Other than the scientific results, several issues were highlighted during the project which will be the background for future studies and projects.

1.3 List of acronyms

ATM	Air Traffic Management
AUROC	Area Under the Receiver Operating Characteristic
FF	Feed-Forward
GNSS	Global Navigation Satellite System
GWMS	Ground Weather Management System
IWV	Integrated Water Vapour
LAMPO	Lombardy based Advanced Meteorological Predictions and Observations
NN	Neural Network
PPP	Precise Point Positioning
ReLU	Rectified Linear activation function
ROC	Receiver Operating Characteristic
SMOTE	Synthetic Minority Over-sampling TEchnique
SPIN	Servizio di Posizionamento Interregionale
TanH	Hyperbolic tangent function
TPR	True Positive Rate
TRT	Thunderstorm Radar Tracking algorithm
WISADS	Weather Information System for Airport Decision Support
WV	Water Vapour
ZHD	Zenith Hydrostatic Delay
ZTD	Zenith Total Delay
ZWD	Zenith Wet Delay

2. Overview of catalyst project

2.1 Operational/technical context

Airports are bottlenecks in the Air Traffic Management (ATM) network when impacted by thunderstorms in the vicinity. In this framework it is required to improve our understanding of deep convective processes locally developed to nowcast the events and provide a useful tool to the controllers.

Monitoring and predicting extreme atmospheric events, is very challenging especially when they develop locally in a short time range. Despite the great improvement due to the use of satellite measurements and improvements in model parametrisations, there are still large uncertainties on the knowledge of the dynamical processes of deep convective systems. Deep convective systems are destructive events causing every year many deaths, injuries and damages, they account for the major economic damages in several countries and they are one of the major risks for aviation safety. The number and the intensity of such phenomena increased in the last decades in some areas of the globe including Europe. In a climate change environment, the number and intensity of severe weather events are expected to increase. The atmospheric Water Vapor (WV) plays a key role in the storm development; in fact, a small increase of WV is amplified in weather system promoting more intense extreme events. We are currently able to forecast with good accuracy large systems within 12-24 hours, but many issues still come from small cells locally developing at small spatial and temporal scale because the models are not able to resolve them.

Several studies have shown in the past the capabilities of ground based Global Navigation Satellite System (GNSS) for studying and predicting severe weather events. The GNSS measurements allow the estimation of the atmospheric Integrated WV (IWV) which is the engine of the convection. Moreover, there is evidence that an abrupt increase in the total lightning discharge rate, often precedes severe weather occurrences on the ground having consequence on the rapid intensification of the updraft that leads to an increasing number of ice particle collisions and thus greater charge separation and lightning activity.

Lombardy region (North Italy) is often affected by severe weather events with an increase of frequency and intensity in the latest decades. Milano Malpensa airport is located in Lombardy and it is the second airport of Italy in terms of passengers (after Roma Fiumicino) and the first in terms of goods, so it is a good hotspot to develop a nowcasting algorithm for severe weather and to understand the impact of the severe weather on air traffic.

A network of ground based GNSS sensors was already placed in the area thanks to a previous project, a good meteorological stations network is managed by the Regional Environmental Protection Agency (ARPA), lightning detectors on the area of interest can be collected even though not specifically acquired for this purpose. The context, as a whole, looks adequate to develop the CARGO project.

2.2 Project scope and objectives

The project proposes an innovative approach integrating in situ and satellite-based observations/products for the improvement of meteorological forecasts. The final goal is the test of severe weather events nowcasting algorithm with high spatial resolution for improving aviation safety and the development of a user-friendly tailored final product easily understandable by the ATM stakeholder. Due to the short duration of the project the scientific analysis has been based on the background and network of a similar project (LAMPO) with the objective of forecasting floods in the Lombardy region, area particularly affected by seasonal inundations. A dense dedicated networks of monitoring units consisting in weather sensors (i.e. pressure, temperature, humidity, wind and rain)

and low-cost GNSS receivers are deployed in sensitive areas to work in synergy with already existing standard monitoring networks. The low-cost GNSS receivers are exploited as meteorological sensors, since the impact of the atmospheric WV along the vertical direction above the GNSS antenna can be determined from the analysis of the GNSS microwave signal delays. Two lightning detectors (Earth Networks Inc) are already present in the area and the data are available for the project; one additional detector (Biral Thunderstorm Detectors) has been purchased within this project and has been located on the top of a mountain looking toward Malpensa airport to increase the density and accuracy of the measurements. The objective of this project was to create an innovative nowcasting algorithm able to nowcast the development and intensification of convective cells from GNSS-derived measurements and lightning rate measurements. The development of such models is based on the comparison between the GNSS delay and lightning spatio-temporal behavior (potential heavy rain precursors) and the rain phenomena detected by meteorological radars. The nowcasting model should allow for the implementation of early warning systems to the advantage of public administrations and ATM. The project main objective is to pose the base for the deployment of a future network of low-cost sensors to nowcast instability and convection at very high spatial resolution with the purpose of supporting ATM and the establishment of a working monitoring and prediction system for the nowcasting of severe weather events, functional to the same issue. Although developed for a test area in Lombardy, it will be possible to replicate such system wherever needed, provided that a low-cost dense GNSS and lightning detector network can be deployed within an existing coarser GNSS network. This project aims to become the pioneer in integrating innovative techniques and to build an operative system, from the measurement to the early warning. For this it involves both academic players (Università degli Studi di Padova and Ludwig-Maximilians-Uiversität München) and operative entities (GReD srl and Leonardo Gmbh) working synergically to produce a complete system ready to be deployed and used.

The close connection between the academy, industry and final users will allow to define the optimal tool and a final product customised on the users' requirements. To demonstrate this, the final product will be integrated into the Ground Weather Management System (GWMS) prototype developed in SESAR and SESAR2020 as an implementation of the local 4DWxCube concept. The GWMS will tailor the data and forward them to the Weather Information System for Airport Decision Support (WISADS) which implements the meteorological requirements of TAM elaborated in PJ04 in SESAR2020 with OFA05.01.01 of SESAR 1 as a precursor. Both prototypes have been developed by Leonardo Germany who will take the role of an associated partner.

2.3 Research carried out

Input data

Meteorological data (ARPA Lombardia)

Weather stations distributed in the Lombardy region with 10-minute temporal resolution with 151 termometers, 61 barometeres, 114 hygrometers, 113 anemometers and 75 rain gauge sensors.

Ground-based GNSS receivers data (SPIN+GReD)

Time series of GNSS Zenith Total Delay (ZTD) with 30-second temporal resolution, derived from the processing of dual frequency data collected by the Servizio di Posizionamento Interregionale (SPIN) GNSS network and 9 low-cost single frequency GNSS sensors deployed within the project LAMPO. The standard GNSS network in the area of interest with just 4 stations available from 2001, has been made denser from 2018 by GReD srl adding 9 low-cost ground-based GNSS receivers well distributed in the whole area. ZTD time series have been estimated by applying the Precise Point Positioning (PPP)

strategy. With the assumption of an atmosphere in hydrostatic equilibrium, the IWV can be estimated from the ZTD by its decomposition into the Zenith Hydrostatic Delay (ZHD) dependent on the atmospheric dry air gasses and accounting for the greatest part of the tropospheric delay, and the Zenith Wet Delay (ZWD) due to the presence of the humidity in the atmosphere and so dependent on the tropospheric WV content. The ZWD is given by the equation:

ZWD=ZTD-ZHD

By scaling ZWD values with a conversion coefficient *c* (that is function of the weighted mean temperature of the atmosphere Tm), it is possible to retrieve IWV values:

IWV = *c* * ZWD

For this reason, the measure of ZTD is a good proxy to understand also the trend of IWV and, due to the fact that any conversion introduces an additional error, we decided to directly use the ZTD as input of our algorithm.

Radar data (Meteo Swiss)

Time series of radar attenuation (dBz) have been used as proxy variables to detect the presence of extreme events; those data were collected from a radar owned by the Swiss meteorological service (MeteoSwiss). The Thunderstorm Radar Tracking algorithm (TRT) developed by MeteoSwiss has been used to identify and track convective cells on radar-based EchoMax. The algorithm has the goal to identify areas with defined reflectivity threshold (minimum 36 dBZ), follow them over time and associate to the convection features provided by radar-based products and satellite images (i.e. top height of the cloud, storm motion, maximum reflectivity, probability of hail). The archive provides details about convective cell detections in terms of date/time of the centre of the cell, polygon interested by the cell, intensity of the storm.

Lightning detectors (Earth Network Inc)

The historical lightning dataset has been collected in the format date&time, coordinates, type of the lightning, amplitude of the signal, number of sensors used for the detection, and the possible error. For the purpose of this project, the data has been aggregated as total number of events collocated with the convective cells every 10 minutes. Several different aggregation tests can be made in future with this data, distinguishing the type of the lightning (cloud to ground flash, intra-cloud flash or any lightning that does not have a return stroke), the peak current and/or the height of the intra-cloud lightning. However, this makes more complex the algorithm and takes more time for the implementation and it is kept for future developments.

Data selection

We have selected more than 600 severe/extreme weather cases in the area of interest (Figure 1). The selection of severe weather events is based on echo radar intensity and rain rate from weather station with temporal resolution respectively of 5 minutes and 1 minute. The convective cells affecting the area of interest are tracked by using the so called Thunderstorm Radar Tracking (TRT) algorithm and the data are provided in the form of table in which we get the date, time, the coordinates of the cell centre, and cell dimension (Figure 2). We defined as severe events those ones with rain rate higher than the 95th percentile of a Gumbel distribution of all the archived rain rates characterising the area for the entire period (2001-2019) and insisting into the area for more than 25 minutes. A preliminary study, based on the analysis of several case studies, show that the convective cells usually develop in the area of the Malpensa airport and end in the north/northeastern part of the Region (Figure 1).

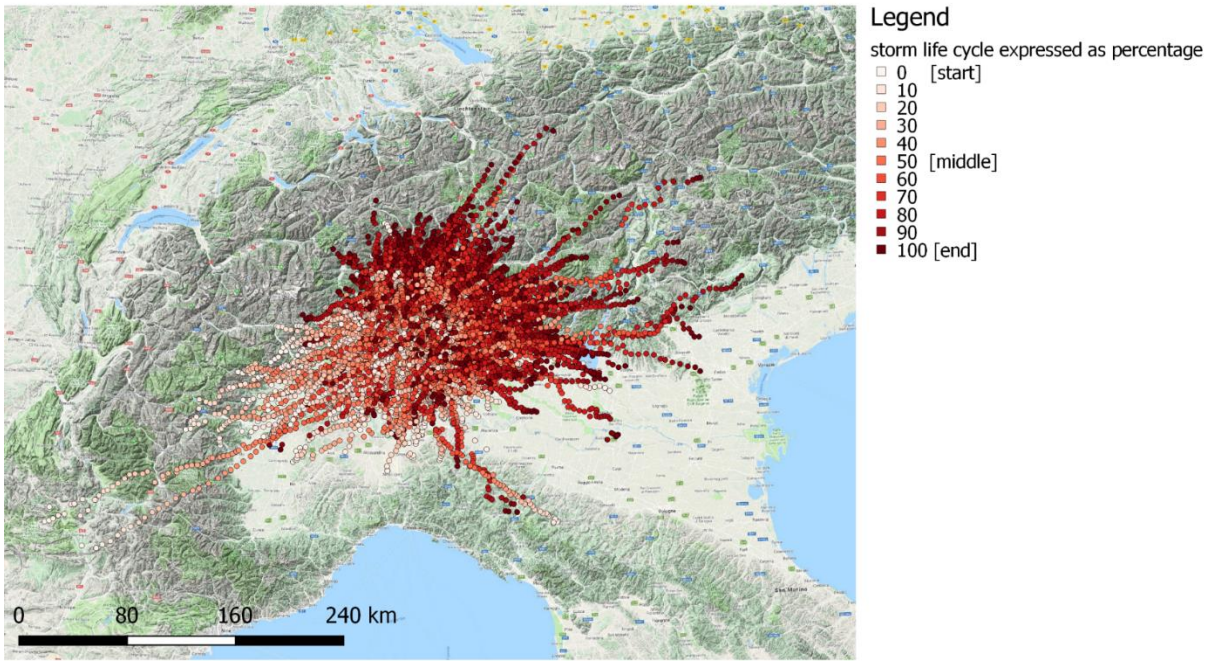

Figure 1. Collection of all the selected severe events developing in the area of Malpensa airport. Bright red denotes the initiation phase and dark red the ending phase of the system.

```
HEADER 201004011055_0001 201004011055 201004011115 25
**********
CELL-INFO  201004011055       36.0000 45.32993986 8.89281606       39.8667       45.0000 15.00000            NaN           NaN
NR-COORD 201004011055        11
LAT-COORD        45.354174       45.345345       45.309376       45.309212       45.300055       45.299720       45.308713       45.317536
LON-COORD        8.8738878       8.8609022       8.8599797       8.8727241       8.8852339       8.9107182       8.9109569       8.9239418
BZC NO       0       0       0       0       0       0       0       0       0
MZC NO       0       0       0       0       0       0       0       0       0
**********
CELL-INFO  201004011100       36.0000 45.33435758 8.90492601       39.3947       43.5000 19.00000       11.2845       6.10519
NR-COORD 201004011100        9
LAT-COORD        45.363001       45.354174       45.309212       45.300055       45.299720       45.317365       45.362326       45.362835
LON-COORD        8.8868775       8.8738878       8.8727241       8.8852339       8.9107182       8.9366879       8.9379036       8.8996342
BZC NO       0       0       0       0       0       0       0       0       0
MZC NO       0       0       0       0       0       0       0       0       0
**********
CELL-INFO  201004011105       36.0000 45.34788143 8.91721655       40.8409       46.5000 22.00000       11.2642       10.1559
NR-COORD 201004011105        14
LAT-COORD        45.380652       45.371827       45.371994       45.363167       45.336189       45.317873       45.317365       45.335003
LON-COORD        8.9128693       8.8998713       8.8871126       8.8741207       8.8734221       8.8984494       8.9366879       8.9626741
BZC YES      2       1       1       0       0       0       0       0
MZC NO       0       0       0       0       0       0       0       0       0
**********
CELL-INFO  201004011110       36.0000 45.36009064 8.92952015       40.1087       44.5000 23.00000       11.2639       12.2725
NR-COORD 201004011110        19
LAT-COORD        45.398637       45.389812       45.380820       45.371994       45.345017       45.335858       45.335520       45.326357
LON-COORD        8.9133483       8.9003460       8.9001086       8.8871126       8.8864075       8.8989231       8.9244237       8.9369308
BZC NO       0       0       0       0       0       0       0       0       0
MZC NO       0       0       0       0       0       0       0       0       0
**********
CELL-INFO  201004011115       36.0000 45.37029913 8.93953623       39.2500       42.0000 18.00000       10.5621       12.7807
NR-COORD 201004011115        9
LAT-COORD        45.398637       45.389812       45.362835       45.344512       45.343820       45.379789       45.398123       45.398467
LON-COORD        8.9133483       8.9003460       8.8996342       8.9246648       8.9756734       8.9766717       8.9516424       8.9261131
BZC NO       0       0       0       0       0       0       0       0       0
MZC NO       0       0       0       0       0       0       0       0       0
```

Figure 2. Table format of Thunderstorm Radar Tracking (TRT).

Data pre-processing

We have worked with different types of data, not specifically set to create a forecasting algorithms and managed by entities with different objectives and requirements. As a consequence, before using the data for our purposes we needed a strong pre-processing to homogenise the acquisitions and to fill the gaps often present. Every time series is affected by a number of issues, the most common being random or systematic absence of data and presence of outliers and duplicates. Moreover, every

sensor was activated in a different time causing each series to have a different length. Series also have different time steps depending on the device that generated it.

The following data pre-processing steps were performed on each time series:

- The GNSS ZTD, the radar and the lightning data has been re-sampled with 10-minutes temporal resolution which is the temporal resolution of the meteorological stations;
- Not a Number values have been added to all the missing 10-minute samples;
- Duplicates present in some meteorological stations were removed;
- Outliers were removed.

Meteorological data clustering

Since we have a large number of meteorological stations in the area of the airport, the data clustering provides a solution to two problems:

- it allows for a reduction in the number of input variables, and consequently of the network complexity. A high network complexity would have required training datasets larger than the one at our disposal;
- it can be exploited to overcome the problem of missing data in the input time series.

Regularity of the input data is an essential requirement for neural network algorithms. Since extreme situations are of utter importance in the project, interpolation of missing data is not a desirable solution as it softens possible missing peak values.

The first raw attempts to overcome these obstacles were based on the assumption that aggregating all of the sensors was enough to deliver the necessary information to describe a meteorological variable. This was obtained by computing the mean and the maximum of the measures from every sensor of a specific variable at each time step, disregarding missing values. Results were not satisfying, as the loss of information turned out to be significant.

Instead, the K-Means clustering algorithm was used to subdivide the sensors in 3 to 5 groups for each variable. Two approaches were tested. The first clustering was based on longitude, latitude and height of each sensor. Clustering was performed on the spatial position of the sensors rather than their actual behaviour under the assumption that meteorological variables don't change too quickly in space. This way, some geographical information is implicitly passed to the model. The second clustering was performed using the values of the time series of each sensor as features in the K-Means algorithm. In this way sensors with similar time series are grouped irrespectively of their position.

We have found out that similar behaviour mainly corresponds to the spatial closeness of the stations. This exercise has been really useful to understand the optimised meteorological stations network for this area and for the available network setting.

Operations before training

In 2011 the meteorological radar in Switzerland was down for maintenance and collected no dBz data for the whole year. Since dBz is going to be used as output for the model, the whole year has been removed.

The GNSS stations except the one in Como, have several sequences of missing data between 2014 and 2015, so the extreme cases in this period with missing data have been removed from the training dataset.

As the best and common modelling practices suggest, the dataset was divided in three parts: years from 2010 to 2017 (included) were used for training, year 2018 was used for validation and year 2019 was used for testing.

Common practice in neural network training also suggests to standardise data in order to have inputs with zero mean and unit variance. This operation aims at levelling off the magnitude of all the input variables, in order to avoid strange effects that may occur during training (differences in the relative magnitude of the inputs may influence the parameters initialisation or push the training to value some variables over other ones). This operation was performed on the whole dataset using the mean and the variance of the training set solely.

Our experience suggests using the anomalies of the different parameters versus the climatological values to highlight the presence of the extreme events, so we computed the climatological values of ZTD, temperature, relative humidity, and pressure and then subtracted them from the actual values to create the anomalies of each parameter. However, the runs using the anomalies as input are still on going and we cannot report the results yet.

Methodology

An artificial Neural Network is a mathematical model that mimics the functioning of a biological brain in order to describe a phenomenon.

For the predictive task required by this project, a feed-forward (FF) fully-connected neural network was chosen among the many different architectures that exist. A FF architecture is the most versatile kind of network and is easily adaptable to our needs. It can capture non-linear behaviours in the relation between variables and is capable of accomplishing both regression and classification tasks with minor modifications. The (simplified) scheme of such a network is shown in Figure 3.

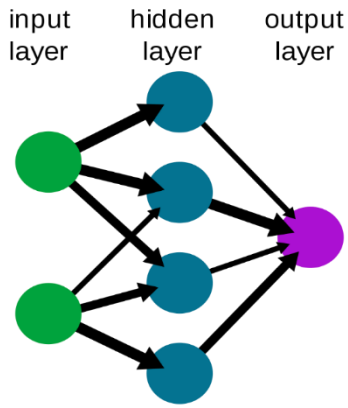

Figure 3. Simplified structure of a feed-forward fully-connected neural network.

The coloured nodes are the neurons of the network and represent some kind of activation function, while the black arches carry a weight and show the path followed by the inputs:

- The input layer is just an identity function that takes the raw input and sends it through the net without changing it. There are as many input neurons as the dimension of the input;
- The data coming from the input layer are linearly combined via the weight of the arches and are fed to the hidden neurons;
- The neurons in the hidden layer represent the activation functions: they take the aggregated inputs and give out a value that is fed forward into the net. Common activation functions

include the rectified linear activation function (ReLU) and the hyperbolic tangent function (tanH).

The ReLU is a linear function that will output the input directly if it is positive, otherwise, it will output zero. It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance.

The tanH is nonlinear activation function that outputs values between -1.0 and 1.0.

- The outputs of the activation functions are again linearly combined and fed to the next layer, where the previous steps are iterated;
- The last layer of the net is the output layer and contains as many neurons as the dimensions of the output. This neuron has an activation function that may be a ReLU or a tanH for regression tasks and a sigmoid for classification. The output of this layer is sometimes referred to as “firing power” in literature and represents the final output of the whole model.

This kind of model has a high number of parameters (weights, W) that depends on the dimension of the input (I), the presence of biases (B) (the addition of which represents another common practice), the number of layers, the number of neurons (H) in each layer and the dimension of the output. Since it is impossible to a priori know which architecture would best fit the data, a large number of different networks are trained and the one with the best performance is selected.

The training of an FF NN is efficiently performed using the back-propagation algorithm, based on the following steps:

- weights are initialised;
- input is fed into the net and the output is computed;
- The output is compared to the target using an adequate metric, called loss function J(W);
- The gradient of the loss function with respect to the weights is computed;
- The error is back-propagated through the network and the weights are updated using the Hebbian Rule;
- The process is iterated until convergence.

Every time the inputs have been fed to the net once, an “epoch” has occurred.

Since the weight update can occur in three different moments during the process, three kind of learning are defined:

- Online learning: the error is computed after each single input is fed into the net and the weights are immediately updated, the resulting loss function is noisy but it converges to a (local) minimum;
- Batch learning: the error is computed after every input is fed into the net but its value is stored in memory until an entire epoch occurs. The weights are updated at the end of the epoch, which means that the network has “seen” all the dataset before updating the weights. In this case, one iteration corresponds to one epoch and the loss function is smooth but the process is more memory intensive;
- Mini-batch learning: mixing the previous approaches, a small set of inputs is fed to the network before upgrading the weights. It has been proven to be the best method, since it has the advantages of both previous ones.

In our case, the mini-batches are fixed sequences, but the order in which they are fed to the NN is random. With the mini-batch learning, there are many critical choices to be made before proceeding with the training: the network architecture, the heuristic to initialise the weights, the loss function, the optimiser, the learning rate (η), the dimension of the mini-batches. In particular, some of those

points can be treated as hyper-parameters to be optimised. In fact, the architecture of the net can be expressed in terms of number of neurons per layer and by number of layers, and it is linked to the input and output dimensions. Moreover, the dimension of the mini-batches can be varied arbitrarily, although it is usually expressed as a power of 2 and the learning rate depends, although not explicitly, on both previous numbers.

The other choices depend mostly on the problem to be solved, if the task is regression, a Mean Square Error loss function and an Adam optimiser are well suited, in other cases instead, a Binary Cross-entropy loss function and a Stochastic Gradient Descent optimiser must be chosen.

Concerning the weight initialisation, the Glorot method is the state of the art for deep-NN (that is a NN with more than one layer) which choses weights from a normal distribution with 0 mean and variance dependent on the dimension of the network.

The design of a NN is strongly defined by the purpose the net must fulfil. Within this project we are interested in regression with the objective to find a continuous relationship between different scalar variables and in classification given a set of labelled data, with the objective to identify the class of an element. The purpose of finding extreme events proves to be particularly insidious and challenging. Neural networks can in fact learn a wide variety of non-linear behaviours, as long as there are enough data for the training. The number of extreme events in a time series are small by definition and thus difficult to identify. The strong imbalance between meteorological circumstances that produce no occurrence of events, occurrence of average storms and occurrence of extreme storms makes the task extremely difficult.

This problem can be managed in two ways. The first method is training a NN for a regression task on the radar reflectivity from the convective cells and then put thresholds based on previous knowledge to separate between average events and extreme events. The second method consists in defining classes a priori and then training a NN for a classification task in order to distinguish between two or three classes. The classification task can be carried out using different combinations of classes:

- 2 classes: “no event” vs “every kind of event”
- 2 classes: “average event” vs “extreme event”
- 3 classes: “no event” vs “average event” vs “extreme event”

The imbalance between classes represents one of the main obstacles in this project. Depending on the task, different actions can be taken to tackle the problem. As both approaches have major drawbacks, it must be noted that the improvement is small, but useful. In classification tasks an over-sampling technique can be applied to the class with fewer elements. Such method is called Synthetic Minority Over-sampling TEchnique (SMOTE). It builds new data modifying randomly those belonging to the minority class, increasing the amount of samples in the latter. This technique gives slightly better results but it has a major flaw as the new samples are random modifications of the initial dataset and there is no check if the new tuples fall in the majority class. In regression tasks, data are not labelled and the SMOTE cannot be applied. In order to re-balance classes, a different method can be implemented. The idea is to “drive” the training of the net deleting mini-batches that do not contain at least one minority class sample. This operation can be interpreted as a majority class sub-sampling, but in a more naïve approach. Nevertheless, it has an edge with respect to random sub-sampling as the chosen approach preserves some temporal dependencies between the data. The gain in terms of performance of the model is mediocre but still visible. It must be noted that in this case whatever performance metric one should use, the evaluation on the training set will not be much higher than on the validation and test sets because not all the training samples have been fed to the net during the training phase.

Evaluation

In order to assess the performance of each trained model, a suitable evaluation metric must be chosen. This metric should be able to account for different aspects of the model behaviour and for the task the model is created for. For the regression task, the model output is the flat value of dBz at a specific time step. As such, the model assessment can be performed using a distance metric between output and target, the classical R^2 statistic has been chosen. The coefficient of determination is simple to compute and gives a measure of how much the model performance increases with respect to the average of the data:

- $R^2 = 1$ means that the model perfectly fits the data;
- $R^2 = 0$ means that the model has the same performance as taking the average of the target (i.e. useless model);
- $R^2 < 0$ means that the model behaves worse than the average of the target (i.e. the model produces a systematic error and/or fails in learning the desired behaviour from the data).

For the classification task, the output of the model is the probability of belonging to each class. Usually, an automatic selection of the class with highest probability can be performed in order to obtain a one-dimension output each time step. In fact, it is good practice to avoid this operation and instead explore the model behaviour by putting and varying some thresholds on the class probabilities, i.e. performing a sensitivity analysis on the thresholds. The importance of this step will be clearer later.

Once the output is computed, the most common approach to model assessment for a classification task is producing the confusion matrix and computing recall and precision for each class and overall accuracy for the model. The confusion matrix is a tool to visualise the number of outputs correctly classified in each class and the ones misclassified, giving an insight on the type of errors made by the model. It is sometimes useful to divide the number in the matrix by the total number of sample, switching from the absolute values to the corresponding rates. The confusion matrix is squared with as many dimensions as the number of classes (i.e. 2 classes -> 2x2 matrix, 3 classes -> 3x3 matrix). An example is provided in Figure 4, instead of true and false, we have class + and class -. In the 3-classes case, a one-vs-the rest approach may be used to compute precision and recall for each class instead.

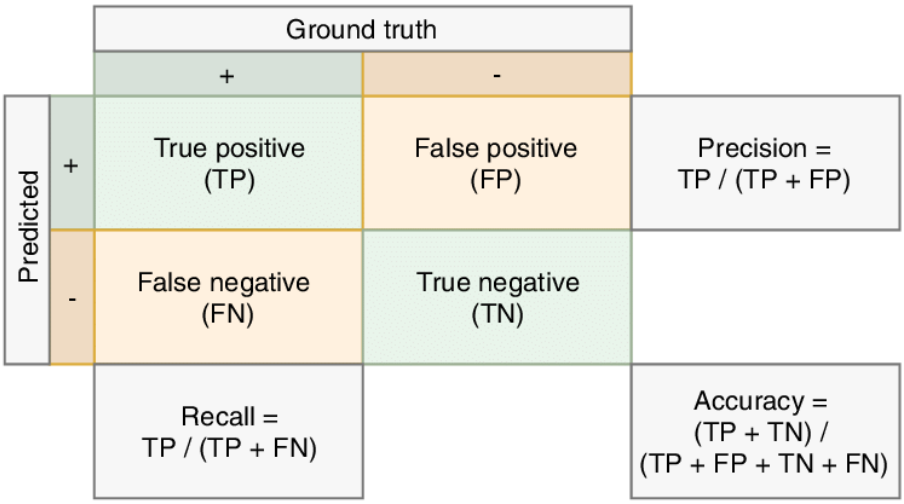

Figure 4. Confusion matrix, also reporting statistics parameters (Precision, Recall and accuracy).

Statistics that can be computed from the confusion matrix are:

- Precision: number of samples of one class that are predicted correctly, divided by the total number of true elements of that class;
- Recall: number of samples of one class that are predicted correctly, divided by the total number of elements predicted in that class;
- Accuracy (or "overall accuracy"): sum of the elements on the diagonal of the confusion matrix, divided by the total number of elements in the matrix.

The overall accuracy seems to be the most important metric to look for evaluating the model performance. Actually, information conveyed by that number alone is not sufficient to capture the true behaviour of the model, especially in highly imbalance datasets as in our case. When classes are imbalanced, in fact, the overall accuracy masks different types of errors made by the model on the minority class. Assume to have a confusion matrix as the following Figure 5:

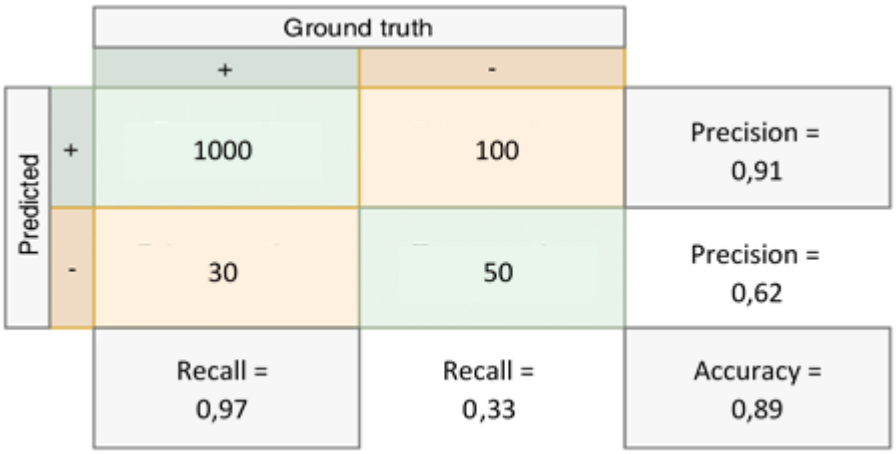

Figure 5. Example of confusion matrix.

As can be easily reckoned, in this example the performance of the model in its entirety is good enough. The overall accuracy is as high as 89%, meaning that the model makes a correct prediction almost 9 times out of 10. Taking a quick look at the performance of each individual class, though, reveals that the model is strongly biased towards the positive class: in fact, only the 33% of the samples belonging to the minority class are correctly identified against the 97% of the majority one. Moreover, only 50 out of 80 samples predicted as negative are correct, accounting for as low as the 62% -much lower than the 91% of the majority class.

For this reason, in order to extract all the useful information from a confusion matrix it is necessary to look at multiple statistics, turning the model assessment in a multi-objective problem with at least five variables. Considering that the number of NN trained for each experiment varies between 144 and 288, the problem becomes intractable. To overcome this obstacle, a new evaluation method is introduced: the Receiver Operating Characteristic (ROC) curve.

The ROC curve is a graph obtained computing the variation of the True Positive Rate (TPR) and the False Positive Rate (FPR) due to changes in the thresholds on the probability distributions of the model's output. Exploring the response of the confusion matrix to changes in the thresholds allows us to infer how much the model is capable of distinguishing between different classes. For the 2-classes problem, only one curve is computed, as the curve of one class is the reciprocal of the curve of the

other class; for the 3-classes problem a 1-vs-the rest approach may be used and the different curves are then aggregated.

The area under the ROC curve is used (AUROC) is often considered as a suitable metric for model assessment. In particular, the higher the area under the ROC curve, the more the model is able to distinguish between the classes. Since the TPR and the FPR range from 0 to 1, the maximum area is equal to 1. If the ROC curve lies on the diagonal of the plot, the area underneath equals 0,5. Figure 6 shows how the AUROC works.

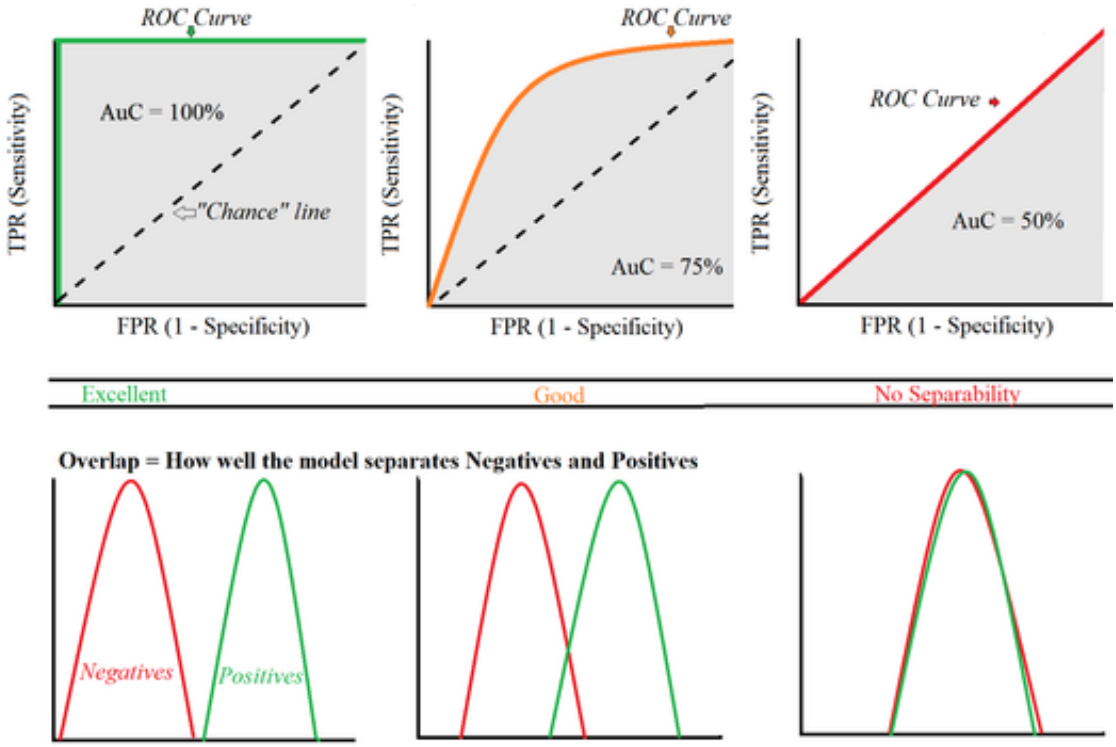

Figure 6. Scheme showing how Receiver Operating Characteristic (ROC) curve works and how good are the performances of the model to distinguish different classes.

The graphs on the lower line are examples of probability distributions produced by the model, while the plots on the upper line show the behaviour of the corresponding ROC curve and the values of the AUROC. If the model does not distinguish between classes, it generates two completely overlapping probability distributions and the AUROC is 0.5. On the opposite, if the model perfectly separates the two classes, the distributions do not overlap and the AUROC is 1. On this scale, a result is considered satisfying if the AUROC is at least over 0.6. The ROC curves are generated placing thresholds on the output of the model. Varying those thresholds allows the computation of the distribution probabilities of the outputs and the subsequent plot of the ROC curves.

In order to evaluate the model performance in a cleaner and more intuitive fashion, each point of the ROC curves can be transformed back into confusion matrices. In particular, it is of most interesting the threshold corresponding to the intersection between the ROC curve and the counter-diagonal of the graph: this operating point generates a confusion matrix that has equal elements on the diagonal. This peculiar choice allows the analyst to completely overcome the class imbalance problem and to compare different models fairly.

2.4 Results

We performed several experiments using a different number and type of inputs, but the details and performances of the experiments will be shown later (Figure 8). We first show the results of 2 experiments by training the NN using just the ZTD as input to understand what is the impact of a different number of GNSS stations on the algorithm performance.

The 1st experiment was performed using the ZTD information from 6 sensors around the area of interest to predict the presence of a storm. In the Table 1, the confusion matrix resulting from the original output of the model (on the left) and the confusion matrix obtained adjusting the thresholds (on the right).

		Ground truth	
		Event	No event
Predicted	Event	**0,45**	**0,22**
	No event	**0,55**	**0,78**

		Ground truth	
		Event	No event
Predicted	Event	**0,67**	**0,33**
	No event	**0,33**	**0,67**

Table 1. Nowcasting using ZTD from 6 GNSS receivers. Right: Confusion matrix from the original output. Left: confusion matrix from the adjusted thresholds.

The 2nd experiment was performed using the ZTD information from 3 sensors around the area of interest to predict the presence of a storm. In Table 2 left, the confusion matrix resulting from the original output of the model. In Table 2 right, the confusion matrix obtained adjusting the thresholds.

		Ground truth	
		Event	No event
Predicted	Event	**0,37**	**0,16**
	No event	**0,63**	**0,84**

		Ground truth	
		Event	No event
Predicted	Event	**0,62**	**0,38**
	No event	**0,38**	**0,62**

Table 2. Nowcasting using ZTD from 3 GNSS receivers. Right: Confusion matrix from the original output. Left: confusion matrix from the adjusted thresholds.

In order to test if the number of sensors used for the prediction has an impact on the model behaviour, some events have been isolated. The characteristic of those events is that they appear South-West of the area of interest and move North-East. Each event is composed by many steps that were tracked by the NN, producing a sequence of predictions for each event. The performance of the models is showed in Figure 7 as a survival curve: the graph shows the number of events predicted with a specific overall accuracy, that is the percentage of steps predicted correctly in each event. It should be noted that the red line is always higher than the blue one, meaning that the 3-sensors network has worse performance on predicting events coming from South-West.

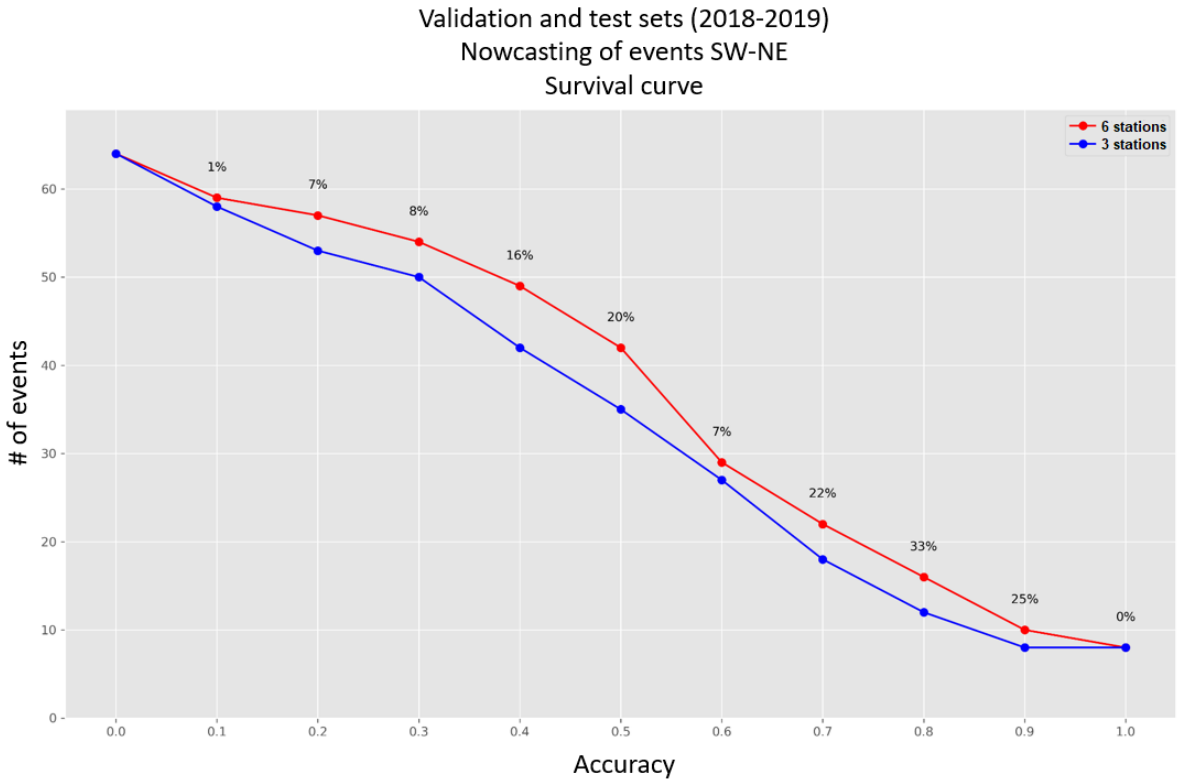

Figure 7. Survival of the nowcasting of extreme events by using as NN input the ZTD from 3 or 6 GNSS stations.

Figure 8 shows the results of all the experiments carried out. In the first column there is a brief description of what was used as input; in the columns from two to four there are the "adjusted" accuracies computed on training, validation and test sets - the most important being the latter; in the fifth column there is the number of training samples used in the training phase, while in the sixth column is shown the number of parameters of the NN, i.e. the dimension of the network: the ratio between the last two numbers is shown in the last column.

The accuracy of the model on the test set is the most important value to look at, as it is a good metric for the model generalisation capability. Usually, the accuracy on the training set is very close to 1: in our case, though, the training dataset does not correspond exactly with the dataset used in the training phase because of the application of the SMOTE technique to overcome the class imbalance problem. As a result, the performance of the training and validation sets tend to be similar. Some of the results with lower performances can be explained looking at the ratio between the number of training samples and the number of parameters: the lower the ratio, the lower the ability of the NN to learn complex non-linear behaviours. A rule of thumb often applied in NN projects is that there should be at least 30 sample for each parameter to be trained. It should be noted that the networks 1 and 3 have the biggest input. On the other side, it is not guaranteed that the variables used to make predictions are actually the most One aspect that needs further exploration is the innate difficulty to make predictions in a particular year. In other words, the events occurred during year 2018 – which is used as a test - might have been triggered by phenomena that are slightly different from the drivers of the storms occurred in the previous years: the variation in the meteorological variables might have been quicker or slower and more or less concentrated in space. As a result, the NN could have been trained on an "easy" dataset and the results on the test set might be biased. Due to the limited dimension of the input of experiment number 5, this test was chosen to carry out a brief exploration on the importance of past time steps to include in the input.

Definitions:

- Lead: number of timesteps in the future, i.e. the time interval between the moment the prediction is performed at the instant in which the prediction should occur;
- Lag: number of timesteps in the past used to make the prediction.

The time step is 10 minutes, so 3 steps equal 30 minutes, 6 steps equal 1 hour and 9 steps equal 90 minutes.

The best accuracy in the test dataset was achieved by using all the variables as input (GNSS ZTD from 6 stations, meteorological parameters, lightning) with an accuracy of 0.75. In this case the number of training samples is very large (almost 150000), but also the number of parameters is large and the ratio between the two is close to 30 as requested by the NN. Using 3 GNSS ZTD instead of 6, decreases the accuracy to 0.73, while excluding the some of the inputs or nowcasting the events more in advance makes the performance of the algorithms coarser. The minimum accuracy (0.58) is achieved by only using the ZTD and wind direction.

Experiment:	Training set	Validation set	Test set	Nr of training samples	Nr of parameters	Ratio
1) All variables, clusters on normalised coordinates, min/avg/max for each cluster	0,84	0,80	0,75	146.067	5.402	27,04
2) All variables, clusters on normalised coordinates, 1 sensor for each cluster	0,82	0,81	0,69	93.400	4.202	22,23
3) 3 ZTD sensors, clusters on normalised coordinates, min/avg/max for each cluster	0,82	0,81	0,73	146.067	6.242	23,40
4) All variables, clusters on time series, 1 sensor for each cluster	0,80	0,80	0,67	87.175	2.222	39,23
4) All variables, clusters on time series, 1 sensor for each cluster; -lead: 6 -lag: 3	0,80	0,80	0,67	87.175	2.222	39,23
5) As above; -lead: 6 -lag: 6	0,83	0,80	0,69	69.820	3.542	19,71
6) As above; -lead: 6 -lag: 9	0,81	0,80	0,66	57.435	2.452	23,42
7) ZTD and pressure, clusters on normalised coordinates, 1 sensor for each cluster	0,68	0,75	0,61	151.795	152	998,65
8) ZTD and relative humidity, clusters on normalised coordinates, 1 sensor for each cluster	0,71	0,68	0,61	153.763	4.142	37,12
9) ZTD and temperature, clusters on normalised coordinates, 1 sensor for each cluster	0,71	0,70	0,64	147.717	2.042	73,34
10) ZTD and wind direction, clusters on normalised coordinates, 1 sensor for each cluster	0,64	0,62	0,58	120.434	2.042	58,98
11) ZTD and wind velocity, clusters on normalised coordinates, 1 sensor for each cluster	0,60	0,64	0,62	125.482	2.402	52,24
12) ZTD, wind speed and direction, clusters on normalised coordinates, 1 sensor for each cluster	0,71	0,67	0,64	103.056	2.852	36,13
13) ZTD, pressure, relative humidity and temperature, clusters on normalised coordinates, 1 sensor for each cluster	0,79	0,81	0,70	141.960	3.302	42,99
14) As above; -lead 6 -lag 6	0,80	0,80	0,70	137.022	3.542	38,68
15) As above; -lead 6 -lag 9	0,77	0,80	0,70	132.442	962	137,67

Figure 8. Experiments performed during the project.

Case study

Figure 8 highlights that the wind direction and wind intensity, when assimilated together with the ZTD, produce a decrease of the NN performances. This is rather unusual since from the literature we know that the wind field is a relevant parameter to forecast the weather. To understand this issue, we decided to deepen the analysis on wind field and ZTD and to select some case studies. In fact, out of the >600 cases selected we have found a recurrent ZTD increase combined to a convergent wind field preceding the precipitation, and the sharp decrease of ZTD just before the maximum intensity.

We show here a case study developed on the 16th of June 2016 over Milan (Italy) well characterising the typical convective environment of the area. The 16th of June the synoptic and mesoscale weather analysis shows an expansion towards the high Atlantic latitudes of Azores ridge. This allows the descent along its eastern edge of an Atlantic trough toward the western Mediterranean driven by a low-pressure system on the northern Europe. The geopotential maps show that the northwestern part of Italy is located on the lifting part of the depression, characterised by differential potential vorticity advection and ascending speed velocity, enhanced by jet stream passage and consequently upper-level divergence. Heat and moisture at low level (from 925 to 850 hPa pressure level) are advected towards Po valley, while dry air is entering above 700 hPa pressure level. Consequently, the atmospheric profile is unstable and favorable to convection over Milan.

The orography of the Po valley, with southern wet stream, allows the accumulation of potential energy on the northern edge and maximum values of Convective Available Potential Energy (CAPE) close to the pre-Alps. In contrast, in the southern Po valley is present drier air in the lower layers, with winds from the Ligurian-Emilian Apennines (dryline) inhibiting the convection. The boundary between the two air masses (over Milan) is the area with the greatest probability of developing convection. At 15 local time, a few storm cells developed along the dryline and got organised in a squall line system thanks to a strong vertical wind shear environment.

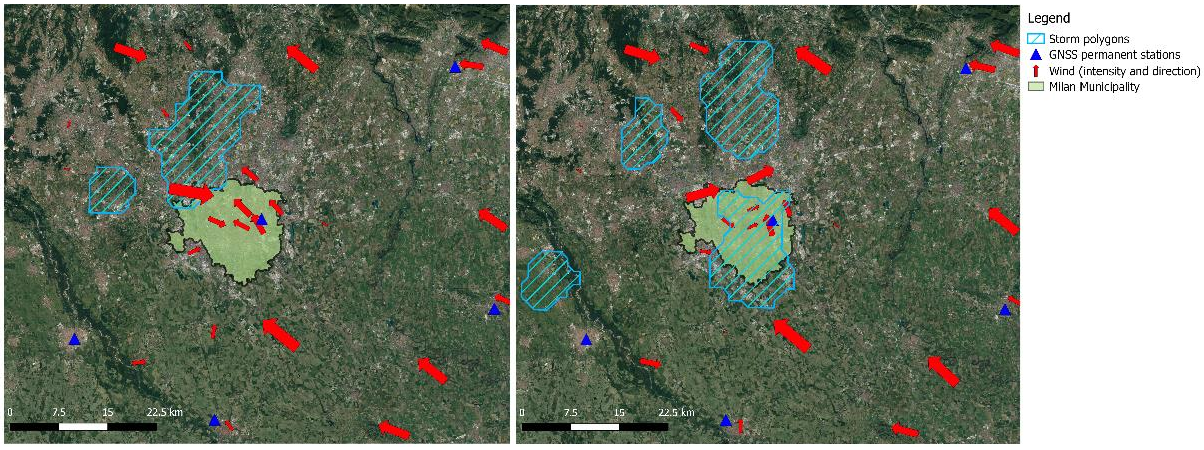

Figure 9. Left: Convergence wind fields (red arrows) 30 minutes before the rain starts (15:30 local time). Right: Convergence wind fields (red arrows) 30 minutes before the maximum rain rate (16:15 local time).

With a dense network of weather stations in the area, we computed wind convergence at the ground 30 minutes before the development of the first storm (Figure 9 left) and 30 minutes before the maximum rain rate (Figure 9 right) and we overlapped it with storm cells which occurred later. In condition of instability, if there is a trigger mechanism of lift, air parcel can maintain its positive buoyancy and thus vertical motion. In this case study, atmosphere was unstable over Milan and the

wind field area of convergence at low levels was the trigger mechanism to lift air, so that thunderstorms could develop. In Figure 10 we show the ZTD trend estimated by the GNSS station in Milan and the rain rate measured by the closest weather station. In this specific case, it is possible to notice an increase in the ZTD during the convergence phase and subsequently a sharp decrease that precedes the passage of the storm cell on the GNSS station. However, in the ZTD time series there are other variations which can't be attributed to the storm so the ZTD alone is not able to provide sufficient information to the storm development.

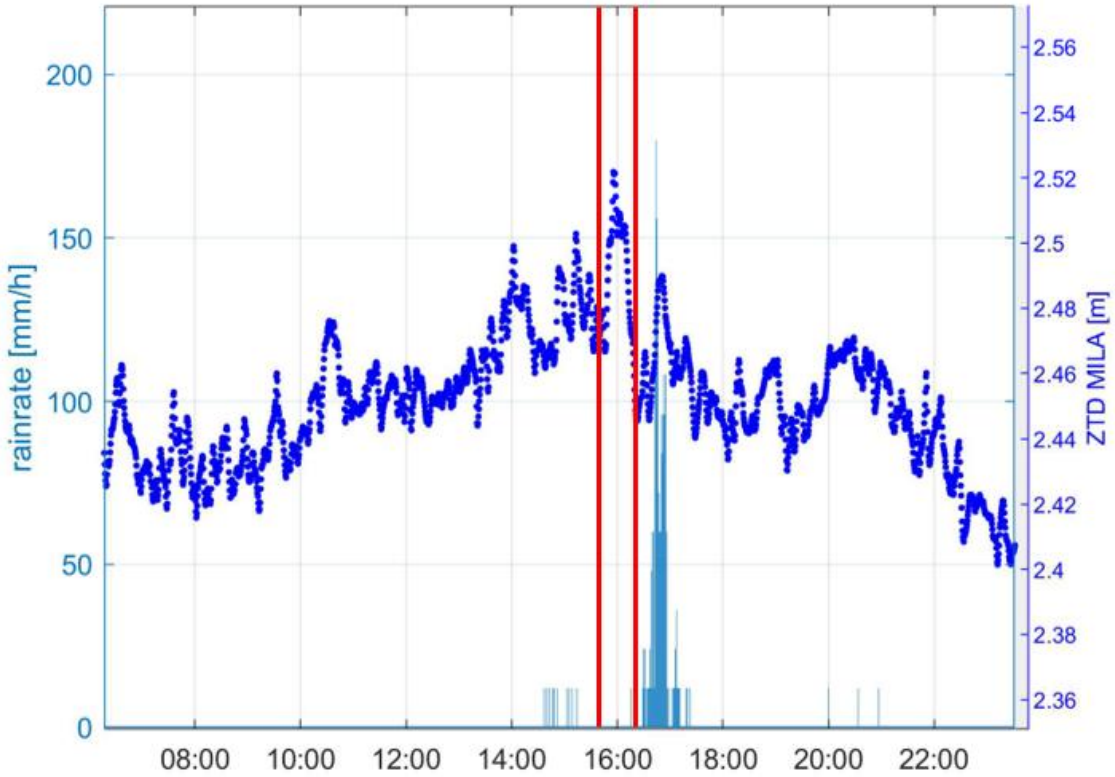

Figure 10. Zenith Total Delay time series before and after the convection (dark blue) from the closest GNSS station and corresponding rain rate (light blue). In red the time steps corresponding to 30 minutes before the rain starts (Figure 9 left) and 30 minutes before the rain rate maximum (Figure 9 right).

A statistical analysis based on 40 case studies (Table 3) randomly chosen in the period 2010-2018, shows that 78% of the extreme events are characterised by wind field convergence, just 25% of the cases have a decreasing trend of ZTD 30 minutes before the rainfall, and there is never decreasing ZTD trend 30 minutes before the event corresponding to wind field divergence. It is interesting to notice that all the cases (10 in total) with a sharp decrease of ZTD (and consequently IWV) just before the maximum intensity are also characterised by convergent wind field.

	Wind field converge	Wind field diverge
ZTD trend increase	11	0
ZTD trend decrease	10	5
No clear trend	10	4

Table 3. Statistical analysis showing the ZTD trend 30 minutes before the rainfall event vs the wind field convergence/divergence.

This is a preliminary analysis to understand the best predictors for locally developed severe convection. Despite the dataset is still under investigation (more than 600 extreme events selected), we believe that the analysis of 40 cases is already statistically relevant to lead to some conclusions. Two important results must be highlighted:

- 25% of the analyzed cases show a sharp decrease of ZTD before the event always corresponding to low-level convergence during the initiation of the convective cell;

- 100% of the cases with wind field divergence never show an increasing trend of ZTD 30 minutes before the event.

These results show, as expected, that the wind is an important parameter to be considered to nowcast the extreme weather events in the area of interest. But this is not consistent with the results shown in the previous section where the performance of the NN decreases with the use of ZTD and wind information. The reason of the inconsistency can be due to an incorrect way to provide this information to the neural network, and this will be a topic that must be deepen in future studies.

3. Conclusions, next steps and lessons learned

3.1 Conclusions

The CARGO project has been an interesting collaboration between academic institutes and industry represented by a SME (GReD srl) and a large enterprise (Leonardo Gmbh). GReD srl directly developed and managed the GNSS sensors fundamental for this project and had a vital role to install the lightning detector at Campo dei Fiori. Leonardo Gmbh has been involved in the latest period to validate the results, however the validation is still on-going while this report was written.

The final results were not reached in the expected time mainly due to the COVID-19 crisis delaying some activities. We were not able to assimilate into the final algorithm all the datasets we planned to include (i.e. the radio occultations) because the spatial and temporal resolution of the acquisitions were too coarse to fit with the other parameters.

However, we were able to develop an efficient algorithm which can be optimised in the near future and reused in different locations for other experiments. CARGO was the first experiment collecting so many atmospheric sensors in a localised area to nowcast extreme events with ATM purposes. Several issues were highlighted during the project which will be the background for future studies and projects. The project can be considered successful since it posed the bases for the planning and development of 2 new larger SESAR projects (SINOPTICA and ALARM). The algorithm developed during CARGO will be the starting point for the studies of ALARM and the issues found during this project, will allow ALARM to get already a clear overview of the problems.

The lightning sensor which could not be used during CARGO, is now part of the Earth network inc. and will be used for the new projects.

3.2 Next steps

Even though the project is ended, the partners of the project are interested to follow the collaboration. The final product which was expected to be delivered by CARGO, the polygon with the predicted severe event to be validated by Leonardo, will be delivered in the next months.

The background of CARGO will be an important and significant starting point for the development of the new SESAR ALARM project and all the outcomes of ALARM will refer to CARGO.

Could be interesting in future to invest in proof of concept and/or operational instruments to support the aviation safety.

3.3 Lessons learned

- the sensors network where not set for this specific purpose, so the spatial distribution is not optimised and the temporal resolutions are not homogeneous;
- several different tests are required to optimised the algorithm and we did not have the time to run all the tests;
- the customised lightning detector was not installed on time for the planned campaign;
- we did not find the way to add vertical information from radio occultation to the algorithm since the temporal and spatial resolutions are too coarse in comparison to the other measurements;
- we ran the algorithm for each single stations but we did not have time yet to make the software to create the final polygon which must be compared to the Ground Weather Management System (GWMS) provided by Leonardo.

Several issues were highlighted during the project which will be the background for future studies and projects. Within this project, other than the real results of the nowcasting exercise, we have understood that:

- A local characterisation of the convection must be done before developing the algorithm, since the atmospheric characteristics of each single hotspot could be different and making the algorithm more complex;
- The choice of training, validation and test dataset could be crucial for the final results. In our experiments we have used as validation and test all the events form a specific year (respectively 2018 and 2019), however, a single year would have specific atmospheric characteristics different than those of the training and a random choice would provide different results;
- We must reach a significant number of cases to train the neural network and this means that the threshold to define the extreme must be tuned according to the objective of the work;
- The definition that we gave to the “extreme”, could not be adequate for ATM/ATC purposes and this should be better re-defined according to other parameters;
- When working with sensors/datasets coming from different networks, we must be aware that they have not been deployed for this specific use (some were dismissed during the period of interest some new were installed, several technical issue can happen …) and this decreases the number of events that can be used for the analysis. It is important to perform a strategy to fill the gaps;
- We have finally seen that the density instruments of each network allows us to get different performances so it is important to understand what is the minimum number of sensors (weather stations, ground based GNSS, lightnings detector) allowing high performances of the nowcasting algorithm and at the same time optimising the costs.

About the SEASR Engage KTN catalyst call:

- The Engage KTN catalyst funding is a very interesting initiative, being more flexible than larger calls and manageable with less bureaucracy, it is advisable to keep such kind of call in future;
- The period of 1 year allows to study some research feasibility but it is really difficult to complete a specific research on a such short period. In my field it is almost impossible to

publish a paper in less than 8 months, so 1 year is not enough time to prepare the research and make it public. I would suggest giving the possibility to the applicant to use the budget for a longer period.

4. References

4.1 Project outputs

- E. Solazzo, **P.-Y. Tournigand**, S. Barindelli, V. Guglieri, E. Realini, L. Nisi and **R. Biondi**, "Understanding severe weather events at airport spatial scale", 2020 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Virtual Symposium, 26 September- 2 October 2020
- **R. Biondi** and S. Corradini, "A European Network for Extreme Atmospheric Events Detection and Monitoring", EGU 2020 General Assembly, Vienna, Austria, 4-8 May 2020, EGU2020-20601, https://doi.org/10.5194/egusphere-egu2020-20601
- **R. Biondi**, **P.-Y. Tournigand**, E. Solazzo, E. Realini, C. Cimarelli, and S. Kauczok, "The airport-sCAle seveRe weather nowcastinG prOject (CARGO)", EGU 2020 General Assembly, Vienna, Austria, 4-8 May 2020, EGU2020-15177, https://doi.org/10.5194/egusphere-egu2020-15177
- M. Sangiorgio, S. Barindelli, **R. Biondi**, E. Solazzo, E. Realini, G. Venuti and G. Guariso, A Comparative Study on Machine Learning Techniques for Intense Convective Rainfall Events Forecasting. In: Valenzuela O., Rojas F., Herrera L.J., Pomares H., Rojas I. (eds) Theory and Applications of Time Series Analysis. ITISE 2019. Contributions to Statistics. Springer, Cham. https://doi.org/10.1007/978-3-030-56219-9_20, 2019.

Annex

Neural Network Python code

```
### IMPORT ----------------------------------------------------------------------------
input_tuple = pd.read_csv('D:/CARGO/dati/tentativo_1/input_tuple.txt', header=0)
output_tuple = pd.read_csv('D:/CARGO/dati/tentativo_1/output_tuple.txt', header=0)
val_test = pd.read_csv('D:/CARGO/dati/tentativo_1/val_test.txt', header=0)
variabile = np.loadtxt('D:/CARGO/dati/tentativo_1/variabile.txt', dtype='str').tolist()

input_train = input_tuple.values[:-(val_test.validation.values[0] + val_test.test.values[0]),:]
output_train = output_tuple.values[:-(val_test.validation.values[0] + val_test.test.values[0]),:]

input_val = input_tuple.values[-(val_test.validation.values[0] + val_test.test.values[0]):-
(val_test.test.values[0]),:]
output_val = output_tuple.values[-(val_test.validation.values[0] + val_test.test.values[0]):-
(val_test.test.values[0]),:]


# Initialize tensor for inputs, and outputs
x = torch.from_numpy(input_train.astype('float32'))
y = torch.from_numpy(output_train.astype('int64')).squeeze()

x_val = torch.from_numpy(input_val.astype('float32'))
y_val = torch.from_numpy(output_val.astype('int64')).squeeze()


### NET DEFINITION ---------------------------------------------------------------------
# Create a function to train the network
def rete_neurale(loop):
   # Create a model
   if n_l == 1:
      model = nn.Sequential(nn.Linear(n_in, n_h),
                        nn.ReLU(),
                        nn.Linear(n_h, n_out),
                        nn.Sigmoid(),
                        nn.Softmax(dim=1))

   elif n_l == 2:
      model = nn.Sequential(nn.Linear(n_in, n_h),
```

```
                nn.ReLU(),
                nn.Linear(n_h, n_h),
                nn.ReLU(),
                nn.Linear(n_h, n_out),
                nn.Sigmoid(),
                nn.Softmax(dim=1))

elif n_l == 3:
    model = nn.Sequential(nn.Linear(n_in, n_h),
                nn.ReLU(),
                nn.Linear(n_h, n_h),
                nn.ReLU(),
                nn.Linear(n_h, n_h),
                nn.ReLU(),
                nn.Linear(n_h, n_out),
                nn.Sigmoid(),
                nn.Softmax(dim=1))

# Construct the loss function
criterion = torch.nn.CrossEntropyLoss(weight=pesi)

# Construct the optimizer (Stochastic Gradient Descent in this case)
optimizer = torch.optim.SGD(model.parameters(), lr=learning_rate)

# Initialize the lists to store the loss values
loss_list = []
val_loss_list = [5]*350 + [1]*150

# Start timing
st = time.time()

# Gradient Descent
for epoch in range(n_epochs):
    # Forward pass: Compute predicted y by passing x to the model
    y_pred = model(x)

    y_pred_val = model(x_val)

    # Compute and print loss
    loss = criterion(y_pred, y)
```

```
        loss_list.append(loss.item())

        val_loss = criterion(y_pred_val, y_val)
        val_loss_list.append(val_loss.item())

        # Early stopping
        if np.array(val_loss_list[-500:]).std() <= threshold:
            print('Early stopping on epoch:', epoch)
            break

        # Zero gradients, perform a backward pass, and update the weights.
        optimizer.zero_grad()

        # perform a backward pass (backpropagation)
        loss.backward()

        # Update the parameters
        optimizer.step()

        if epoch%500 == 0:
            print('epoch:', epoch, 'loss:', round(loss.item(),4), 'v_loss:', round(val_loss.item(),4))

    del(val_loss_list[:500])

    # Saving model and loss functions
    torch.save(model.state_dict(), 'D:/CARGO/dati/tentativo_1/t1_'+str(loop)+'_m.pt')
    pd.DataFrame({'train_loss':loss_list,
'valid_loss':val_loss_list}).to_csv('D:/CARGO/dati/tentativo_1/t1_'+str(loop)+'_loss.txt')
    metadati = {'n_in':n_in,'n_h':n_h, 'n_layer':n_l, 'n_out':n_out,'lr':learning_rate,'n_ep':epoch,
'weights':pesi, 'input':variabile}
    torch.save(metadati, 'D:/CARGO/dati/tentativo_1/t1_'+str(loop)+'_metadati.txt')

    # Ending timing
    et = time.time()
    print('time: ', round((et-st)/60, 2), 'minuti')



### HYPERPARAMETERS AND TRAINING ------------------------------------------------------------
# Definition of hyperparameters
n_in = input_train[0].size      # input layers
```

```
n_out = 3                  # output layers
n_epochs = 100000          # number of epochs (expected to produce a good result)
threshold = 0.00001        # early stopping threshold on validation loss function

# Weight of each class, used in the cross-entropy loss computation
pesi = torch.Tensor([len(output_train)/(output_train==0).sum(),
                     len(output_train)/(output_train==1).sum(),
                     len(output_train)/(output_train==2).sum()])

lista_h = [5, 10, 20, 30]          # number of neurons
lista_layer = [1,2,3]              # number of layers
lista_lr = [0.03, 0.01, 0.003, 0.001]   # learning rates
loop = 0

# Training
for n_h in lista_h:
   for n_l in lista_layer:
      for learning_rate in lista_lr:
         print('\nLoop:',loop)
         rete_neurale(loop)
         gc.collect()
         loop += 1
```