A tool for emergency detection with deep learning neural networks

Emanuele Cipolla, Riccardo Rizzo, Dario Stabile, Filippo Vella

Institute for High Performance Computing and Networking - ICAR
National Research Council of Italy - Palermo, Italy
emanuele.cipolla@icar.cnr.it,riccardo.rizzo@icar.cnr.it
dario.stabile@icar.cnr.it, filippo.vella@icar.cnr.it

Abstract. The ubiquitous presence of sensor networks, control units and detection devices allows for a significant availability of data. The increased computational power also encourages a wider development of deep neural networks that represent data in multiple levels of abstraction. In this contribution we present a tool that process the daily precipitation amount in Tuscany region and the emergency situations reported in web news, in order to detect emergency situations. The results are encouraging and show how machine learning can help in predicting emergency situations and to reduce the impact of critical situations.

1 INTRODUCTION

The possibility to collect and store large amount of data captured by sensor networks enables the search of connections among data and the effects of natural events that generate great damages to people and things. Here we aim at detecting emergency situations processing data sampled by a set of pluviometers through Deep Convolutional Neural Network. A recent survey about Deep Neural Networks has been published by Schmidhuber [1]. Many different application domains are taken into account in the work, with the notable exception of meteorological emergency alerts and risk. Kang et al. proposed a system for the emergency alert system based on a deep learning architecture that takes as input images from closed circuits camera images and detects events bound to fire or car accidents [2]. Basha et al. use data from a sensor network to predict river flood through linear regression models [3]. The emergency situations can also be characterized as outliers in a network of sensors analyzed as a minimum spanning tree [4].

In this work we considered the data from a set of pluviometers and we desire to assess if the given pattern in the input will produce a sort of emergency or not. We use a new deep convolutional architecture and compared it with a more traditional neural network as the multilayer Perceptron(MLP) in order to understand whether there are applications that require one or the other kind of network in the field of time-series processing.

We trained the networks using freely available measurements by Servizio Idrogeologico Regionale della Toscana (SIR)[1], gathered by sensor networks with emergency notifications commonly found online newspapers and weblogs. We tested the technique on a dataset of Tuscanian meteorological data ranging from 2012 to 2014, and we have compared these values with the emergency detection in the same region along the same years, with promising results. [5]

The paper is organized as follows. The next section presents a description of Deep Learning Neural Networks. In Section 3 we describe the approach used for the classification of emergencies and the operations of pre-processing for the construction of the dataset, in Section 4 the experimental results are presented. Finally, in Section 5 discusses the future directions of this work.

2 NEURAL NETWORKS FOR EMERGENCY CLASSIFICATION

Deep-learning methods typically employ from 5 to 20 non-linear modules that extracts a set of features from the input and transfer them to the next module. [6] The weights of the layers of features are learned directly from data, allowing to discover intricate structures in high-dimensional data, regardless of their domain (science, business, etc.). With this mechanism very complex functions can be learned combining these modules: the resulting networks are often very sensitive to minute details and insensitive to large irrelevant variations.

2.1 MLP Multilayer Perceptron

A multilayer perceptron (MLP) is a feedforward network that maps sets of input data onto a set of appropriate outputs; it consists of at least three layers of fully connected nodes in a directed graph: an input layer, an hidden layer and an output layer. Except for the input nodes, each node is a neuron (or processing element) with a nonlinear activation function - usually a sigmoid, or the hyperbolic tangent, chosen to model the bioelectrical behaviour of biological neurons in a natural brain. Learning occurs through backpropagation algorithm that modifies connections weights in order to minimize the difference between the actual network ouput and the expected result.

2.2 CNN-Convolutional Neural Network

Convolutional Neural Networks (CNN) are a variant of multilayer perceptrons inspired by visual mechanisms found in living organisms, where arrangements of dedicated cells of very different complexities are found: each one of them is assigned to small, overlapping sub-regions of the visual field. This processing behaviour can be reproduced using a convolutional filter over a given signal, hence the name of the network configuration.

[1] `http://www.sir.toscana.it/`

With respect to MLP, neurons in CNN are arranged in three dimensions, and only some of the layers they form are fully connected to each other; connectivity patterns reflecting spatial local correlation are actively sought. Moreover, to achieve translational invariance - very useful in image processing - each filter is replicated across the entire visual field.

In a CNN we can find three kinds of layers:

- **Convolutional layers** are driven by a set of learnable filters with very specific purpose spanning over the whole input matrix. The convolution of each filter across the input gives an activation map as output that is used to determine which feature is detected by a given neuron;
- **Pooling layers** perform a non-linear downsampling to progressively reduce the spatial size of the representation in order to use less parameters and computations and prevent overfitting. It is common to use a pooling layer between 2 convolutional layers: the dropout technique is often used.
- **ReLU layers** increase the non-linearity of the decision function by applying the activation function $f(x) = max(0, x)$. Other functions may be used.
- **Fully connected layers** are stacked after several nonlinear layers to perform high-level reasoning

3 EMERGENCY DETECTION THROUGH CLASSIFICATION

A neural network-based approach is presented for the detection of emergency situations, through rainfall level measurements. The approach is based on the training of a neural network with a set of data relating to the rainfall level measurements coming from a network of sensors, together with a series of emergency notifications that are commonly found in online newspapers and weblogs. A neural network uses a mathematical pattern recognition paradigm to learn complex interactions between inputs and outputs. The purpose of the implemented neural network is to detect potential risk situations, taking the rainfall levels as input.

The methodology is applied to actual data obtained from a set of hundreds of meteorological stations placed in Tuscany, made available by SIR-Toscana in 2012-2014 period. This sensor and surveillance network, can provide both real-time and historic samples from hydrometric, pluviometric, thermometric, hygrometric, freatimetric and mareographic sensors, allowing a general characterization of hydroclimatic phenomena.

In this work we focused on data relating to the rainfall levels. We assumed that traces of past emergency situations can be found in the World Wide Web as online newspaper articles, forums or personal blog. We collected two set of words that we used to compose the queries in the web: the set A is formed by key words about hydrogeological emergencies such as: *esondazione* (overflow), *violento temporale* (cloud burst), *diluvio* (deluge), *allagamento* (flooding), *inondazione* (flood), *rovinosa tempesta* (severe storm), *violento acquazzone* (violent downpour); the set B is formed by the names of the cities in the Tuscany region

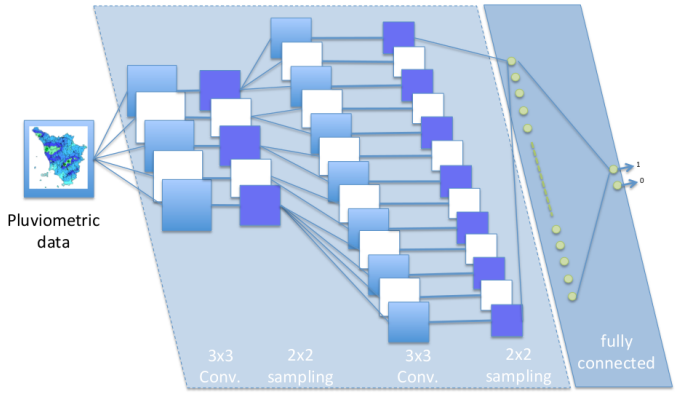


Fig. 1: Deep Neural Network architecture for Emergency classification

such as: Firenze, Pisa, Livorno, Grosseto, Lucca, Siena, Massa, Carrara, Pistoia. We have automatically queried the Bing™search engine, through its Search API, using keywords in the set given by the Cartesian product of the set A for the set $B, C = A \times B$. We dated the resulting pages using a supervised approach, after duplicate URLs had been removed. An approach based on regular expressions applied over the text-only description extracted by the Bing bot has first been used, with mixed outcomes. Excluding erroneous and mixed matches, there was no real guarantee that the writers would not have altered a date for their own reasons so, a further control based on the Last-Modified HTTP header has been used. Finally, we used the subset of search results that employed so-called *pretty* URLs, in particular those with day, month and year information separated by forward slashes, as they require a little more expertise to get altered after publication.

To define the pattern classification problem, we have arranged a set of input vectors as rows in a matrix. Each row contains a label for the input sequence (typically its detection date), a k-day-long sequence of measurements for all the stations in a given area, a label to indicate emergency/not emergency and a label to identify the quadrant in which emergency is verified.

In order to test the behaviour and efficacy of the Convolutional Neural Network, 4 different experiments were performed:

- The first experiment, called "original" in the following section, uses raw data without any preprocessing;
- The second experiment uses a "balanced" version of the original dataset having an equal number of emergencies and "quiet days". This dataset has been randomly formed discarding a set of negative days in order to have a comparable number between the positive and negative examples.

– The third,"quantized" experiment, was performed on the result of the use the Adaptive Extended Local Ternary Patterns (AELTP) quantization algorithm on the original dataset.This processing tends to enhance the differences among near values and highlight the derivatives [7].
– The fourth and final "balanced-quantized" experiment of tests combines the balancing technique of the second experiment and the quantization of the third.

To implement the above architectures and perform the tests, we used Keras, an high-level Python neural networks library, capable of running on top of two of the most important libraries for numerical computation used for deep learning: TensorFlow and Theano. The use of higher level libraries like Keras allows developers and data scientists to rapidly produce and test prototypes, while relaying most implementation details to the chosen lower level library.

Modular structure facilities to build both convolutional networks and recurrent networks and these are available as well as combinations of the two. The models built with Keras are understood as a sequence of modules like neural layers, cost functions, optimizers, initialization schemes, activation functions and regularization schemes, that can be plugged together.

The convolutional neural network we adopted takes as input the collection of the pluviometric data for a single day in form of a matrix. In fig.1 is shown a schematic representation of the network.

The net architecture has two main convolutional stages followed by the subsampling and a fully connected stage. The first convolutional stage is performed with thirty two kernels with size 3x3 followed by a processing with rectified linear units and a dropout with parameter equal to 0.25. The second convolution stage is performed with sixty four kernels with size 3x3 followed by a processing with rectified linear units. Before the fully connected step a dropout with parameter equal to 0.25 is performed. The last stage of the net, with fully connected units is formed by linear rectified units followed by a dropout step with parameter equal to 0.5 and a set of units with softmax activation function.

4 EXPERIMENTAL RESULTS

Experiments were conducted using different batch sizes in the training phase to evaluate if the training is stable versus different training settings. The experimental results are reported in an tabular form in tables 1 and 2 where the results in terms of Accuracy, Precision, Recall and F1 score. For these quantities the mean and the variance, while the batch size varied, have been calculated. In 1, the experiments with the original dataset and the dataset with quantized values, obtained through [7] algorithm, are compared. The results with the original dataset are better than the results with the "quantized" dataset for both networks. The CNN does not detect any emergency (positive) sample showing that this processing is negative for the learning performance of these models. Accuracy is high for both the models and also for the experiment with the quantized dataset. Somehow, since the dataset is strongly asymmetrical, this parameter

		Original dataset				Quantized dataset			
		CNN		**MLP**		**CNN**		**MLP**	
		μ	σ	μ	σ	μ	σ	μ	σ
Batch size=3	*Accuracy*	93.3%	0.3%	92.1%	0.5%	0%	0%	89.4%	1.2%
	Precision	90.0%	31.6%	42.1%	5.6%	0%	0%	10.0%	2.5%
	Recall	7.4%	4.4%	21.6%	3.0%	0%	0%	5.3%	0.0%
	F1 score	13.4%	7.6%	**28.3**%	2.9%	0%	0%	6.8%	0.7%
Batch size=6	*Accuracy*	93.1%	0.2%	92.2%	0.9%	0%	0%	89.4%	2.3%
	Precision	80.0%	32.2%	43.5%	11.1%	0%	0%	11.6%	3.3%
	Recall	6.8%	3.6%	20.0%	4.2%	0%	0%	5.8%	1.7%
	F1 score	12.4%	6.0%	**27.2**%	5.8%	0%	0%	7.4%	1.1%
Batch size=9	*Accuracy*	93.1%	0.4%	92.9%	0.6%	0%	0%	88.4%	3.9%
	Precision	62.5%	37.1%	54.3%	10.0%	0%	0%	11.0%	2.5%
	Recall	7.9%	5.7%	22.6%	2.5%	0%	0%	6.8%	3.6%
	F1 score	13.7%	9.5%	**31.8%**	3.9%	0%	0%	7.7%	1.0%
Batch size=12	*Accuracy*	93.3%	0.4%	91.1%	3.8%	0%	0%	88.5%	4.7%
	Precision	87.5%	31.7%	44.8%	21.0%	0%	0%	11.3%	2.9%
	Recall	8.4%	5.7%	21.6%	3.9%	0%	0%	7.4%	6.7%
	F1 score	15.0%	9.4%	**28.2**%	9.0%	0%	0%	7.8%	2.1%
Batch size=15	*Accuracy*	93.6%	0.3%	91.1%	2.9%	0%	0%	89.7%	1.4%
	Precision	96.7%	10.5%	42.7%	23.5%	0%	0%	11.6%	4.1%
	Recall	12.1%	3.6%	20.0%	4.2%	0%	0%	5.8%	1.7%
	F1 score	21.3%	5.8%	**26.3**%	9.1%	0%	0%	7.6%	2.3%
Batch size=18	*Accuracy*	93.4%	0.3%	93.0%	1.0%	0%	0%	90.2%	0.6%
	Precision	93.3%	14.1%	56.4%	13.2%	0%	0%	11.8%	2.1%
	Recall	11.1%	3.9%	21.1%	3.5%	0%	0%	5.3%	0.0%
	F1 score	19.5%	6.4%	**30.5**%	5.5%	0%	0%	7.2%	0.4%

Table 1: CNN and MLP performance with Original and Quantized dataset

loses importance as the overwhelming number of negative samples hides the performance on the limited set of positive samples. It can be seen that the MLP network has a better performance than the CNN, obtaining a maximum peak of F1 score of 31.8% with batch-size equal to nine.

Although the performance of the MLP decreases a using quantized dataset (third experiment), its performances are still higher compared to CNN that has not been able to recognize any true positive and false positive, because the network did not properly learn.

Result of the other experiments are shown in table 2. Using the balanced dataset, with a number of positive samples equal to the number of the negative samples that have been randomly reduced, the CNN network always has better performance than the MLP network, obtaining a maximum peak with batch-size equal to twelve.

Considering the F1 measure alone, a synthetic value for the result the value of 71.9% is the highest value for all the experiments and let us draw the consequence

		Balanced dataset				**Balanced-Quant. dataset**			
		CNN		**MLP**		**CNN**		**MLP**	
		μ	σ	μ	σ	μ	σ	μ	σ
Batch size=3	*Accuracy*	72.9%	3.6%	66.7%	12.3%	58.1%	3.9%	52.9%	7.3%
	Precision	88.1%	9.1%	75.3%	16.9%	36.9%	40.6%	49.7%	8.3%
	Recall	47.4%	9.9%	54.2%	12.7%	13.7%	15.9%	48.4%	5.4%
	F1 score	**60.7**%	8.1%	60.1%	4.7%	19.1%	21.0%	48.3%	2.8%
Batch size=6	*Accuracy*	73.3%	2.9%	65.2%	13.3%	57.1%	4.6%	52.4%	6.1%
	Precision	78.8%	6.5%	75.8%	18.3%	43.4%	26.6%	48.6%	6.3%
	Recall	57.4%	10.7%	52.6%	16.5%	17.9%	16.1%	47.9%	7.2%
	F1 score	**65.6**%	6.2%	58.0%	5.5%	24.3%	19.8%	47.6%	3.6%
Batch size=9	*Accuracy*	73.8%	3.7%	68.1%	11.0%	54.3%	2.5%	54.5%	5.8%
	Precision	82.6%	8.0%	77.3%	13.8%	34.3%	47.2%	50.4%	6.0%
	Recall	54.2%	7.9%	51.1%	11.7%	3.2%	5.1%	50.5%	3.7%
	F1 score	**65.0**%	6.0%	59.4%	6.0%	5.3%	7.8%	50.2%	3.6%
Batch size=12	*Accuracy*	74.8%	2.8%	64.0%	12.6%	59.0%	2.5%	58.3%	3.2%
	Precision	72.6%	4.2%	73.8%	17.3%	63.3%	13.9%	54.6%	4.0%
	Recall	71.6%	5.7%	50.5%	16.3%	29.5%	9.0%	48.9%	7.0%
	F1 score	**71.9%**	3.4%	56.0%	5.0%	38.6%	8.8%	51.3%	4.5%
Batch size=15	*Accuracy*	76.7%	2.5%	64.3%	14.0%	54.8%	4.2%	51.0%	8.3%
	Precision	83.2%	4.4%	73.1%	17.2%	47.5%	26.7%	47.4%	8.2%
	Recall	61.1%	6.2%	52.6%	14.7%	18.9%	14.3%	48.9%	2.5%
	F1 score	**70.2**%	4.0%	57.7%	4.9%	24.8%	16.6%	47.8%	4.2%
Batch size=18	*Accuracy*	76.4%	3.3%	61.9%	15.0%	56.0%	3.2%	50.2%	8.4%
	Precision	79.5%	6.7%	69.8%	19.6%	51.9%	5.5%	46.8%	8.7%
	Recall	65.3%	3.7%	55.8%	16.1%	33.7%	12.2%	47.9%	4.6%
	F1 score	**71.5**%	3.3%	57.5%	5.8%	39.5%	12.1%	46.8%	4.5%

Table 2: CNN and MLP performance with Balanced and Balanced-Quantized dataset

that the CNN, although with an increased computation cost, has the best results for this problem. In general F1 has been chosen since its value takes into account Precision and Recall values and allows a general comparison. A classification with Support Vector Machines [8] has been done and the results are shown in table 3. The values of F1 score is increased when the balanced dataset is used. Moreover, the performance with the balanced and quantized dataset obtained a value that is comparable. In both cases the results with SVM (55.2% and 48.7%) are lower if compared with the CNN results where a balanced dataset is used showing that a linear separation is not the best solution for this problem.

Figure 2 shows the plots of the F1 measure for all experiments with the neural networks. The plots are just a more detailed version of the values shown in the above tables and they show how the best results are got the the CNN with a balanced dataset and in particular with a batch size equal to twelve but also with other values of the batch size the F1 is higher that 60% and in any

		Original dataset	Orig.-Quant. dataset	Balanced dataset	Balanced-Quant. dataset
SVM	*Accuracy*	88.9%	83.9%	69.1%	53.7%
SVM	*Precision*	22.2%	17.1%	80.0%	50.0%
SVM	*Recall*	21.1%	31.6%	42.1%	47.4%
SVM	*F1 Score*	21.6%	22.2%	**55.2%**	48.7%

Table 3: Performance of Classification of the datasets with support vector machines

case the performance of CNN overcome the values of the MLP. For all the other experiments the values obtained with MLP are more stable and the statistics of the box plot show values that are near the average value. The results of the CNN are more variable and can be also less good than the result of the MLP. The experimental set with CNN when a balanced dataset is used allows the best result and should be chosen for this kind if problems.

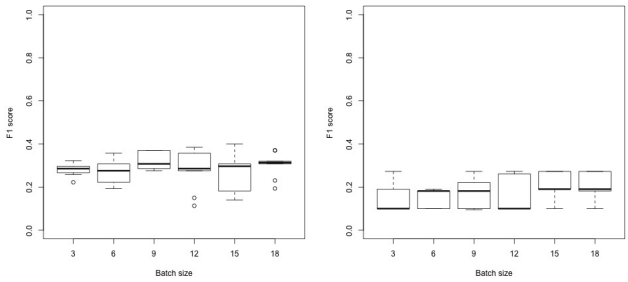
5 CONCLUSIONS

Looking at overall experimental results, we can say that, with this specific dataset and the quantization method that we have chosen, both types of neural network showed a significant loss of performance in the classification of emergencies. The best results are obtained when a reduced number of negative samples are used and the number of positive samples and the negative samples is quite similar. Since the dataset is unbalanced we cut off a set of negative samples so that the quantity of samples in the two set is equal. In this case less in more, since the results are the best we obtained limiting the overfitting over the negative samples and suitably learning the positive (emergency) cases. Moreover, in this first approach we worked with data from a wide geographic area. Given the promising results obtained, we would like to extend the tests using a larger dataset, for example relating to hourly measurements of rainfall, and using multiple parallel networks, to be able to classify the various stages that precede a possible emergency and alert a specific geographic area.

References

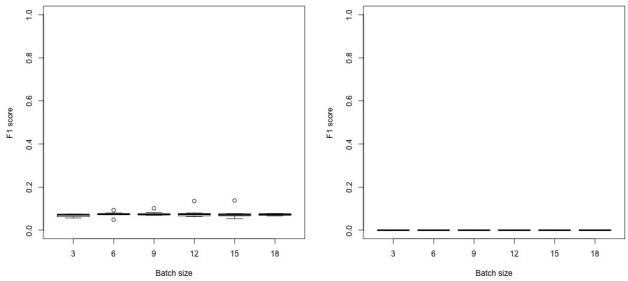
1. J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.
2. B. Kang and H. Choo, "A deep-learning-based emergency alert system," *ICT Express*, 2016.
3. E. A. Basha, S. Ravela, and D. Rus, "Model-based monitoring for early warning flood detection," in *Proceedings of the 6th ACM conference on Embedded network sensor systems*. ACM, 2008, pp. 295–308.
4. E. Cipolla and F. Vella, "Identification of spatio-temporal outliers through minimum spanning tree," in *Signal-Image Technology and Internet-Based Systems (SITIS), 2014 Tenth International Conference on*. IEEE, 2014, pp. 248–255.

5. E. Cipolla, U. Maniscalco, R. Rizzo, D. Stabile, and F. Vella, "Analysis and visualization of meteorological emergencies," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–12, 2016. [Online]. Available: http://dx.doi.org/10.1007/s12652-016-0351-x
6. Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
7. A. A. Mohamed and R. V. Yampolskiy, "Adaptive extended local ternary pattern (aeltp) for recognizing avatar faces," in *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, vol. 1. IEEE, 2012, pp. 57–62.
8. C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.



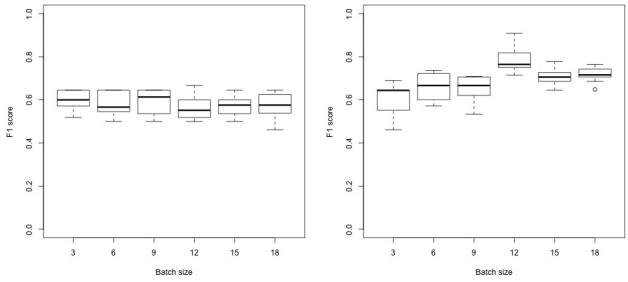
(a) MLP with Original Dataset

(b) CNN with Original Dataset



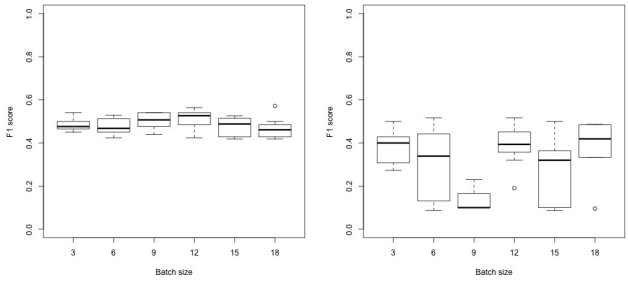
(c) MLP with Quantized Dataset

(d) CNN with Quantized Dataset



(e) MLP with balanced Dataset

(f) CNN with balanced Dataset



(g) MLP with Quantized and balanced Dataset

(h) CNN with Quantized and balanced Dataset

Fig. 2: F1 Score evaluation versus training batch sizes with different data preprocessing